Software Engineering Project (CS261)

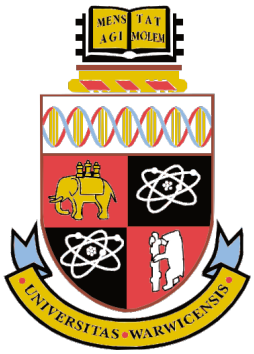
Project Group 6

Final Report

Identifying emergent behavioural patterns in the Enron corpus *via* exploratory data analysis

version 1.0, March 9, 2016

Project Team:	Zak Edwards Karim Hedna Sandeep Singh Thomas Thorpe
Project Manager:	Robert Seddon-Greve
Advisor:	Sarunkorn Chotvijit
Client:	Deutsche Bank AG



Contents

*We include here all definitions and abbreviations for the main report, in addition to appendices **A** and **B**.

Appendix

List of Program Listings[†]

[†]We omit those *complete* listings contained in Appendix **D**; only the relevant excerpts of those listings are included in the main document, and listed here.

Final Report

Identifying emergent behavioural patterns in the Enron corpus *via* exploratory data analysis

Zak Edwards*, Karim Hedna†, Robert Seddon-Greve‡,
Sandeep Singh§ and Thomas Thorpe¶

Abstract

We present the implementation of the system for which we have determined various functional and non-functional requirements (*see* Appendix **A**), together with a discussion of its relation to the design framework upon which we intended it to be built (*see* Appendix **B**). The stakeholders for the this project are the client, Deutsche Bank Group Technology & Operations (GTO), the project advisor Mr Sarunkorn Chotvijit, and Professor Stephen Jarvis. The intended audiences for this document are the aforementioned stakeholders, in addition to the representatives of Deutsche Bank to be present at the presentation of our implementation.

1 Introduction

1.1 Problem statement

The *Enron corpus*[1] serves as a snapshot of internal communications during a period in which the Enron organisation actively engaged in fraudulent, manipulative activity. As a substantial collection of historic communications data, the corpus functions as an archetypal testing ground for the algorithmic detection of manipulative and otherwise anomalous trends in large datasets.

Idealistically, such an algorithm would allow the immediate, direct detection of anomalous activity without manual intervention. However, as such algorithmic detection is constantly circumvented by malicious traders, this proves, in practice, infeasible. Thus, one must instead remain vigilant for emergent behavioural patterns, on which one can conduct further manual analysis.

1.2 Purpose of the system

Fundamentally, we aim to develop software to facilitate the investigation of *patterns* and *structures* present within communications data. The large volume of messages precludes any effective manual analysis of the raw data; thus, the principal aim of the project is to provide analysts with an efficient and accessible means of detecting patterns algorithmically determined to be *of significant interest* within an inputted dataset – specifically, the Enron corpus – and reporting these findings to the analyst in a manner understandable to those of a non-technical inclination.

{*Z.Edwards,†K.Hedna,‡R.F.Seddon-Greve,§Sandeep.Singh}@warwick.ac.uk ¶T.Thorpe@warwick.ac.uk
Department of Computer Science — University of Warwick
Department of Statistics — University of Warwick

[1] Available at `https://www.cs.cmu.edu/~./enron/enron_mail_20150507.tgz`. The corpus comprises 150 folders – containing around half a million individual files, each representing an e-mail message – corresponding to Enron employees' e-mail addresses.

1.3 Scope of the system

In order to perform the most elementary of its functions, our system must accept large datasets, precisely as large as the dataset obtained from the Enron corpus subsequent to data cleansing. Our system must identify, and report to the user, the necessary behavioural patterns as determined in **A.2.1**; further, in order to satisfy the aforestated condition of reported findings being perspicuous to those of a non-technical persuasion, we aim to provide visualisations of the data corresponding to these findings where appropriate. We intend for our system to provide its functions at the explicit request of the analyst; thus, there shall exist a strong component of interactivity to ensure the functions *executed* are precisely those functions *requested* by the user.

1.4 Structure of this document

Principally, we intend for this document to provide a cohesive overview of the system; thus, the majority of this document is dedicated to the direct discussion of the implementation and testing thereof.

The scope of this document is limited to: a discussion of *existing research*, which shall outline previous contributions to the EDA of the Enron corpus, in addition to a justification of our system's existence; a *detailed, low-level, language-specific overview* of each functional system component, in addition to the motivation for any architectural deviation from the initial planning, and a discussion of *non-functional considerations*; a thorough examination of the *testing* of our system, including the *unit*, *component integration* and *user acceptance* testing of each dominant function; a consideration of our *project management*, particularly with respect to any relevant problems encountered and the logistic solutions thereto, from initial planning to implementation; and a brief *evaluation* of the system with explicit reference to the *performance* and *reliability* of the system and the incompletion of certain system functions.

We conclude by evaluating the success of the project, in addition to considering the extensibility of the system and any feasible future advancements that may be facilitated by the current state of the system.

1.5 Definitions and abbreviations*

API *Application Programming Interface.* A set of *routines*, *protocols* and *tools* for building software and applications.[2]

Data cleansing The process of absolving an unfiltered dataset of various obstacles to the useful algorithmic extraction of patterns, minimally limited to the following quality issues: *duplicate e-mails; unconventional encoding; inconsistencies in structure and addresses; missing fields;* and *partial data.*

EDA *Exploratory Data Analysis.* The process of analysing datasets to search for and identify any *patterns* or *structures* therein.

HITS *Hyperlink-Induced Topic Search.* A link analysis algorithm introduced in Kleinberg [6]; also known as *'hubs-and-authorities'.*

UML *Unified Modelling Language.* A developmental modelling language, intended to provide a high-level visualisation of the design of a system.

2 Research

2.1 Justification of the system

There exists a wealth of literature pertaining to novel techniques for EDA on large datsets – particularly on the Enron corpus, due to its ease of availability. Despire this, our client is justified in outsourcing for this software to be developed, as many of the academic papers on EDA are wholly theoretical and do not present a framework for the adequate, concrete implementation of the detailed techniques.

[2]For the API specification for `Java`, *see* `https://docs.oracle.com/javase/7/docs/api/`.

Whilst existing software may be available to our client, many malicious traders are vigilant to the techniques these systems employ; thus, it is feasible that such traders may avoid detection. Our client has requested that our group develop a new system – that may use a slightly different EDA method – which may detect a communication pattern(s) between malicious traders, without the risk of these traders being wise to the system's architecture.

2.1.1 Previous work on the EDA of the Enron corpus

In *Extracting Hidden Groups and their Structure from Streaming Interaction Data,* Goldberg *et al.* [3] validate their algorithms on the Enron email corpus, and observe that 'analysis of the results reveals that our algorithms extract meaningful hidden group structures' (*Ibid., p.* 1). Keila and Skillicorn [5] discuss a general approach to detecting *'unusual and deceptive*' communication in e-mail, with specific application to the Enron corpus.

Wilson *et al.* [12] '[...] propose and investigate a testing based community detection procedure called Extraction of Statistically Significant Communities (ESSC). The ESSC procedure is based on p-values for the strength of connection between a single vertex and a set of vertices under a reference distribution derived from a conditional configuration network model' (*Ibid., p.* 1); we observe similarities between this approach and the application of the metrics of centrality and connectedness to determine *communities.*

Regarding this approach, in accordance with which our system at present successfully identifies communities, our principal reference in implementation was *Network Analysis with the Enron Email Corpus,* wherein Hardin and Sarkis [4] '[...] use the Enron email corpus to study relationships in a network by applying six dierent measures of centrality' (*Ibid., p.* 1). We observe similarities between this approach and the approach upon which we based our initial algorithmic developments, detailed in Mall. *et al.* [8].

Further literature upon which our initial developments were based include Brandes [1] and Rowe *et al.* [9]. Our initial developments towards a *social hierarchy* component can be found in Sims *et al.* [10].

2.2 Existing EDA libraries

Several *external libraries* were identified as having the potential to prove useful during the implementation stage of the system.

Firstly, after inspecting some of the e-mail files, it became apparent that the files had been created using a library called `JavaMail` – and, as such, it was a simple decision to use the `JavaMail API` in order to extract the relevant information from the e-mail files that those contributing towards the implementation would need. Additionally, a database library would be necessary in order to store the information extracted, which was decided, for reasons of efficiency and efficacy, to be `SQLite`.

Secondly, the `Java` library `JUNG`[3] allows for the creation and visualisation of directed and undirected graphs, which could then be used to compute both ranking in addition to employee grouping. This library would allow the developers to not have to spend valuable time implementing complicated algorithms, and instead call a function which has already been thoroughly tested and optimised.

Further, in order to read a whitelist file that exists in a `JSON` format[4], a library that allows for easy `JSON` reading was believed to be necessary. After looking at the different libraries that were available, we decided that `GSON`[5], Google's `JSON` reading and writing `API,` would be adequate for our needs.

Finally, in order to attempt to detect aliases, rather than implementing our own functions for computing the edit distances of two strings, it was decided that utilising a library (`Java-String-Similarity`) would allow labour to be focused elsewhere and not unnecessarily spent.

[3] *Java Universal Network/Graph Framework.* An open-source graph modelling and visualisation framework written in `Java`; *see* `http://jung.sourceforge.net/`.

[4] *JavaScript Object Notation.* An open-standard format that uses human-readable text to transmit data objects consisting of *attribute–value* pairs; *see* `http://www.json.org/`.

[5] 'A `Java` serialisation library that can convert `Java Objects` into `JSON` and back'; *see* `https://github.com/google/gson`.

3 Implementation

3.1 Design considerations

3.1.1 Fault-tolerant design

We have designed the system so that any application errors are handled by common fault detection services in `Java`. During any task processing, we will perform error checks on user inputs and system calculations.

If the database we've designed (see **3.3**) is deleted during runtime, the system should restrict the analyst's ability to execute functions, whilst concurrently *failing gracefully*.

3.1.2 Design conventions

The design of the system employs elements from the *object-oriented methodology*; that is, the system is designed on a *modular component* basis, which allows the *re-use* of existing components and the *sharing* of components throughout the system.

In the object-oriented tradition, we have also used existing libraries (see **2.2**) to aid in the performance and accuracy of the system.

3.1.3 Architectural design pattern

Our system will employ the *repository* architectural design pattern; we have isolated the back-end data store system layer, allowing *unit testing* (*see* **4.1**) to be conducted. The data source is accessed by the analyst from different areas of the system; thus, we employ *centrally managed* access rules and logic across all front-end components.

3.1.4 Choice of programming languages

Our implementation uses a `Java` backend and frontend, with the backend being connected to a `SQLite` database. This decision is largely predicated on the fact that all group members are confident using `Java`; thus, if any other group members were needed to assist with the implementation, it would not require them to learn a new programming language in addition to having to become familiar with the codebase. Additionally, having both a `Java` frontend and backend allows for easy packaging and distribution, in addition to being able to work offline and working on all systems that support `Java 1.8`.

The `Java` frontend was created using the `JavaFX API`, chosen over the `Swing API` due to its more appropriate components. Furthermore, it is easier to maintain given that there are plans to deprecate the `Swing API` in the next `Java` version. By creating the UI using `JavaFX`, it was easier to style and theme our frontend to create a visually pleasing user interface.

A `SQLite` database was used primarily because it is a serverless database; therefore, a separate server process was not needed when running the system. In addition, the `SQLite` file format is cross-platform, which allows the database to be transferred to a different computer and still be used.[6]

3.2 Preparation of the Enron dataset

Upon initialisation of the system, the user is required to select a *whitelist*, and a *mail directory folder* if a database is not found. The whitelist is used in all three components of the system, and acts as a way of reducing the data that is being dealt with at any one time.

In order to reduce the size of the database being created, a data cleanser was written in `Java` which prevents any duplicate e-mails from being inserted into the database. This was achieved by computing an `MD5` hash of the e-mail message contents, and using this hash as a primary key in the relevant tables. When a duplicate e-mail is inserted into the database, it will not insert the e-mail due to the unique constraint failing.

Additionally, after the first iteration of the data cleansing being performed, we noticed that there were some e-mails that had a timestamp of`01/01/1980` and some that had a timestamp after the year `2040`. Any timestamps with a year of `1980` are clearly invalid[7]; the date `01/01/1980` is used as the epoch date of certain filesystems which makes it likely that there

[6] *See* `https://www.sqlite.org/different.html`.

[7] We observe that Enron was not founded until 1985; *cf.* `www.auburn.edu/~stanwsd/enron.doc`.

was an error when converting the e-mails into text files. Naturally, the e-mails with a timestamp in the year `2040` have an invalid timestamp as that is a date in the future. As it was not possible to discern the genuine timestamp of these e-mails, we have decided to remove them so that when we display our data, it will not be influenced by these anomalies.

Certain senders and recipients of e-mails have e-mail addresses which are not valid, such as having two `'@'` symbols in their e-mail address. As these addresses are not possible to have, we have removed them from our database.

Finally, there are some e-mails that have no recipients. As such, it is a reasonable assumption that these 'e-mails' are more likely to be contact lists or reminders rather than actual e-mails, and as such, they are not required.

3.3 System architecture

In the following sections, we will discuss the implementation of the individual components, each of which map to a functional requirement in the initial *Requirements Analysis* (*see* Appendix **A**). The high-level architecture of the system, as represented in Appendix **B**, Figure **4**, remains conceptually applicable.

The ontology of the relational `SQL` database has been altered slightly since the *Design and Planning Document* (*see* Appendix **B**); Figure **1** represents the new schema design for the database architecture.

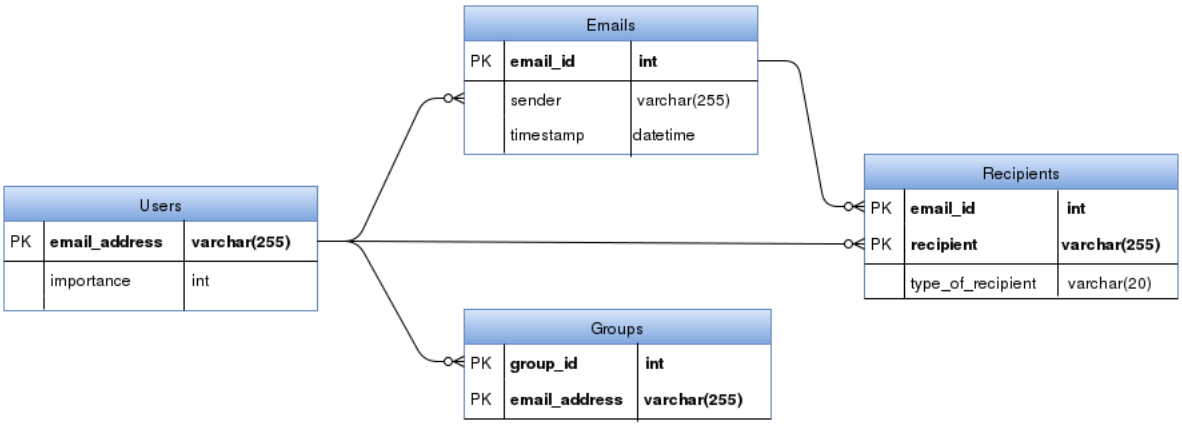


Figure 1: Schematic entity relations of the existing database

3.3.1 Detecting aliases

The detection of aliases was implemented by using the normalised **Levenshtein** algorithm[8] (*see* **Algorithm 1**) which calculates the edit distance of two strings, and then normalizes the result to provide an edit distance between one and zero.

The edit distance between all pairs of strings is calculated, and if the distance is less than `0.25`, then the two e-mail addresses are flagged as *potential aliases*. After all of the distances have been computed, a list of potential aliases is returned.

Given additional time, a more in depth algorithm would have been implemented which would use the longest common substring, in addition to the normalised Levenshtein algorithm in order to remove any false positives and prevent false negatives from occurring.[9]

3.3.2 Displaying message rates

The user is required to select an e-mail address from the provided whitelist. Once an e-mail address has been selected, the user can choose to display the *incoming*, *outgoing* or *combined* message rates by clicking the appropriate checkbox. In order to provide a greater choice, the user is able to select whether the e-mails should be grouped by *day*, *week*, *month* or *year*.

[8] Introduced in Levenshtein [7].

[9] See **Listing D.1.1** and **Listing D.1.2**, respectively, for the unabridged versions of the following listings; we omit code wherever dots (`...`) are present.

Algorithm 1: `LevenshteinDistance`(s_1, s_2)

Input: strings s_1, s_2, to be compared
variable $\varphi \leftarrow 0$
matrix $m[i, j] \leftarrow 0$
for $i \leftarrow 1$ *to* $|s_1|$ **do**
  $m[i, 0] \leftarrow i$
end
for $j \leftarrow 1$ *to* $|s_2|$ **do**
  $m[0, j] \leftarrow j$
end
for $i \leftarrow 1$ *to* $|s_1|$ **do**
  for $j \leftarrow 1$ *to* $|s_2|$ **do**
    if $s_1[i] = s_2[j]$ **then** $\varphi \leftarrow 0$;
    else $\varphi \leftarrow 1$;
    $m[i, j] \leftarrow \min\{m[i-1, j-1] + \varphi, m[i-1, j] + 1, m[i, j-1] + 1\}$
  end
end
Output: matrix $m[|s_1|, |s_2|]$

```
public class AliasIdentifier {

  DatabaseConnector conn;
  ArrayList<String> whitelistList;
  ArrayList<Set<AliasGroupObject>> listOfAliasSets;

  public AliasIdentifier(DatabaseConnector conn, ArrayList<String> whitelistList) {
    this.conn = conn;
    this.whitelistList = whitelistList;
    listOfAliasSets = new ArrayList<Set<AliasGroupObject>>();
  }

  public void identifyAliases() {
    System.out.println("Starting to identify aliases");
    long start = System.nanoTime();
    NormalizedLevenshtein normalizedLevenshtein = new NormalizedLevenshtein();
    ArrayList<AliasGroupObject> emailAddresses = new ArrayList<AliasGroupObject>();
    for (String email : whitelistList) {
      emailAddresses.add(new AliasGroupObject(email, 0));
    }
    listOfAliasSets = new ArrayList<Set<AliasGroupObject>>();
    int numGroups = 0;
    ...
    long end = System.nanoTime();
  }
  ...
}
```

The message rates are calculated by querying the database for all e-mails that include the e-mail address that has been selected, and counting the number of e-mails that were sent, received or both for the selected type of grouping. Once message rates have been calculated, the data is padded in order to provide a realistic representation of the data on days when no message have been sent or received.

E-mail addresses can also be placed into custom groups which, when displayed, show the combined incoming and outgoing message rates for all employees in the group.

The identified *stretch goal* concerns allowing the analyst to look at particular windows of time within the data. This is achieved by using two 'date picker' buttons; one to set the *lower bound* of the window of time, and the other to set the *upper bound* of the window of time. Once the bounds have been set, only data within this window will be displayed on the line chart.[10]

[10]The listing provided in the appendix (**Listing D.2**) remains abridged, due to the excessive length of the implemented function in its

```
public class MessageRatesScene {
  ...
  public ObservableList<XYChart.Data<Date, Number>> padSeriesData(ObservableList<XYChart.Data<Date, Number>>
      seriesData, int typeOfData) {
    ObservableList<XYChart.Data<Date, Number>> paddedSeriesData = FXCollections.observableArrayList();
    for (int i = 0; i<seriesData.size(); i++) {
      Date date = seriesData.get(i).getXValue();
      if (i == 0) {
        System.out.println("Adding first point");
        paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, -1, typeOfData), 0));
      } else {
        if (!paddedSeriesData.get(paddedSeriesData.size()-1).getXValue().equals(changeDate(date, -1,
            typeOfData))) {
          paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, -1, typeOfData), 0));
        }
      }
      paddedSeriesData.add(seriesData.get(i));
      if (i == (seriesData.size() -1)) {
        System.out.println("Adding last point");
        paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, 1, typeOfData), 0));
      } else {
        if (!seriesData.get(i+1).getXValue().equals(changeDate(date, 1, typeOfData))) {
          paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, 1, typeOfData), 0));
        }
      }
    }
    System.out.println(paddedSeriesData);
    return paddedSeriesData;
  }
  ...
}
```

3.3.3 Community detection

After further research into community detection algorithms, we identified that the algorithms laid out in the *Design and Planning Document* would not be appropriate for the system. Instead, *voltage clustering*[11] is used in order to detect communication patterns within the whitelist. Ideally, the **Girvan-Newman** algorithm[12] would have been used instead; however, this algorithm took too long to compute the clusters for a small number of e-mail addresses, let alone all of the e-mail addresses on the whitelist. Due to this, and the implementation needing to be responsive, we opted to select a more efficient community detection algorithm.

In order to implement the voltage clustering, a `Java` library called `JUNG` was utilised which provided methods to both compute cluster and display them. If time constraints were less restrictive, those contributing towards the implementation of this function would have implemented their own algorithm; it was believed, however, that there would not have been sufficient time to implement and fully test their algorithm.

The clusters are only computed when the whitelist changes from the previous time the system was run to reduce the loading time as the clustering algorithm can take up to one minute to complete.

In order to compute the clusters, all of the edges between members on the whitelist are retrieved from the database, and added to a directed graph. Using this directed graph, groupings are calculated using the `JUNG` library, and then inserted into the database.

When displaying the clusters, the groups are retrieved from the database, in addition to the edges between all of the members in the same group (that is, not including edges between two different groups). If there are multiple e-mails that have been sent between two employees, they are combined into one edge which, once displayed, will have an increased edge thickness. Each cluster is given a different colour to provide the analyst with another method of identifying different clusters other than there not being any edges between distinct clusters.

entirety.

[11] *See* `http://jung.sourceforge.net/doc/api/edu/uci/ics/jung/algorithms/cluster/VoltageClusterer.html`.

[12] Introduced in Girvan and Newman [2].

Once the graph is displayed, the analyst is able to select an e-mail address from the whitelist which will display a list of all the other e-mail addresses within the same group, as well as the colour of the cluster that the e-mail is in to enable the analyst to identify the cluster in the graph.[13]

```
public class GroupingScene {
  ...
  public void generateGroups() {
    Graph<String, EdgeObject> graphToIdentifyClusters = new SparseMultigraph<String, EdgeObject>();
    for (String user : whitelistList) {
      graphToIdentifyClusters.addVertex(user);
    }
    ArrayList<EdgeObject> clusterCreationEdges = conn.getAllEdgesForClusteringCreation(whitelistList);
    for (EdgeObject edge : clusterCreationEdges) {
      if (edge.getReceiver() != null) {
        graphToIdentifyClusters.addEdge(edge, edge.getSender(), edge.getReceiver(), EdgeType.DIRECTED);
      }
    }
    VoltageClusterer<String, EdgeObject> voltageClusterer = new VoltageClusterer<String, EdgeObject>(
        graphToIdentifyClusters, graphToIdentifyClusters.getVertexCount()/10);
    LinkedList<Set<String>> voltageClusters = (LinkedList<Set<String>>) voltageClusterer.cluster(
        graphToIdentifyClusters.getVertexCount()/10);
    System.out.println("There␣are␣" + voltageClusters.size() + "␣clusters,␣and␣" + graphToIdentifyClusters.
        getEdgeCount() + "␣edges");
    conn.dropGroups();
    conn.commit();
    int groupNumber = 1;
    for (Iterator<Set<String>> cIt = voltageClusters.iterator(); cIt.hasNext();) {
      Set<String> vertices = cIt.next();
      System.out.println(vertices.toString());
      for (Iterator<String> iterator = vertices.iterator(); iterator.hasNext();) {
        String email = iterator.next();
        System.out.print("Group␣number:␣" + groupNumber + "␣Email:␣" + email + "␣");
        System.out.println(conn.insertIntoGroupStatement(groupNumber, email));
      }
      groupNumber++;
    }
    conn.commit();
  }
  ...
}
```

3.3.4 Generating a corporate hierarchy

Due to time constraints and delays in the implementation, the developers and the business analyst concluded that there were other tasks to be completed of a higher priority.

3.3.5 Ranking Enron employees

In a similar fashion to the community detection, the ranking of employees is only performed on the e-mail addresses in the whitelist, and the ranks are only generated if each employee on the whitelist does not already have a ranking in the database.

Firstly, in order to rank the Enron employees, a graph is created with each e-mail address on the whitelist as a vertex, and the edges between each e-mail address are retrieved from the database. Once the graph has been created, the *centrality* of each vertex is calculated using the *betweenness centrality* algorithm (*see* **Algorithm 2**) – every possible shortest path between two pairs of vertices is calculated, and the score of each node that lies on that path is increased. Once this has been computed for each pair of vertices, the *score* of the vertex is the *total rank* of that vertex. This score is then normalised so that it is a value between `0` and `1`, with a higher value indicating a greater centrality. Once the centrality has been computed, it is stored in the database for later access.

[13] The more exhaustive listing provided in the appendix (**Listing D.3**) remains abridged.

Upon retrieval of the rankings from the database, the number of e-mails between each pair of vertices is also retrieved, which is used to determine the thickness of an edge between each pair of vertices. Additionally, the size of each vertex is proportional to the ranking of the employee, so it is easier to see who the more central employees are.

Finally, upon selection of an employee from the drop-down box, the employee's normalised ranking and their respective position within the whitelist is plainly displayed, in addition to the vertex and all edges incident to that vertex being highlighted on the graph.[14]

Algorithm 2: `BetweennessCentrality`(V, E)

Input: directed graph $G = (V, E)$
queue $Q \leftarrow 0$, stack $S \leftarrow 0$
$\forall s \in V$:
  dist$[v]$:=distance from source
  pred$[v]$:=list of predecessors on shortest path from source
  $\sigma[v]$:=number of shortest paths from source to $v \in V$
  $\delta[v]$:=dependency of source on $v \in V$
  $c_B[v] \leftarrow 0$
for $s \in V$ **do**
  for $w \in V$ **do**
    pred$[w] \leftarrow \varnothing$
  end
  for $t \in V$ **do**
    dist$[t] \leftarrow \infty$
  end
  $\sigma[t] \leftarrow 0$, $\sigma[s] \leftarrow 1$, dist$[s] \leftarrow 0$
  enqueue $s \rightarrow Q$
  while $Q \neq \varnothing$ **do**
    dequeue $v \leftarrow Q$, push $v \rightarrow S$
    foreach $w \in V : (v, w) \in E$ **do**
      if dist$[w] = \infty$ **then**
        dist$[w] \leftarrow$ dist$[v] + 1$
        enqueue $w \rightarrow Q$
      end
      if dist$[w] =$ dist$[v] + 1$ **then**
        $\sigma[w] \leftarrow \sigma[w] + \sigma[v]$
        append $v \rightarrow$ pred$[w]$
      end
    end
  end
  for $v \in V$ **do**
    $\delta[v] \leftarrow 0$
  end
  while $S \neq \varnothing$ **do**
    pop $w \leftarrow S$
    for $v \in$ pred$[w]$ **do**
      $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$
    end
    if $w \neq s$ **then**
      $c_B[w] \leftarrow c_B[w] + \delta[w]$
    end
  end
end
Output: betweenness centrality $c_B[v]$ for all $v \in V$

[14] See **Listing D.4** for the semi-unabridged version of the included excerpt.

```
public class RankingScene {
  ...
  public void computeRankings() {
    rankEmailGraph = rankgraphCreator();
    BetweennessCentrality<String, GraphEdge> ranker = new BetweennessCentrality<String, GraphEdge>(
        rankEmailGraph, true, false);

    ranker.setRemoveRankScoresOnFinalize(false);
    ranker.evaluate();

    int vertexCount = rankEmailGraph.getVertexCount();
    double maxRanking = ranker.getRankings().get(0).rankScore;
    double minRanking = ranker.getRankings().get(vertexCount-1).rankScore;

    for (int j = 0; j < vertexCount; j++) {
      String emailAddress = (String)ranker.getRankings().get(j).getRanked();
      Double normalizedRank = normalizeRankings(ranker.getRankings().get(j).rankScore, maxRanking, minRanking)
          ;
      emailRankHashMap.put(emailAddress, normalizedRank);
      conn.updateRanking(emailAddress, normalizedRank);
    }
    conn.commit();
  }
  ...
  public Double normalizeRankings(Double rank, Double maxRanking, Double minRanking) {
    return (rank - minRanking)/(maxRanking-minRanking); // Scaled ranking so min(rank)=0 and max(rank)=1
  }
  ...
}
```

3.4 Non-functional considerations

The system, as currently developed, is able to accept large datasets by storing all of the data from a mail directory in a local database. Such a configuration allows for the fast retrieval of data, and ensures that the mail directory only needs to be searched through once to populate the database.

The developers established a strong, intuitive GUI by ensuring that all buttons are appropriately labelled, in addition to providing pictures for buttons that are appropriate to the action being conducted.

As specified in the Requirements Analysis (*see* Appendix **A**), if the estimated execution time of a functional process is *greater than five seconds*, a *loading bar* should be displayed. Due to the data being stored in a database which allows for fast retrieval of data, the only functional process that necessitates a loading bar is the population of the database, which takes approximately 12 minutes to execute. Thus, in accordance with the non-functional requirements of the system, a loading bar is displayed during this process, informing the analyst how far through the mail directory the system has reached.

Any errors and exceptions are handled appropriately, as necessary for the analyst's straightforward and uninterrupted use of the system. Upon the event of a fatal exception occurring (*i.e.,* an external OS error) which would result in the system 'crashing', a log file is created and a message is displayed to the analyst asking them to send the log file to one of the developers.

As specified in the Requirements Analysis, the system is primarily designed to be usable by non-technical analysts; as such, there is no technical jargon or idiosynchratic language that the analyst is expected to be familiar with prior to using the system. In addition, a *user guide* has been created, which can help the analyst to perform a function should they be unsure about precisely how to do so.

Figure 2: The system GUI displaying message rates over time

4 Testing

This section will outline the processes used to test each of the implemented features, and document some of the key (or major) tests which had failed and how they were resolved. *The complete test plan can be found in* Appendix **C**.

4.1 Component testing

The sets of tests we have designed and documented are the individual *unit tests,* in addition to the *component integration tests.*

4.1.1 Detecting aliases

The *aliases* feature was mainly tested using unit tests and code inspection. Basic unit tests were required to ensure that the analyst can interact with the system and navigate the suspected aliases interface; this was supplemented with code inspection to ensure the implemented method for detecting aliases is (adequately) accurate.

4.1.2 Displaying message rates

To test the *message rates* feature, a variety of unit tests were used to test each of the individual components, such as *checkboxes* and *combo boxes* on the interface. The individual tested components were then grouped together and integration testing was performed on the feature as a whole. The overall aim of unit testing the message rates feature was to ensure that all fields which require user input only accept what is expected to be entered into the system. One example of a unit test performed on the message rates component, as shown below, involved testing to see if the analyst can successfully select an employee from the combo box:

	Test description	**Expected result**	**Actual result**	`Pass/Fail`
18	Can you select an employee from the combo box?	Combo box should function correctly by allowing the analyst to select an employee from the combo box.	Combo box functions as expected.	`Pass`

Once each of the individual components had been tested – principally to ensure that all user-inputted data causes the system to behave as expected – we proceeded with *component integration* testing. One example of a component integration test which we performed, and passed, was testing to see if clicking on the *'incoming e-mails'* button for the selected employee resulted in a line, corresponding to the associated data in the database, being generated on the graph. Not all component integration tests passed first time – we provide an example of such a test case below:

32	Does choosing a begin and end date to display data between (where the end date is before the begin date) alert the analyst they cannot do so?	The analyst should not be prevented from doing this.	The analyst was allowed to do this and caused the system functionality to be broken.	`Fail`

Initially, *validation features* (to compare the resultant values of the two *date picker* components on the interface) were not implemented, as attempting to do so would cause the system to lose functionality; graphs could not be displayed in such a situation. This was resolved by utilising *validation checking,* which prevents the analyst from setting the *begin date* to a date after the *current end date* value and an *end date* which was before the *current begin date* value.

Another component integration test which failed can be found in Appendix **C**, *test cases* **36** and **37**. In this instance, the test failed due to the system not correctly re-calculating the combined message rates over time, and re-drawing the new line after a new employee was added or an old employee was removed. This was resolved by ensuring the system auto-refreshes the checkbox value for the group, quickly causing the line to be re-drawn.

4.1.3 Community detection

When testing the *community detection* feature, the majority of the testing was done through code inspection; thus, there were not as many unit and component integration tests, but rather instances of verifying the *semantics* of the system were correct and the system indeed generates an accurate result.

44	Are employees on the whitelist clickable?	The analyst should be able to click on any email on the whitelist.	Employees on the whitelist can be clicked.	`Pass`

Unit testing the above feature was particularly important, as it would later be combined in a component integration test whereby clicking on an employee would show the e-mails of all members of the group (documented in Appendix **C**, *test case* **45**).

Another important component integration test was ensuring the list of e-mails in a group matched what was generated by the system and displayed in the graph. As groups could be of an arbitrarily large size (given the size of the data set), manually clicking on each individual node in a group would be tedious for the analyst. The success of this test is documented in Appendix **C**, *test case* **46**.

4.1.4 Generating a corporate hierarchy

Due to delays in implementation, the corporate hierarchy feature was not implemented by the developers and therefore could not be tested.

4.1.5 Ranking Enron employees

Testing the *employee ranking* feature was principally accomplished – in a similar way to the *community detection* feature – through code inspection, due largely to the comparative lack of user interaction with the feature (*e.g.,* compared to the *message rates* feature). One unit test, however, which had to be run, concerned selecting an employee from the *combo box of all employees*:

47	Can you select employees in the combo box?	The system should allow you to select employees from the combo box.	Employees in the combo box are selectable.	`Pass`

This test was important, as it is one of the components later involved in various component integration tests (*as shown below*). When an employee was chosen from a combo box, the system would re-draw the graph and, in the process, highlight the vertex corresponding to that employee green. In addition, the system would display an *employee rank* and *raw rank score* – the value generated by the algorithm – for the employee:

48	When you select an employee from the combo box, does it display their employee position and raw rank score?	The system should display the employee position and raw rank score of the employee selected below the combo box.	The employee's position and raw rank is displayed below the combo box as expected.	`Pass`
49	When an employee is selected from the combo box, is the graph re-drawn to show their node in the graph?	The graph should be re-drawn and the employee selected should have their node and any edges connected to it highlighted green.	The employee selected is highlighted in the graph on the right.	`Pass`

4.2 Data cleansing and data insertion

The effectiveness of the *data cleansing script* was tested by querying the `SQL` database for different cases, as documented in Appendix **C,** *test cases* **51**–**53**. This ensured no e-mails with timestamps in `1980` or after `2040` were inserted into the database, as we have ascertained their timestamps to be corrupted. It also ensured that no duplicate e-mails existed in the database – necessary to avoid data redundancy – as well as ensuring that e-mails with no recipient are removed.

4.3 User acceptance testing

In order to carry out *user acceptance testing*, we asked Mr Gavin Walker for some general feedback and, in particular, answers to the following four questions; Mr Walker's responses are detailed below each question respectively.

> *Does the system display strong, intuitive GUI design?*
>
> The user interface was easy to use. Having checkboxes and buttons clearly labelled have meant that it's obvious what the user is meant to do and what the outcome of doing those things will be. Some of the titles were not easy to read due to the colour of the background clashing with colour of the text.
>
> *Is the system easy to use without the need of technical assistance?*
>
> No assistance was needed to use the system and get the intended results.
>
> *Does the system keep the analyst informed of its status?*
>
> Every operation that was tested by myself was completed almost instantly. It was clear by fading that the chart reset when the "Reset Chart" button was pressed. It was obvious what filter rate was used as the image representing this was coloured green. The graph changed quickly in the ranking tab when a new employee was selected.
>
> *Does the system satisfy the requirements envisaged by the client?*
>
> The social hierarchy is not implemented within the system. Apart from that, all other requirements seem fulfilled from a client's perspective.

On account of the feedback from our testing, we have changed the text colour of titles from the original *black* to *white*, providing a greater contrast between the background and the on-screen text.

5 Project management

5.1 Team structure

Robert Seddon-Greve, our Project Manager and Software Developer, worked on the design of the *data cleansing script,* as well as the concrete implementation of of algorithms. In addition, he organised meeting times for our group and organised the division of labour over the remainder of the project.

Software Developer and Designer Karim Hedna worked on designing the system's UI elements, and was responsible for the concrete implementation of algorithms. Karim contributed towards the integration of the different components that were independently developed.

As Software Architect, Zak Edwards will work towards researching and comparing possible approaches to resolving the functional objectives of the system – focusing on relevant applications of the most efficient and well-documented existing methods (in particular, the utilisation of external ibraries) – as well as assisting in the implementation of algorithms. In addition, he worked on the *Requirements Analysis* and *Design and Planning Document.*

Business Analyst and Software Architect Thomas Thorpe worked on research into the algorithmic basis of the system's functions, as well as managing the direction and focus of the project. In addition, he worked on the final report and on conducting the user acceptance testing.

As Software Tester and Designer, Sandeep Singh worked on testing both individual components of the system and the integration between them. He also worked on the design of the system's UI and assisted in the implementation of the system.

5.2 Monitoring progress

Throughout the project, we met as a group at least once a week at a designated time to discuss the progress each group member had made towards the project; these meetings also provided an optimal opportunity to discuss any encountered problems, and any administrative actions that were necessary to consider as a result. All instances of these meetings were recorded on a Dictaphone, which the Project Manager used to write minutes after each meeting. These minutes were sent to the rest of the group so that any comments/suggestions made could be referenced at a future time.

Outside of the meetings, communication was primarily conducted in the form of a `WhatsApp` group, with more formal information such as seminar meetings being sent *via* e-mail.

A `github` repository, created by the Project Manager, allowed the developers to work on the implementation independently and straightforwardly merge code at a later point in time. Furthermore, the tester could flag any bugs and problems that were found during the testing stage, and thus the developers could amend any identified issues concurrently. The use of a `github` repository allowed for constant monitoring of the progress being made throughout the implementation stage and helped to keep the project on track with the Gantt chart.

In order to share and communicate our group's progress towards the documentation required for the deliverables, a `Google Drive` folder was created, which contained all the individual components necessary for the deliverables. A *check list* of sections that were unfinished was kept up-to-date, allowing our group to monitor what had been completed and what remained uncompleted.

5.3 Logistical problems and their solutions

During the first week of our implementation stage, Karim Hedna was unable to begin work on the system development for which he was partially responsible, due to illness. In order to overcome this setback, the Project Manager assigned Sandeep Singh to assume the responsibilities of the Software Developer until Karim had recovered.

Immediately before the project deadline of the first deliverable, Zak Edwards – who was instrumental to the success of the first deliverable – also fell ill and thus, whilst he was able to continue to conduct research into the algorithmic basis of the system and finalise the *Design and Planning Document*, he was unable to attend the formal meeting with our client's representatives. Nevertheless, the Project Manager and the Business Analyst were still able to attend, and could successfully present two complete documents.

The Gantt chart (*see* **B.2.4**) was designed such that any delays in the schedule could be handled by pragmatic, responsible project management. For example, there were significant delays within the implementation stage due to a lack of realisation

of the scope of the project. To overcome these delays, we had members of the team that were not scheduled to help with implementation contribute towards this effort; as a result, the consequential severity of these delays was adequately minimised to the best of our abilities.

6 Project evaluation

6.1 Performance and reliability of the system

In order to ensure a high standard of performance, those responsible for the concrete implementation of the system ensured that all implemented algorithms were optimised *via* the utilisation of external libraries; as such libraries have been exhaustively tested and revised by multiple contributors, the reliable, tractible nature of these libraries can reasonably be assured. The system's reliability has been further ensured by thorough and complete exception handling; any exceptional events that could conceivably occur are handled correctly and discretely by the system. The modular discretion of exception handling facilitates the analyst's uninterrupted experience whilst using the software.

6.2 Incomplete requirements

Of the initial functional requirements (as detailed in **A.2.1**), **C**-REQUIREMENT 3 and **C**-REQUIREMENT 5 have not been completely fulfilled by the system. The former requirement, concerning the displaying of an *n-level social hierarchy*, remains wholly unimplemented; the latter requirement, concerning *alias detection*, is *partially* satisfied by the system, insofar as potential aliases are indeed identifed and subsequently output to the analyst. It was detailed within the requirements, however, that the system should combine these potential aliases together, producing a single vertex in the generated graph network to which a single employee is associated. Due to oppressive time constraints (as discussed in **5.3**), it was decided that the implementation of other functions were of higher respective priority (in accordance with the initial requirements analysis; *see* Appendix **A**), and thus any effort to complete the implementation of this precise, auxiliary function was discontinued.

In contrast to the partial fulfillment of the latter requirement, the former requirement was not implemented at all within the system; similarly, this failure of implementation was predicated on restrictive time constraints, and the higher respective priorities of other functional requirements. The fundamental design of these functions, for which we have presented algorithms (*see* Appendix **B**, *section* **3.3**), remains applicable and could plausibly be implemented as a future update, as discussed further in **7.3**.

7 Conclusion

7.1 Comparison with initial requirements

In this section, we compare the system that has been produced to the initial client (**C**-)requirements that were agreed upon at the start of the project; any conscious deviations from the initial design, if not done so previously, shall here be justified.

7.1.1 Functional requirements

All functional requirement references can be found in Appendix **A**, *section* **2.1**.

1. **C**-REQUIREMENT: *The system shall display Enron employees' e-mail* ***message rates*** *over time.*
   A *data cleansing script* (*see* **3.2**) was implemented to eliminate anomalous and otherwise problematic e-mail message data. To facilitate useability (and satisfy the respective developer **(D**-)requirements), a method of selecting *one*, *some* or *all* employees within the generated *whitelist* has also been implemented. In direct satisfaction of the above requirement, the system makes use of a method of analysing message rates over time, for either the complete period spanned by the Enron corpus or a specific window of time decided by the analyst; further, these message rates are displayed graphically by virtue of the graphical user interface (*see* **3.3.2**). One observes *equidistant temporal intervals* along the x-axis and the *number of e-mails* along the y-axis.
   We conclude, therefore, that functional requirement 1 has been entirely fulfilled by the system.

2. **C**-REQUIREMENT: *The system shall identify and display to the analyst* ***communities*** *[...] amongst Enron employees.*
   A graph, based on the Enron corpus, is generated to display communities amongst Enron employees (*see* **3.3.3**). A *directed graph* is produced, whose vertices represent employees (specifically, distinct e-mail addresses) and whose edges represents e-mails from *senders* to *recipients*. Clusters (*communities*) are identified using an in-built library based on employee communication patterns. The system's graphical interface represents these communities visually; in addition, specific communities may be examined at the analyst's discretion.
   We observe a discrepancy between the initial design of community-detection function and the concrete implementation; whilst the initial design employs an *undirected* graph, the ultimate implementation of the function exploits a *directed* graph. It was decided that the more intuitively applicable directed graph was singularly appropriate for the implementation.
   We conclude, therefore, that functional requirement 2 has been entirely fulfilled by the system.

3. **C**-REQUIREMENT: *The system shall derive and display an n-level* ***social hierarchy****.*
   As previously discussed (*see* **6.2**), this requirement has not been met; this failure of implementation was predicated on restrictive time constraints, and the higher respective priorities of other functional requirements. In **7.3**, we discuss this function as a possible future update to the system.
   We conclude, therefore, that functional requirement 3 has *not* been fulfilled by the system.

4. **C**-REQUIREMENT: *The system shall assign a* ***ranking*** *to each Enron employee.*
   An *in-built library* has been utilised to derive, through considering the concepts of *centrality* and *connectedness*, the notion of *importance* (*see* **3.3.5**). We proceed to normalise each employee's associated importance (initially a numerical quantity) and map each employee accordingly to a level within a *ranking*, based on the designated numerical boundaries demarcating these iterative levels. A list is then straightforwardly output in *descending order of importance,* and each employee can be selected by the analyst. At present, there does not exist an option to reverse this order, and thus produce an ascending list. In addition to a plainly presented list, a graphical representation has been implemented to display the rankings of employees, relating each employee to the *communities* identified in accordance with functional requirement 2.
   We conclude, therefore, that functional requirement 4 has been *mostly* satisfied, with the exception of the trivial option to *reverse-sort* into ascending order. Following discussions with the client, it was decided that this was not a necessary feature, and the initial requirements were, in this respect, inaccurate.

5. **C**-REQUIREMENT: *The system shall identify (and make known to the analyst) instances where employees are using* ***aliases****.*
   The system successfully identifies any *aliases* that employees *may* be using; the analyst may press a graphical button to display a list of all the alias groups found by the system. There remains at present no option to combine these aliases into a single vertex, thus allowing them to be treated as representative of a single individual; in **7.3**, we discuss this function as a possible future update to the system.
   We conclude, therefore, that functional requirement 5 has been *partially* fulfilled. Combining vertices in the generated graph network to concomitantly represent a distinct individual shall be regarded as a feasible extension to the project; we omit the implementation of this function due to restrictive time constraints.

7.1.2 Non-functional requirements

All non-functional requirement references can be found in Appendix **A**, *section* **2.2**.

1. **C**-REQUIREMENT: *The system shall be capable of* ***accepting large datasets*** *as input [...] precisely as large as the Enron corpus.*
   A locally-stored `SQLite` database has been created to contain the inputted Enron corpus. The database allows large datasets – precisely *at least* as large as the Enron corpus – to be stored and accessed efficiently and responsively. The schematic attributes within the database contain enough information about the dataset to allow EDA to occur across the dataset.
   We conclude, therefore, that non-functional requirement 1 has been entirely fulfilled by the system.

2. **C**-REQUIREMENT: *The system shall display strong, intuitive* ***graphical user interface*** *(GUI)* ***design****, exhibiting consistency and accessibility.*

The graphical user interface, by virtue of which the analyst interacts with the system's modular front-end components, is consistent throughout the system; that is, an aesthetically similar interface design is used for each component. Graphical representations of each component's output are clearly visible, and are immediately obvious to the analyst as representative of each function. The interface is accessible and minimal, allowing functions to be plainly executable through the interface. Conventional user interface design elements such as *buttons*, *checkboxes* and *windows* have been used to make the interface as accessible to the analyst as possible. Through our *user acceptance testing*, we determine that the user interface is indeed user-friendly and accessible.
We conclude, therefore, that non-functional requirement 2 has been entirely fulfilled by the system.

3. **C**-REQUIREMENT: *The system shall keep the analyst* ***informed of its status****, wherever a nontrivial [...] functional process is in execution.*
   We present a number of situations, exemplary of the system keeping the analyst informed of its status. Subsequent to the selection of the *whitelist* and *output folder*, an intuitive 'loading circle' animation is displayed, informing the analyst that the system is currently processing the request. A *green tick* shall appear when a whitelist has been successfully entered; a *red cross* shall appear otherwise. When an employee is added to a group in order to calculate *message rates over time*, the analyst is notified of the employee's addition when the e-mail address appears within the group box. When the analyst clicks on a vertex within the *connectedness graph*, the employee's e-mail address appears; in the *ranking list*, when the analyst selects an employee, the employee's ranking appears and their position is highlighted within the graphical representation.
   We conclude, therefore, that non-functional requirement 3 has been entirely fulfilled by the system.

4. **C**-REQUIREMENT: *There shall be no* ***unhandled errors*** *or* ***exceptional events*** *due to incorrect or unorthodox input from the analyst.*
   Through testing, we concluded that any incorrect or unorthodox input from the analyst is handled by the system using `Java`'s in-built exception catching method. Whilst loading the Enron corpus, if a dataset is not selected, a *red cross* appears, intuitively suggesting to the analyst that the loading of the dataset was unsuccessful; the system prohibits the analyst from proceeding if a whitelist remains unselected.
   We conclude, therefore, that non-functional requirement 4 has been entirely fulfilled by the system.

5. **C**-REQUIREMENT: *The system shall be delivered to the client as* ***intuitively*** *and* ***plainly operable****.*
   The final system has been delivered as a self-contained executable `.jar` file, the initialisation of which simply requires command-line execution or opening through a graphical file manager. Any instructions provided through the interface are in plain, non-technical English, as opposed to technical jargon. 'Help buttons' are included within the interface to aid in the use of the system, should the analyst have any problems.
   We conclude, therefore, that non-functional requirement 5 has been entirely fulfilled by the system.

7.2 Evaluation of success

Comparing the final system to the initial functional requirements, we observe that, for reasons aforestated, two lower-priority requirements were not fully completed – those requirements deemed *essential,* however, have been fulfilled. Further, in enabling the analyst to display message rates over time for a *user-inputted time range* – as opposed to the entire time period spanned by the Enron corpus – the system has fulfilled one of the identified *stretch goals*. In identifying *potential aliases of individuals*, the system has partially satisfied a further lower-priority requirement deemed to be a *stretch goal*.

All three predominant, high-priority aims of the project have been satisfied in entirety; the implemented system is fully capable of *displaying message rates over time*, of *ranking employees* (on the principle of connectedness), and of *grouping employees* (*i.e.,* identifying *communities*) based on their communication patterns.

The satisfiability of all initially determined non-functional requirements has been demonstrated, through both user-acceptance testing (*see* **4.3**) and the feedback received from the client throughout the project; our feedback suggests the GUI, for instance, of being *informative*, *responsive* and *easy to use*. The design of the system remains consistent throughout all components subsumed by the system.

The project was completed by the deadline of the second deliverable; although there were some delays within the project, these setbacks were overcome with the good project management detailed in **5.3**. The Gantt chart and Risk Analysis, as presented within the *Design and Planning Document* (*see* Appendix **B**), enabled us to remain on schedule and circumvent any insurmountable failures with efficacy and efficiency.

By virtue of pragmatic project management, the system – satisfying the majority of the initial requirements and the entirety of the high-priority requirements – has been delivered in accordance with the prescribed project deadline. As we detail in **7.3**, it is plausible that functional improvements to the system may have been incorporated should greater time and resources have been provided.

7.3 Extensibility and future work

The project, and in part the resultant system, was in scope a *proof of concept*. Thus, there are several prospective features that, in future updates, could be incorporated to allow the system to satisfy all initially determined functional requirements; such a system, we suspect, may be suitable for commercial use.

As currently implemented, the system utilises a *locally stored database* constructed in `SQLite`. In order to extend this database – and thus enable the system to accept larger datasets, in addition to performing more comparatively efficient computations – a physical server (utilising a `SQL Server` database) may be purchased. This prospective efficiency of execution would be facilitated by the local storage of commonly calculated results, naturally entailed by a larger physical medium.

In our initial discussion of functional requirements – and in the subsequent, considerable algorithmic design pertaining thereto – we discussed the derivation of a *social hierarchy*. As a *stretch goal* – and, as such, occupying a *lower priority* – the implementation of such a function was precluded by oppresive delays in the implementation stage. Any future updates would be immediately concerned with the implementation of this function. Further, as only *message rates over time* can currently be analysed as such, the analysis of communities and employee rankings *over a specific period of time* may constitute a desirable and plausible future update.

In addition, it was discussed that employees for whom there exists associated (*potential*) aliases should have all such associated addresses combined into a single vertex of the generated graph, upon which EDA could then be conducted. As the current system fails to perform this combination[15], this function may constitute a prospective future improvement.

As was also discussed, an effort to enhance the *security of information* could be undertaken as a future improvement. Currently, there does not exist any specific implementation of security measures (*e.g.* a *hashing function*, as briefly discussed in **B.3.4.6**) facilitating the secure containment of the inputted dataset. In addition to hashing, a *login system,* ensuring the *restricted access* to the results of the conducted EDA, may also be implemented.

Finally, a future update would allow not only the Enron corpus to be accepted by the system as input, but also any other arbitrarily large dataset that adheres to the constraints of permitted formatting. Thus, the analyst could conduct exploratory analysis on personal, confidential datasets and identify communication patterns between whatever agents may occupy the position of vertices in the generated graph network.

[15] This specific function's status as a *stretch goal* of lower priority ultimately prevented its completion, due to the oppresive time constraints previously discussed.

References

[1] Brandes, U. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks 30*, 2 (2008), 136–145.

[2] Girvan, M., and Newman, M. E. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences 99*, 12 (2002), 7821–7826.

[3] Goldberg, M. K., et al. Extracting hidden groups and their structure from streaming interaction data. *arXiv preprint arXiv:1502.04154* (2015).

[4] Hardin, J., and Sarkis, G. Network analysis with the enron email corpus. *arXiv preprint arXiv:1410.2759* (2014).

[5] Keila, P. S., and Skillicorn, D. Detecting unusual and deceptive communication in email. In *Centers for Advanced Studies Conference* (2005), pp. 17–20.

[6] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM) 46*, 5 (1999), 604–632.

[7] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (1966), vol. 10, pp. 707–710.

[8] Mall, R., Langone, R., and Suykens, J. A. Multilevel hierarchical kernel spectral clustering for real-life large scale complex networks. *PLOS One 9*, 6 (2014).

[9] Rowe, R., Creamer, G., Hershkop, S., and Stolfo, S. J. Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web Mining and Social Network Analysis* (2007), ACM, pp. 109–117.

[10] Sims, B. H., Sinitsyn, N., and Eidenbenz, S. J. Hierarchical and matrix structures in a large organizational email network: Visualization and modeling approaches. In *Social Network Analysis-Community Detection and Evolution*. Springer, 2014, pp. 27–43.

[11] Wang, H., et al. Locality statistics for anomaly detection in time series of graphs. *Signal Processing, IEEE Transactions on 62*, 3 (2014), 703–717.

[12] Wilson, J. D., et al. A testing based extraction algorithm for identifying significant communities in networks. *The Annals of Applied Statistics 8*, 3 (2014), 1853–1891.

This page intentionally left blank

Requirements Analysis Report

Appendix A

Contents

Requirements Analysis Report

Identifying emergent behavioural patterns in the Enron corpus *via* exploratory data analysis

Zak Edwards, Karim Hedna, Robert Seddon-Greve,
Sandeep Singh and Thomas Thorpe

Preface

This document addresses the requirements of the pattern-detection system we intend to design and develop, the purpose, scope and high-level function of which we detail herein. The intended audiences for this document are the client, Deutsche Bank Group Technology & Operations (GTO), in addition to those responsible for its development. Our stakeholders are: the aforementioned client; Project Advisor Sarunkorn Chotvijit; and Professor Stephen Jarvis.

1 Introduction

1.1 Problem statement

The Enron corpus serves as a snapshot of internal communications during a period in which the Enron organisation actively engaged in fraudulent, manipulative activity. As a substantial collection of historic communications data, the corpus functions as an archetypal testing ground for the algorithmic detection of manipulative and otherwise anomalous trends in large datasets.

Idealistically, such an algorithm would allow the immediate, direct detection of anomalous activity without manual intervention. However, as such algorithmic detection is constantly circumvented by malicious traders, this proves, in practice, infeasible. Thus, one must instead remain vigilant for emergent behavioural patterns, on which one can conduct further manual analysis.

1.2 Purpose of the system

Fundamentally, we aim to develop software to facilitate the investigation of *patterns* and *structures* present within communications data. The large volume of messages precludes any effective manual analysis of the raw data; thus, the principal aim of the project is to provide analysts with an efficient and accessible means of detecting patterns algorithmically determined to be *of significant interest* within an inputted dataset – specifically, the Enron corpus – and reporting these findings to the analyst in a manner understandable to those of a non-technical inclination.

1.3 Scope of the system

In order to perform the most elementary of its functions, our system must accept large datasets, precisely as large as the dataset obtained from the Enron corpus subsequent to **data cleansing**. Our system must identify, and report to the user, the necessary behavioural patterns as determined in **2.1**; further, in order to satisfy the aforestated condition of reported findings being perspicuous to those of a non-technical persuasion, we aim to provide visualisations of the data corresponding to these findings where appropriate. We intend for our system to provide its functions at the explicit request of the analyst; thus, there shall exist a strong component of interactivity to ensure the functions *executed* are precisely those functions *requested* by the user.

2 Proposed System

2.1 Functional requirements

By **C**- and **D**-requirements we refer to *client* and *developer* requirements respectively. We assign one of the priority settings **H**, **M** or **L** to each primary requirement, representing *high*, *medium* and *low* priority respectively; it is assumed that each **D**-requirement is assigned the same priority as its parent **C**-requirement.

[**H**] 1. **C**-REQUIREMENT: The system shall display Enron employees' e-mail message rates over time – including the complete period spanned by the Enron corpus – and allow analysts to specify a *window of time* within the data for analysis of greater specificity. Message rates may be viewed for *any* and *all* employees, at the analyst's specification. The analyst may view the rate of *incoming* messages, *outgoing* messages, or a combination of the two.

   (a) **D**-REQUIREMENT: Assign clear attributes (`incoming`/`outgoing`) to e-mails in order to facilitate the seperability of the displaying of e-mail message rates (into *incoming, outgoing* or a combination thereof). A single, distinct e-mail may only be counted once as *outgoing,* but counted numerous times as *incoming* due to its possible characteristic of having multiple recipients. In this latter case, the e-mail shall have numerous `incoming` attributes.

   (b) **D**-REQUIREMENT: Develop a means of displaying e-mail message rates graphically – *i.e.*, with the possible assistance of existing developments facilitating the visualisation of large datasets, allow ascertained numerical quantities (corresponding to the number of e-mails of either homogeneous of heterogeneous attribution, in accordance with **2.2.**1. (a)) to be mapped to a graphical point at equidistant temporal intervals.

   (c) **D**-REQUIREMENT: Conduct data cleansing on the inputted dataset to eliminate anomalous and problematic e-mail message data.

[**H**] 2. **C**-REQUIREMENT: The system shall identify and display to the analyst *communities* (or, equivalently, *clusters*) amongst Enron employees – *i.e.,* groups characterised by communication patterns amongst members of those groups.

   (a) **D**-REQUIREMENT: Algorithmically generate a undirected graph from the inputted dataset, whose vertex set is composed of distinct Enron employees and whose edge set consists of e-mails from senders to recipients.

   (b) **D**-REQUIREMENT: Identify clusters by conducting **EDA** on *relations between vertices* in the generated graph network. A cluster shall be characterised as a subset of connected vertices exhibiting mutual graph-theoretical properties to be later determined.

   (c) **D**-REQUIREMENT: Develop a means of visually representing these communities graphically, by clearly highlighting the set of vertices constituting the community cluster within the graph network.

[**M**] 3. **C**-REQUIREMENT: The system shall derive and display an n-level social hierarchy, aimed to closely correspond with (*what is known about*) the historical corporate organisation of Enron employees.

   (a) **D**-REQUIREMENT: Assign a normalised, scaled *social factor* S ($0 \leq S \leq 100$) to each Enron employee (*i.e.,* to each distinct graph vertex) which in turn, *via* grouping based on S-similarity, assigns each employee to one of n hierarchical levels. Those employees assigned a higher value of S shall exist at a higher hierarchical level, and at a leftmost position respective to each level. The factor S shall be determined by considering a number of weighted graph-theoretical parameters for each vertex.

   (b) **D**-REQUIREMENT: Develop a means of visually representing this hierarchy.

[**H**] 4. **C**-REQUIREMENT: The system shall assign a ranking to each Enron employee, based on an intuitive notion of each employee's *importance* with respect to Enron's corporate organisation. The analyst shall be allowed access to a list of ranked employees.

   (a) **D**-REQUIREMENT: Determine a precise, graph-theoretical notion of *importance* which can justifiably be said to correlate to the corporate structure of Enron, and to a degree of which each and every Enron employee can be assigned. *Importance* as a quantitative property, by virtue of the way the social factor S is determined for each employee (as detailed in **2.2.**3. (a)), is precisely equivalent to the raw numerical value of S; 'important' vertices shall be minimally characterised by the degree of *centrality* and *connectedness* they exhibit.

(b) **D**-REQUIREMENT: Output a list of Enron employees in *descending* (by default) order of importance, where the order may be changed to *ascending* at the analyst's request. This shall be achieved in a manner logically equivalent to reading the generated hierarchy (see **2.2.**3.) in a *top-bottom* and *left-right* manner.

[**M**] 5. **C**-REQUIREMENT: The system shall identify (and make known to the analyst) instances where employees are using *aliases;* that is, instances where *one or more* distinct e-mail addresses can *reasonably be determined* to belong to an individual for whom there exists an already-associated address.

(a) **D**-REQUIREMENT: Conduct EDA to identify aliases; directories in the extracted corpus determined to belong to an individual for whom there exists an already-associated directory shall result in the e-mail address associated with the former directories being flagged as aliases.

(b) **D**-REQUIREMENT: Combine vertices in the generated graph network determined (see **2.2.**5. (a)) to represent the same individual.

2.2 Nonfunctional requirements

[**H**] 1. **C**-REQUIREMENT: The system shall be capable of accepting large datasets as *input* (that is, the dataset must be *processible* by the framework of the system), precisely as large as the Enron corpus, and remain responsive.

(a) **D**-REQUIREMENT: A *locally stored* relational database, wherein Enron employees' e-mails exist as entries with the necessary schematic attributes, shall be utilised to facilitate the fast access and retrieval of corpus data; the system will execute commands to the database to retrieve data where required.

[**M**] 2. **C**-REQUIREMENT: The system shall display strong, intuitive graphical user interface (GUI) design, exhibiting consistency and accessibility.

(a) **D**-REQUIREMENT: There shall exist *buttons*, *windows* and *forms* (with a predetermined range of acceptable form input) that serve as satisfying components for each of the functional requirements. The implementation of these features shall be contingent on the suitability of each feature in satisfying each respective function.

(b) **D**-REQUIREMENT: Interface design shall be aesthetically consistent; there shall be no aesthetic variation amongst system components beyond what is necessary for the successful operation and coherent presentation of those components.

(c) **D**-REQUIREMENT: Graphical visualisation techniques shall be employed for displaying *message rates* (**2.2.**1.), any identified *communities* (**2.2.**2.) and the *organisational hierarchy* (**2.2.**3.); an ordered list shall be plainly displayed as a visualisation of *employee rankings* (**2.2.**4.).

(d) **D**-REQUIREMENT: There will exist an accessible and minimal interface (that is, components shall make explicit no technical information beyond what is intuitive to and desired by the analyst), and all desired functions (satisfying the aforestated functional requirements) will be plainly simple to execute through the interface.

[**L**] 3. **C**-REQUIREMENT: The system shall keep the analyst informed of its status, wherever a nontrivial (*of estimated execution time* $\geq$ 5s) functional process is in execution.

(a) **D**-REQUIREMENT: Wherever a nontrivial process is in execution, the system shall make use of a *loading bar* displaying its current progress.

[**M**] 4. **C**-REQUIREMENT: There shall be no unhandled errors or exceptional events due to incorrect or unorthodox input from the analyst.

(a) **D**-REQUIREMENT: Wherever the system presents the opportunity for user-input (including all instances of interaction with the system interface), there shall exist a function handling all possible errors and exceptional events that could occur.

(b) **D**-REQUIREMENT: The analyst shall be notified wherever an attempt is made at incorrect or unorthodox input, *via* appropriate error messages; such messages shall be expressed in plain language, shall precisely indicate the problem and shall constructively suggest a solution.

This page intentionally left blank

Planning & Design Document

Appendix B

Contents

Nomenclature

G We define a **graph** G as a set (V, E) where $V = v_1, v_2, \ldots, v_n$ is the set of vertices, E is the set of edges, and $e_{ij} \in E$ is the edge adjacent to vertices v_i and v_j.

N We define the **neighbourhood** N_i of a vertex v_i as $N_i = \{v_j\} : \ e_{ij} \in E$.

$\sup(x)$ We define the **supremum** $\sup(x)$ of subset S of a partially ordered set T to be the least element in T that is greater than or equal to all elements of S, if such an element exists.

$\inf(x)$ We define the **infimum** $\sup(x)$ of subset S of a partially ordered set T to be the greatest element in T that is less than or equal to all elements of S, if such an element exists.

$|x|$ We define the **cardinality** $|x|$ of a set x to be the number of elements contained within the set x.

$\|x\|$ If a *norm* $p : X \to R$ – which assigns a strictly positive length or size to each $x \in X$ – is given on a vector space X, we denote by $\|x\|$ the **normal length** of a vector $x \in X$.

$\mathbf{A}^\intercal$ The **transpose**, $\mathbf{A}^\intercal$, of a matrix $\mathbf{A}$ is a reflection of $\mathbf{A}$ over its leading diagonal.

Planning & Design Document

Identifying emergent behavioural patterns in the Enron corpus *via* exploratory data analysis

Zak Edwards, Karim Hedna, Robert Seddon-Greve,
Sandeep Singh and Thomas Thorpe

Abstract

We present the design of the system for which we have determined various functional and non-functional requirements, together with a planning considerations and a preliminary schedule of how (and when) this design is to be implemented. The intended audiences for this document are the client, Deutsche Bank Group Technology & Operations (GTO), in addition to those responsible for its design and development.

1 Introduction

1.1 Purpose of this document

The purpose of this document is to provide the reader with a detailed understanding of the *planning* and *architectural design* of the system we review in **1.3**.

1.2 Scope of this document

Principally, we intend for this document to establish a *blueprint* for the concrete implementation of the software system; thus, we detail the sufficiently low-level architecture of each system component – and the interaction between these components – in order to establish a guideline for the ultimate language-specific implementation of these components. The scope of the *planning* in this document is limited to: a discussion of our *objectives and priorities*; any *assumptions*, *dependencies and constraints*; the identification of possible *project risks* and the management thereof; and a *preliminary schedule* detailing the division of labour for the remainder of the project.

1.3 Project overview

We aim to develop software to facilitate the investigation of *patterns* and *structures* present within communications data; thus, our project's principal aim is to provide analysts with an efficient and accessible means of detecting patterns within the Enron corpus. These findings shall be reported to the analyst in a manner understandable to those of a non-technical inclination.

2 Planning

2.1 Objectives and priorities

The management objective is to deliver the system – in a plainly operable self-contained form – in the accordance with the schedule we present in **2.4**; we intend also for our system to traceably satisfy the numerous functional and non-functional requirements set forth in our *Requirements Analysis* report. Our Project Manager shall ensure these objectives by respectively: checking that progress is made as planned; and monitoring the quality of the system at various stages.

A number of risk factors are taken into account, that may cause insufficient progress to be made respective to any further point in the project schedule; we identify and manage these in **2.3**. By virtue of our assignment of a *priority* to each (functional) requirement, those requirements determined to be of low priority may be dropped out if our time constraints become oppressive.

2.2 Assumptions

We assume that the system for which we present a design framework *(see* **3.3***)* need not be integrated *in any way* into any existing systems, developed by Deutsche Bank GTO or otherwise.

2.3 Risk management

There are multiple project risks which we have identified and analysed in **Figure 1**.

2.4 Preliminary schedule

We present a Gannt chart *(see* **Figure 2***)* showing the distribution of labour for the remainder of the project.

3 Design

3.1 Design considerations

3.1.1 Fault-tolerant design

We incorporate into our system design a number of *fault-tolerant* properties.

Application errors, for instance, will be handled by common fault detection services; *e.g.,* common `Java` exception handling, and error checking on task processing.

If the database (the function of which we detail in **3.3**) is deleted during runtime, the interface of the system shall restricts the analyst's ability to execute functions; the system shall effectively fail gracefully. Similarly, if the database is not present upon system initialisation, user interface elements shall be restricted.

3.1.2 Design conventions

The architectural design of the system employs elements the *Object Oriented Methodology*. We design, and intend to develop, the system on a modular *component* basis which enables the effective re-use of existing components, and facilitates the sharing of its components wherever necessary by other parts of the system.

3.1.3 Architectural design pattern

Concerning the application of established *design patterns*, our system employs the *Repository* pattern, diagrammatically represented in **Figure 4**.

We inted to isolate the *back-end* data store layer to facilitate unit testing. This data source is accessed from many locations; thus, we apply centrally managed, consistent access rules and logic across all *front-end* components.

3.2 Preparation of the Enron dataset

The aggressive data cleansing script provided to us by Deutsche Bank GTO has several flaws. Firstly, it does not differentiate between the type of recipient (`To`, `CC`, `BCC`); this distinction, however, is necessary for the successful implementation of our design components. Additionally, some of the e-mails contain *invalid data;* we give the example of e-mail addresses containing two '`@`' symbols, and e-mails which appear to have been sent on *1/1/1980* (the system's default). In our preparation of the dataset, this problematic data will need to be filtered out. Finally, the provided cleansing script still permits the existence of duplicate e-mails, as the `BCC` recipients will be hidden from all other recipients – which, due to the system's implementation of the cleansing script, will result in the same email appearing twice, once with the `BCC` recipients and once without.

Low	Medium	High	V High

		A	B	C	D	E
		Minimal	Minor	Moderate	Significant	Severe
A	Not likely	Low	Low	Low	Low	Medium
B	Low Likelihood	Low	Low	Medium	Medium	High
C	Likely	Low	Medium	Medium	High	V High
D	Highly Likely	Low	Medium	High	V High	V High
E	Near Certainty	Medium	High	V High	V High	V High

ID	Title	Risk Type	Severity	Likelihood	Total Risk	Avoidance	Minimisation	Contingency Plan
1	Team members leaving project	Project Risk	C	A	Low	Keep morale in team high and regularly communicate	Organise the team well so that each job has multiple workers on it	Distribute work load to other team members
2	Large number of requirements changing	Project and Product Risk	C	C	Medium	Keep in regular contact with DB and confirm the requirements they wish us to complete	Use some agile methods so that the system can be adapted quickly to any changes	Same as minimisation
3	Scope of project not realised (eg. No. of emails)	Product Risk	D	B	Medium	Regularly contact DB and check the estimated size of the email database.	Implement the system so that it is adaptable and can be adjusted to allow larger datasets.	As minimisation, but also check whether current architecture is suitable for new dataset
4	Communication breaks down in team	Project Risk	E	A	Medium	Provide multiple communication links between team members (email, WhatsApp, Facebook)	Same as avoidance	Call an emergency meeting to establish communication links and remind team members the importance of communication
5	Underestimate size of project	Product Risk	C	B	Medium	Keep in regular contact with DB and SJ to ensure our understanding of the project is correct	Use our Gantt chart to keep our project on track.	At any point during the project, use the team members that haven't got a scheduled work slot, to step in and aid in the piece of work that the team is behind on
6	Team member becomes ill	Project Risk	C	A	Low	N/A	Organise the team so everyone has access to the resources they need to work on the system	Either distribute work to other team members or have the ill team member work from home.
7	Product doesn't meet requirements	Product and Business Risk	E	B	High	Ask DB lots of questions regarding the specification and get regular feedback from DB during implementation	Have an adaptable system so that things can be revamped and changed easier if things don't meet the requirement	Use all team resources to get a correct list of requirements and rework the system so that the new requirements can be implemented on schedule.
8	Algorithms not appropriate for task at hand	Project and Product Risk	D	A	Low	Contact DB and SJ to confirm our algorithm is based mathematically on the concept being requested.	Have dedicated algorithm designers who will work on coming up with appropriate algorithms	Adapt the system so that any existing algorithms can be replaced, so that they're still compatible with our data format
9	Other groups producing higher quality software	Product and Business Risk	E	B	High	Analyse similar products to find their strengths and weaknesses, then improving our system to better the other products.	Create a system that meets all the requirements the client has asked us to do, to the best of our ability	Use our team resources to improve any parts of the system that other groups have got a better version of.

Figure 1: Identification and analysis of project risks

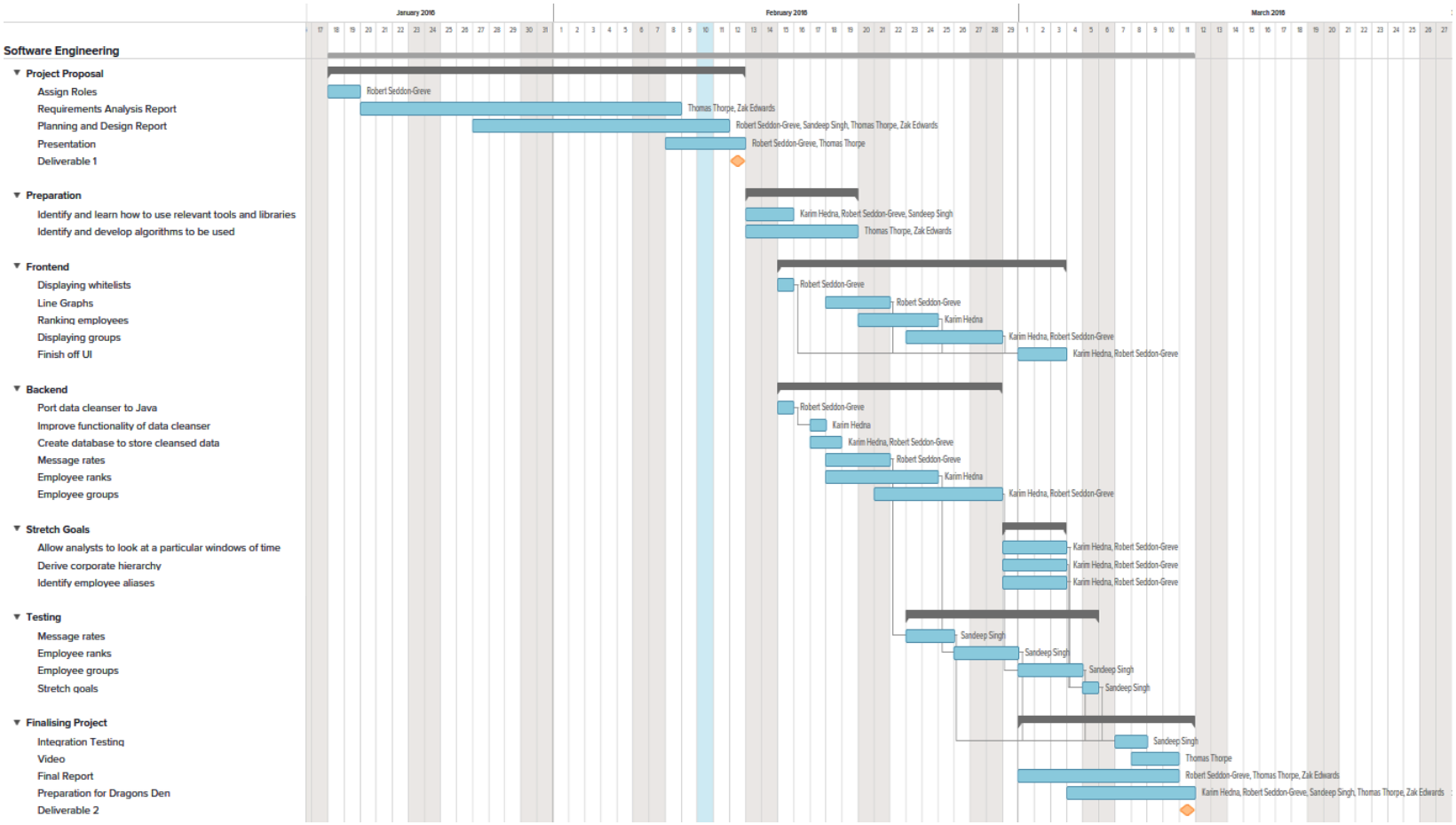


Figure 2: Gannt chart showing the distribution of labour for the remainder of the project

3.3 System architecture

Of the relational `SQL` database on which our system integrally depends, we present in **Figure 3** a schematic diagram showing relations between database entities; *primary keys* (**PK**) are clearly identified.

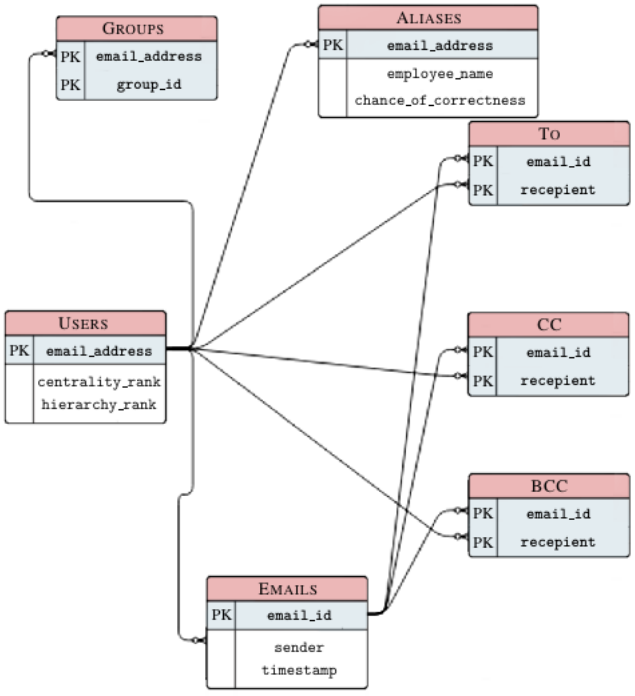


Figure 3: Schematic entity relations of the proposed database

We present in **Figure 4** a high-level view of the system, with components and functional relationships between the analyst and these components clearly identified.

3.3.1 Detecting aliases

Sequentially, we intend to develop an implementation of alias detection for the system before the other system components, allowing vertices in the generated graph network to be combined (Enron employees exist in the network as vertices, and thus aliases exist initially as *distinct* vertices) and ensuring the correctness of the remaining components' respective functions.

For each e-mail file in the Enron corpus (not yet in the database), we record the name of the employee to whom that email was delivered. Once all emails have been processed, the employee who has received the most e-mails sent by a particular sender is determined; our alias detection components functions under the assumption that *it is likely that* this particular sender's e-mail address belongs to the determined employee. By iterating this process for each address in the corpus, we can determine which employee owns each e-mail address, and thus determined whether there are employees that own multiple e-mail addresses.

3.3.2 Displaying message rates

Our system is intended to defaultly display message rates for both `incoming` and `outgoing` messages; thus, displaying message rates for the selected employee(s) involves counting the number of e-mails that those employees are *involved in* (*i.e.,* whose address exists in the fields `From`, `To`, `CC`, `BCC`) for each day, and representing this data graphically.

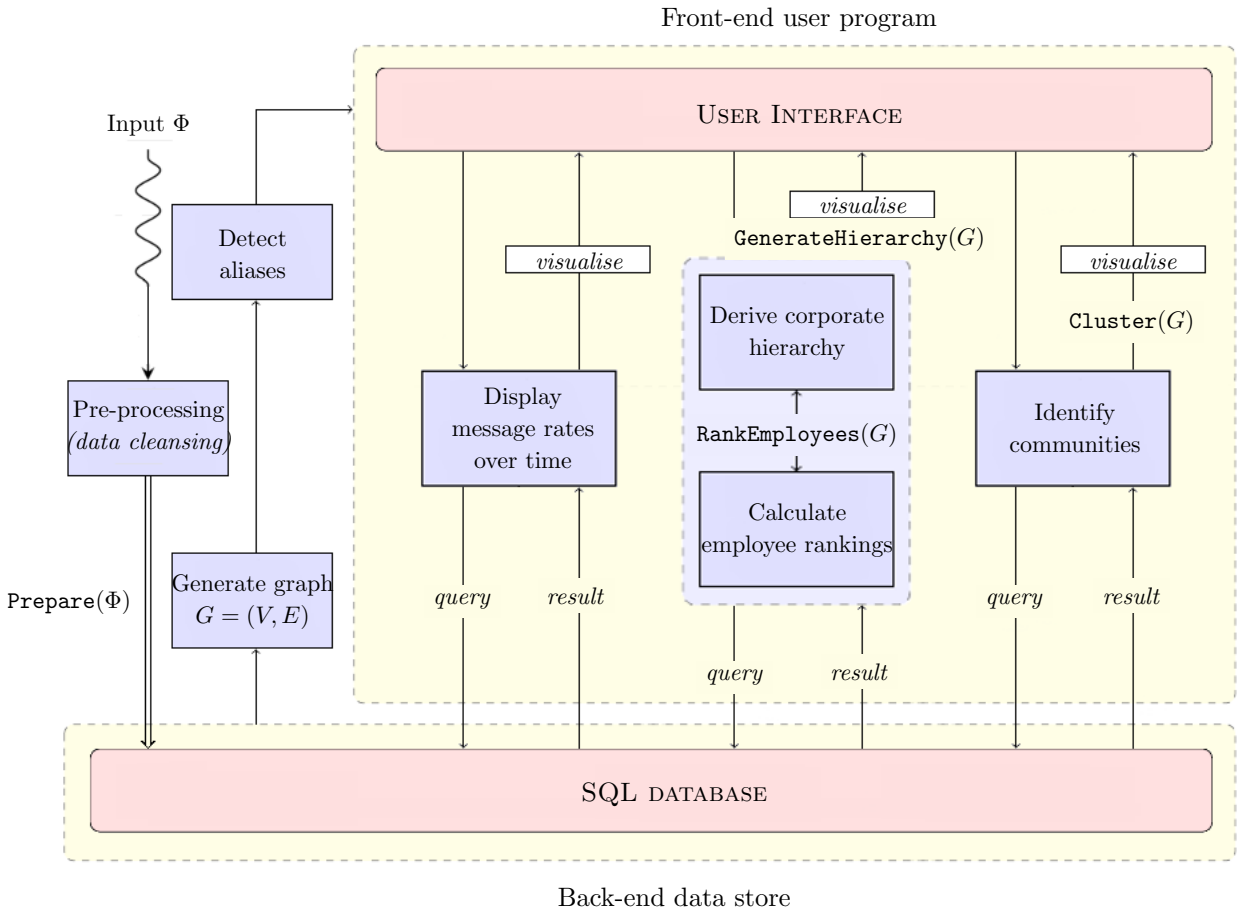


Figure 4: High-level architecture of the system

3.3.3 Community detection

Our system's community detection functionality utilises the **MH-KSC** algorithm (*see* **Algorithm 3**) introduced in Mall *et al.* [3], using the below detailed algorithms `GreedyMaxOrder`(S, t) and `GreedyFirstOrder`$(P_{test}, t^{(1)})$ to determine clustering information and ultimately propagate cluster memberships iteratively from the first to the coarest level of *cluster hierarchy*.

3.3.4 Generating a corporate hierarchy

In order to generate a *corporate hierarchy*, we intend to perform further analysis on the generated graph. We apply the *automated social hierarchy detection* procedure detailed in Rowe *et al.* [4]; fundamentally, we aim to determine a *social score* $\mathfrak{S}$ for each employee, and thus allocate each employee to a corresponding level (bounded by theoretical values of $\mathfrak{S}$) in the hierarchy. This normalised, weighted score is composed of a number of *factors*, which we proceed to detail.

Firstly, the *response score* factor is a combination of the quantity of an employee's distributed e-mails and their algorithmically determined *average response time*.

Definition (response score). *Let* $R(v_i)$ *denote the* response score *of employee (vertex)* $v_i \in V$. *Then*

$$R(v_i) \leftarrow \texttt{number\_of\_emails}(v_i) \wedge \texttt{avg\_response\_time}(v_i). \qquad \textbf{(Factor 1)}$$

Similarly, we define the *weighted clique score* to be composed of the *number of cliques* to which an employee belongs (straightforwardly algorithmically determined), and the *raw clique score* of that employee, which we define as $Q_R = 2^{n-1}$ for n employees in the clique set. We consider also the employee's average response time.

Algorithm 1: `GreedyMaxOrder`(S, t)

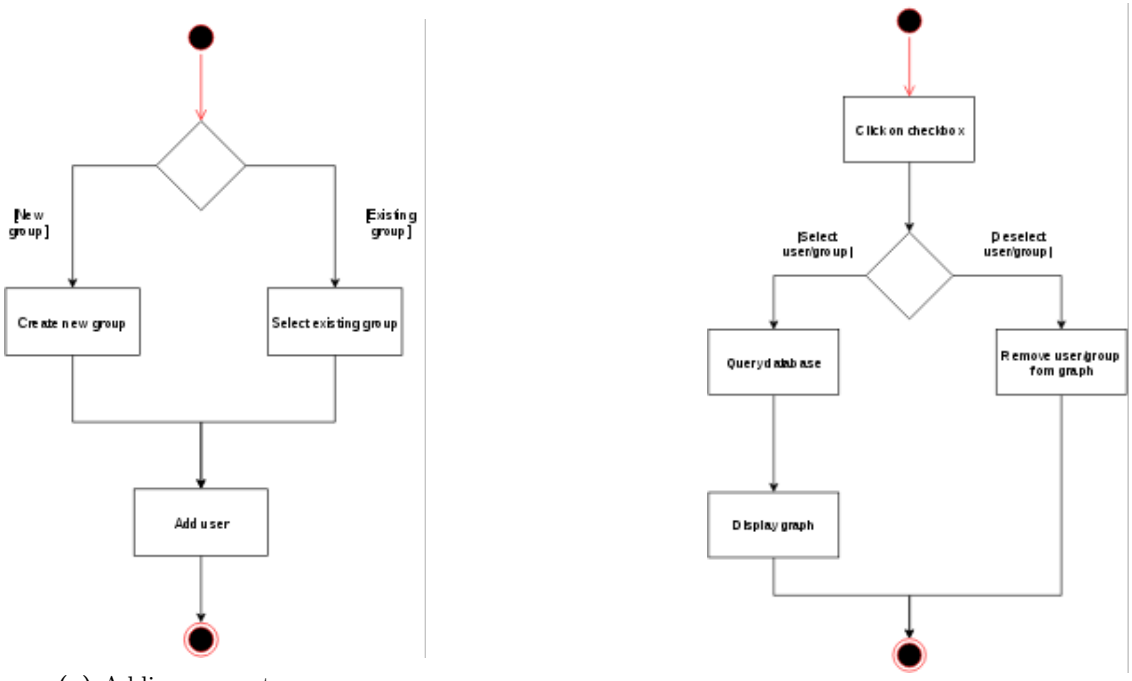
Input: Affinity matrix S and threshold t
variables $k \leftarrow 1$, $totinst \leftarrow 0$
while $totinst \neq |S|$ **do**
  Find i in range $[1, |S|]$ for which the number of instances j, such that $S(i,j) < t, j = 1, \ldots, |S|$, is a *maximum*
  Put indices of instance i and all instances j, such that $S(i,j) < t$, to C_k
  $k \leftarrow k + 1$, $totinst \leftarrow totinst + |C_k|$
  Set all elements corresponding to the indices in C_k to inf in S
  $C \leftarrow C \cup C_k$
end
$k \leftarrow k - 1$
Output: Clustering information C and the number of clusters k

Algorithm 2: `GreedyFirstOrder`$(P_{test}, t^{(1)})$

Input: projection matrix P_{test} and threshold $t^{(1)}$
variable $k \leftarrow 1$
while $|P_{test}| \neq 0$ **do**
  Select first node and locate all j for which `CosDist`$(e_i, e_j) < t^{(1)}$
  Put all these instances in $C_k^{(1)}$ and set to $C^{(1)}$
  $k \leftarrow k + 1$
  Remove these instances from P_{test}
end
$k \leftarrow k - 1$
for $i = 1$ *to* $|C^{(1)}|$ **do**
  for $j = i + 1$ *to* $|C^{(1)}|$ **do**
    Calculate $S_{test}^{(2)}(i,j)$ as the average `CosDist`$(\cdot,\cdot)$ between the eigen-projections of the instances in $C_i^{(1)}$ and $C_j^{(1)}$
  end
end
Output: Affinity matrix $S_{test}^{(2)}$, clustering information $C^{(1)}$ and k

Algorithm 3: `MH-KSC`(V, E)

Input: Undirected graph $G = (V, E)$, where $E \subseteq V \times V$, representing employees' communication patterns
Partition G into the *train*, *validation* and *test* sets V_{tr}, V_{va} and V_{te}
Construct $\mathcal{D} = \{x_i\}_{i=1}^{|V_{tr}|}$, $x_i \in \mathbb{R}^{|V|}$ from V_{tr}
Perform KSC on $\mathcal{D}$ to obtain the predictive model $\hat{e}^{(l)}(x) = \sum_{i=1}^{|V_{tr}|} \alpha_i^{(l)} K(x, x_i) + b_l$, $K : \mathbb{R}^{|V|} \times \mathbb{R}^{|V|} \rightarrow \mathbb{R}$
Obtain $P_{va} = [e_1, \ldots, e_{|V_{va}|}]^\intercal$ using predictive model and V_{va}
Construct $S_{va}^{(0)}(i,j) = \texttt{CosDist}(e_i, e_j) = 1 - \dfrac{e_i^\intercal e_j}{\|e_i\|\|e_j\|}$, $\forall e_i, e_j \in P_{va}$
Begin validation stage with $h = 0, t^{(0)} = 0.15$
$[C^{(0)}, k] =$ `GreedyMaxOrder`$(S_{va}^{(0)}, t^{(0)})$ `/* Algorithm 1 */`
$\mathcal{T} \leftarrow \mathcal{T} \cup t^{(0)}$, $\mathcal{C} \leftarrow \mathcal{C} \cup C^{(0)}$
while $k > 1$ **do**
  variable $h \leftarrow h + 1$
  Create $S_{va}^{(h)}(i,j) = \dfrac{\sum_{m \in C_i^{(h-1)}} \sum_{l \in C_j^{(h-1)}} S_{va}^{(h-1)}(m,l)}{|C_i^{(h-1)}| \times |C_j^{(h-1)}|}$
  Calculate $t^{(h)} = \text{mean}(\min_j(S_{va}^{(h)}(i,j)))$, $i \neq j$
  $[C^{(h)}, k] =$ `GreedyMaxOrder`$(S_{va}^{(h)}, t^{(h)})$
  $\mathcal{T} \leftarrow \mathcal{T} \cup t^{(h)}$, $\mathcal{C} \leftarrow \mathcal{C} \cup C^{(h)}$
end
Obtain P_{te} and begin with $h = 1, t^{(1)} \in \mathcal{T}$
$[S_{te}^{(2)}, C^{(1)}, k] =$ `GreedyFirstOrder`$(P_{te}, t^{(1)})$ `/* Algorithm 2 */`
$\mathcal{C} \leftarrow \mathcal{C} \cup C^{(1)}$
foreach $t^{(h)} \in \mathcal{T}, h > 1$ **do**
  $[C^{(h)}, k] =$ `GreedyMaxOrder`$(S_{te}^{(h)}, t^{(h)})$
  $\mathcal{C} \leftarrow \mathcal{C} \cup C^{(h)}$
  Create $S_{te}^{(h+1)}(i,j) = \dfrac{\sum_{m \in C_i^{(h)}} \sum_{l \in C_j^{(h)}} S_{te}^{(h)}(m,l)}{|C_i^{(h)}| \times |C_j^{(h)}|}$
end
Output: The set C for test set and propagate cluster memberships iteratively from 1^{st} to coarsest level of hierarchy



(a) Adding a user to a group

(b) Displaying the message rates of user(s)/group(s)

Figure 5: UML activity diagrams showing the system's responses to functional requests

Definition (weighted clique score). *Let* $t(v_i) \leftarrow \texttt{avg\_response\_time}(v_i)$. *Then the* weighted clique score

$$Q_W(v_i) = t(v_i) \cdot Q_R(v_i), \quad v_i \in V. \tag{\textbf{Factor 2}}$$

Definition (degree centrality). *Let* $\deg(v_i)$ *be the degree of vertex* $v_i \in V$*, and let* $\mathbf{A}$ *be the adjacency matrix of* G*. Then, we define*

$$\deg(v_i) = \sum_{j \in V} a_{ij} : \quad a_{ij} \in \mathbf{A}. \tag{\textbf{Factor 3}}$$

Definition (clustering coefficient). *We define the* clustering coefficient $C(v_i)$ *as*

$$C(v_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{2|\{e_{ij}\}|}{\deg(v_i) \cdot \deg(v_i - 1)} : \quad v_j \in N_i,\ e_{ij} \in E \tag{\textbf{Factor 4}}$$

for the neighbourhood N_i*.*

Definition (geodesic mean). *We define the matrix* $\mathbf{D}$ *of all shortest paths between all* $v \in V$*. Then, we define the* geodesic mean *respective to a vertex* $v_i \in V$ *as*

$$\text{geo}(v_i) = \frac{1}{n} \sum_{j \in V} d_{ij} : \quad d_{ij} \in \mathbf{D},\ n = |V|. \tag{\textbf{Factor 5}}$$

Definition (betweenness centrality). *Denote by* $\sigma(k, j)$ *the number of* (k, j)*-geodesics (shortest paths from* k *to* j*): i.e.,* $\sigma(k, j) = |\{d_{kj} : \quad d_{kj} \in \mathbf{D}\}|$*. Let* $\sigma(k, j|i)$ *be the number of* (k, j)*-geodesics passing through some vertex* $i \in V$*. If* $k = j$*, let* $\sigma(k, j) = 1$*; if* $i \in \{k, j\}$*, let* $\sigma(k, j|i) = 0$*. Then, the* betweenness centrality $B(v_i)$ *of a vertex* $v_i \in V$ *is defined as* [1]

$$B(v_i) = \sum_{i,j \in V} \frac{\sigma(k, j|i)}{\sigma(k, j)}. \tag{\textbf{Factor 6}}$$

We introduce a formulation of the **HITS** (*'Hubs-and-authorities'; cf.* Kleinberg [2]) algorithm in order to define the next factor. We represent the set of weights $\{x^{\langle v\rangle} : v \in V\}$ as a vector x with a coordinate for each vertex of G, and analogously the set of weights $\{y^{\langle v\rangle}\}$ as a vector y.

Algorithm 4: `Iterate`(G, k) (*HITS* iteration)

Input: graph $G = (V, E)$ and constant $k \in \mathbb{N}$
initial vector $z \leftarrow (1, 1, \dots 1) \in \mathbb{R}^{|V|}$
vector $x_0 \leftarrow z$, vector $y_0 \leftarrow z$
for $i \leftarrow 1$ **to** k **do**
  $x_{i-1}^{\langle v\rangle} \leftarrow \sum_{w:\ (w,v)\in E} y_{i-1}^{\langle w\rangle}$, denote by $x_i^{\langle v\rangle\prime}$ the new x-weights
  $y_{i-1}^{\langle v\rangle} \leftarrow \sum_{w:\ (v,w)\in E} x'_i$, denote by $y_i^{\langle v\rangle\prime}$ the new y-weights
  normalise $x_i^{\langle v\rangle\prime}$ such that $\sum_{v\in V} (x_i^{\langle v\rangle\prime})^2 = 1$, obtaining $x_i^{\langle v\rangle}$
  normalise $y_i^{\langle v\rangle\prime}$ such that $\sum_{v\in V} (y_i^{\langle v\rangle\prime})^2 = 1$, obtaining $y_i^{\langle v\rangle}$
end
Output: vector (x_k, y_k)

Algorithm 5: `Filter`(G, k, n) (*HITS* filtering)

Input: graph $G = (V, E)$ and constants $k, n \in \mathbb{N}$
$(x_k, y_k) \leftarrow$`Iterate`(G, k)
Result: vertices with the c largest coordinates in x_k, i.e., *authorities*, and vertices with the c largest coordinates in y_k, i.e., *hubs*

Definition (HITS-importance). *Denote by* `value(`v_i`)[`φ`]` *the value of* v_i *as determined by the function* ϕ*. Then, we define the* HITS-importance $H(v_i)$ *of a vertex* $v_i \in V$ *as*

$$H(v_i) = \texttt{value[Filter}(G, k, n)\texttt{]}. \qquad \textbf{(Factor 7)}$$

To compute $\mathfrak{S}$, we must first scale and normalize each of the previous statistics which we have gathered. The contribution Δ_x of each factor x is individually mapped to a $[0, 100]$ scale and weighted in accordance with the formula $w_x \cdot \Delta_x$, where w_x is the respective weight for that factor. The values $\sup(x)$ and $\inf(x)$ are computed across all i vertices and x_i is the value for the ith vertex; this normalisation is applied to the above **Factors** (**1**)-(**7**).

$$w_x \cdot \Delta_x = w_x \cdot 100 \cdot \left(\frac{x_i - \inf(x)}{\sup(x) - \inf(x)}\right), \qquad \mathfrak{S} = \frac{\sum_{\text{all } x}(w_x \cdot \Delta_x)}{\sum_{\text{all } x} w_x}.$$

We present a simple pseudocode for the function `GenerateHierarchy`(G, n).

Algorithm 6: `GenerateHierarchy`(G, n)

Input: graph $G = (V, E)$
for $i \leftarrow 1$ **to** $|V|$ **do**
  Calculate $\mathfrak{S}(v_i)$
  Assign v_i to a hierarchical level n *(Grouping algorithm)*
end
Output: n-level hierarchy

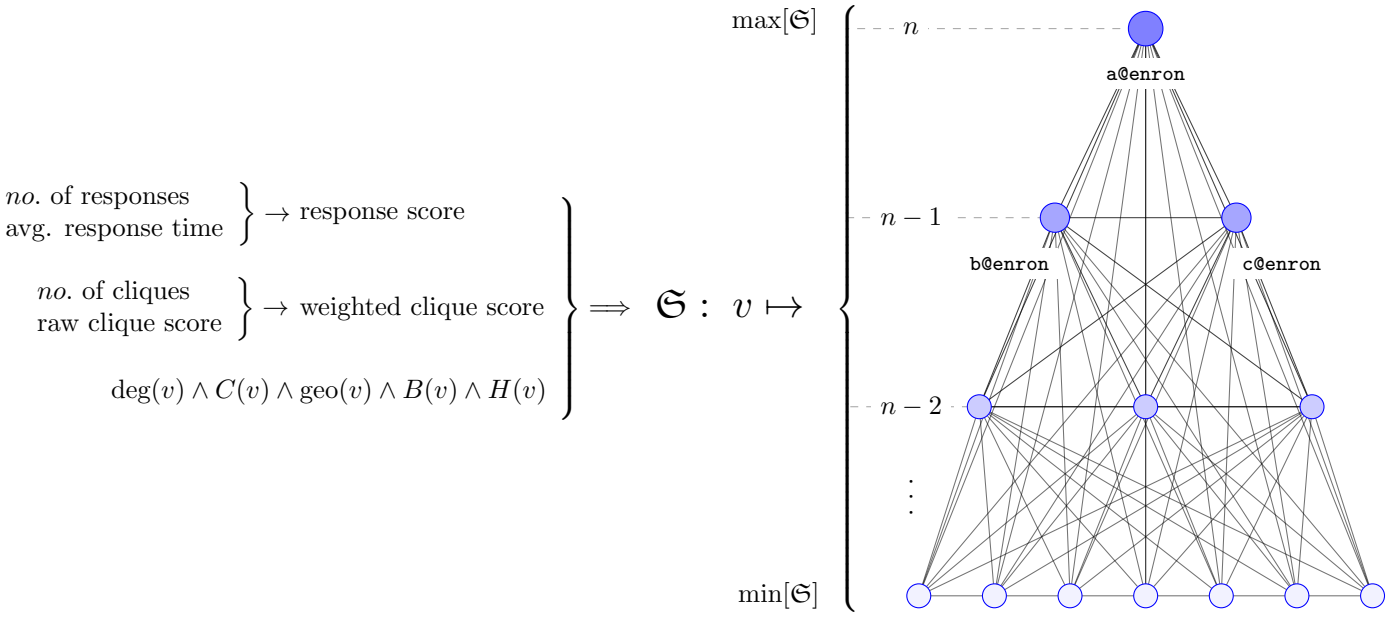


Figure 6: Parametric composition of the *social score* $\mathfrak{S}$

3.3.5 Ranking Enron employees

The *employee ranking* functionality is entailed quite trivially from the development of previous component; we simple rank employees in descending order according to the raw numerical value of $\mathfrak{S}$ corresponding to that employee. We present a simple pseudocode for the function `RankEmployees`(G).

Algorithm 7: `RankEmployees`(G)

Input: graph $G = (V, E)$
for $i \leftarrow 1$ **to** $|V|$ **do**
  Calculate $\mathfrak{S}(v_i)$
end
Output: $\mathfrak{S}(v)$ from `max[`$\mathfrak{S}(v)$`]` to `min[`$\mathfrak{S}(v)$`]`

3.4 System attributes

3.4.1 Extensibility

A number of functional extensions could potentially be introduced to facilitate the further growth of the system. With concern to the existing theoretical constraints on the volume of – and the analyst's interaction with – the inputted dataset, a dedicated database server could be established, thus enabling the system to accept a larger dataset and permitting relational database queries to be performed with greater efficiency. Such an extension to the system would require only trivial modifications to the existing architecture; for instance, a database package suited towards the handling of large data may be plainforwardly utilised.

Currently, no data mining functionality shall be incorporated within the system. In principle, one may consolidate such functionality by extending the existing *community detection* features (*see* **3.3.3**) and performing analysis (including, but not necessarily restricted to, lexicographical analysis) on those messages exchanged between members of the identified communities. Incorporating data mining functionality would thus only require the implementation of an appropriate algorithm(s), accepting the contents of the e-mail body as input, and the storing of the necessary data in the established database.

3.4.2 Reliability

Without the occurence of an external operating system (OS) error, the system shall never *crash* or *hang* – that is, without the appropriate UI elements indicating a process of non-trivial execution time is in progress. In the event of fortuitous

system delays and fatal failures due to OS-errors, or whilst performing unanticipated and sufficiently complex operations on the generated graph, the system shall attempt to maintain graceful degradation.

The *locality* of the relational database, to which the interface issues queries and from which queried data is retrieved, effectively renders the existence of an auxiliary database redundant, as the locally-stored database is at no risk of being subjected to network failure.

3.4.3 Correctness

The system shall only be considered *correct* when all specified requirements, functional and non-functional, have been satisfied. Verification of the system's functional aspects – the execution of each function and the algorithmic process thereof – shall be achieved through modular and monolithic testing, and static and dynamic program analysis.

Further, the system must have also passed any acceptance tests determined by the client, in order to ensure the client has signed off on the completion of the project.

3.4.4 Compatibility and portability

The system shall be distributed as a self-contained `.jar` (`Java` archive) file, which non-technical users shall find easily executable. Due to the nature of the system's distribution, the host machine shall only require the installation of Java SE Runtime Environment (JRE) 8; thus, the system shall be portable to any operating system on which the JRE8 virtual machine is fully supported.[1]

No other specific portability requirements have been identified.

3.4.5 Modularity and re-use

All primary functional components of the system (*see* **Figure 4**: these components are represented diagrammatically as modules within the system's front-end layer) will be developed independently – *viz.,* the algorithmic framework of these components are functionally discrete – thus facilitating the straightforward adding, removal and further manipulation of these components. Additionally, we shall make use of well-developed and thoroughly tested external libraries, such as e-mail readers and `JSON` (JavaScript Object Notation) readers, in order to minimise the amount of code that is unnecessarily reproduced.

The system shall be further modularised by adopting object-oriented design methodologies *(see* **3.1.2***), viz.* employing independent classes and interfaces whose functions are as *general* or *specific* as is determined to be appropriate.

3.4.6 Security

As the system does not ostensibly present the analyst with the opportunity to issue non-predetermined queries to the database, only rudimentary security measures – which shall solely concern interaction with the database – shall be implemented. As we expect and require the database to remain static subsequent to preprocessing (detailed in **3.2**), a simple hash function shall be employed to allow the *state* of the data therein to be represented as an arbitrary, invariable integer. This numerical value shall only be affected by variation if any successful attempt is made at manipulating (issuing `update`, `delete`, *etc.*) any database entry; thus, if any variation is detected, the system shall regenerate the database.

```
                    hash function
SELECT checksum_agg(binary_checksum(*)) FROM User;
/* Result: -34529210 */
UPDATE User SET (hierarchy_rank = -1) WHERE (email_adress = name@enron.com);
SELECT checksum_agg(binary_checksum(*)) FROM User;
/* Result: -93551732 */
```

[1] An exhaustive list of such systems is provided by Oracle: `http://www.oracle.com/technetwork/java/javase/certconfig-2095354.html`

References

[1] Brandes, U. On variants of shortest-path betweenness centrality and their generic computation. *Social Networks 30*, 2 (2008), 136–145.

[2] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM) 46*, 5 (1999), 604–632.

[3] Mall, R., Langone, R., and Suykens, J. A. Multilevel hierarchical kernel spectral clustering for real-life large scale complex networks. *PLOS One 9*, 6 (2014).

[4] Rowe, R., Creamer, G., Hershkop, S., and Stolfo, S. J. Automated social hierarchy detection through email network analysis. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web Mining and Social Network Analysis* (2007), ACM, pp. 109–117.

This page intentionally left blank

Appendix C Test Cases Document

C.1 Initialisation

	Test description	Expected result	Actual result	Pass/Fail
1	Does the select a whitelist option allow you to choose a `.json` file?	The system should accept the `.json` file.	The system accepted a `.json` file containing the whitelist.	Pass
2	Does the select a whitelist option allow you to choose a file that is not of `.json` format?	The system should reject the analyst from selecting a file which is of a non-`.json` format.	The system does not display any non-`.json` files in the window preventing the analyst from choosing the wrong file type.	Pass
3	If you cancel choosing a whitelist, does the system still function correctly?	The system should maintain functionality and let the analyst know they still need to choose a whitelist.	The system crashed and needed to be reloaded to continue use.	Fail
4	Does the select a mail directory option allow you to choose a folder containing the e-mail directories?	The system should accept the folder being chosen.	The system accepted the mail directory containing the e-mails.	Pass
5	Does the select a mail directory option allow you to choose anything other than a folder as the mail directory?	The system should reject the analyst from selecting something which is not a folder.	The system does not allow the analyst to choose anything other than a folder as the mail directory.	Pass
6	If you cancel choosing a mail directory, does the system still function correctly?	The system should maintain functionality and let the analyst know they still need to choose a mail directory.	The system crashed and needed to be reloaded to continue use.	Fail
7	Does the system correctly detect a database file from a previous use of the system?	The system should notify the analyst that it has detected a database already and allow them to continue with system execution, without having to select another mail directory first.	The system correctly detected a database from a previous running of the system and alerted the analyst it had done so.	Pass
8	Does the system allow the analyst to choose another mail directory if it has already detected a database from a previous use?	The system should allow the analyst to select another mail directory by first moving the slider, telling the system they wish to overwrite the current database, and proceed by choosing another.	The system allows the analyst to specify they wish to overwrite the current database and insert a new mail directory into the system.	Pass
9	Does the system let the analyst continue with the system once a whitelist, and mail directory if creating a new database, has been chosen and the tick icon has been clicked?	The system should continue with its execution and load the main interface once it has calculated the groups and ranking.	The system correctly proceeds with execution as expected.	Pass
10	Does the system let the analyst click the tick to continue with the system execution if it has not chosen a whitelist, mail directory (if creating a new database) or both?	The system should prevent the analyst from continuing with system execution until they have specified the whitelist and mail directory (if replacing the current database if it exists).	The system prevents the analyst from proceeding with execution until they have specified what they want for all options.	Pass

C.2 Detecting aliases

11	Can you click on the aliases button?	The system should allow the analyst to click on the aliases button.	The aliases button can be clicked.	Pass

12	Does clicking on the aliases button cause a window to popup which contains a list of suspected aliases?	The system should load another window which contains a list of aliases labelled by number.	A pop up window appears when the aliases button is clicked on. The window contains tabs for any potential alias identified.	Pass
13	Can you click on each of the smaller tabs in the aliases window?	The system should allow you to click the smaller tabs in the aliases window which displays all the e-mails believed to be associated with one person.	Each of the smaller tabs can be clicked on to expand and collapse. When expanded the tab contains a list of e-mails believed to be suspected aliases belonging to one person.	Pass

C.3 Displaying message rates

14	Can you select and deselect the incoming e-mails checkbox?	Checkbox should function correctly.	Checkbox functions as expected.	Pass
15	Can you select and deselect the outgoing e-mails checkbox?	Checkbox should function correctly.	Checkbox functions as expected.	Pass
16	Can you select and deselect the combined e-mails checkbox?	Checkbox should function correctly.	Checkbox functions as expected.	Pass
17	Can you select and deselect the add to group checkbox?	Checkbox should function correctly.	Checkbox functions as expected.	Pass
18	Can you select an employee from the combo box?	Combo box should function correctly by allowing the analyst to select an employee from the combo box.	Combo box functions as expected.	Pass
19	If you select the incoming e-mails checkbox for the selected employee, does it display a line on the graph which correctly reflects their incoming communications?	A line should appear on the graph which correctly matches the specified employee's incoming communications.	A line was displayed on the graph which matched the employee's incoming communications patterns.	Pass
20	If you select the outgoing e-mails checkbox for the selected employee, does it display a line on the graph which correctly reflects their outgoing communications?	A line should appear on the graph which correctly matches the specified employee's outgoing communications.	A line was displayed on the graph which matched the employee's outgoing communications patterns.	Pass
21	If you select the combined e-mails checkbox for the selected employee, does it display a line on the graph which correctly reflects the sum of their incoming and outgoing communications?	A line should appear on the graph which correctly matches the sum of the specified employee's incoming and outgoing communications.	A line was displayed on the graph which matched the sum of the employee's incoming and outgoing communications.	Pass
22	If you deselect the incoming e-mails checkbox for the selected employee, does it remove the relevant line from the graph?	The incoming e-mails line for the selected employee should be removed from the graph.	The correct line was removed from the graph.	Pass
23	If you deselect the outgoing e-mails checkbox for the selected employee, does it remove the relevant line from the graph?	The outgoing e-mails line for the selected employee should be removed from the graph.	The correct line was removed from the graph.	Pass
24	If you deselect the combined e-mails checkbox for the selected employee, does it remove the relevant line from the graph?	The combined e-mails line for the selected employee should be removed from the graph.	The correct line was removed from the graph.	Pass
25	If you select the add to group button for the selected employee, does it add them to the selected group?	The employee should be added to the selected group and their e-mail appears in the list of employee in the group.	The employee was added to the group successfully.	Pass

26	If you deselect the add to group button for the selected employee, does it remove them from the selected group?	The employee should be removed from the selected group and their e-mail address is removed from the list of employee in the group.	The employee was removed from the group successfully.	`Pass`
27	If you click the add group button does it create a new group which is added to the list of groups?	A new group should be created and added to the list of groups.	A new group was created and added to the list of groups.	`Pass`[16]
28	If you select the checkbox next to a group, does it display a line on the graph which correctly reflects the sum of incoming and outgoing e-mails of all members of the group?	A line should be drawn on the graph which correctly shows the sum of incoming and outgoing e-mails of all members in the group.	A line was drawn on the graph which matched the sums of incoming and outgoing communications of each employee in the group on the graph.	`Pass`
29	If you deselect the checkbox next to a group, does it remove the relevant line from the graph?	The line showing the message rates over time for the chosen group should be removed from the graph.	The correct line was removed from the graph.	`Pass`
30	Does quickly selecting and deselecting checkboxes on the interface break the functionality of the system?	The system should still be able to function correctly regardless of the speed of user input.	Clicking checkboxes quickly caused system functionality to break and lines not being generated on the graph as expected.	`Fail`[17]
31	Does choosing a begin and end date to display data between (where the begin date is before the end date), correctly display communications between the given time period?	Any lines which are already on the graph, or any which are added in the future, should only display date between the given range.	The graph's x-axis was modified and the lines re-drawn correctly to only show data sent between the specified time period.	`Pass`
32	Does choosing a begin and end date to display data between (where the end date is before the begin date) alert the analyst they cannot do so?	The analyst should not be prevented from doing this.	The analyst was allowed to do this and caused the system functionality to be broken.	`Fail`[18]
33	Does the analyst entering random alphanumeric text in the begin and end date picker cause the system to stop functioning correctly?	The user input should be ignored and the previous, accepted, date should be maintained.	The system rejected the alphanumeric input and kept the dates as the original dates before trying to change the input.	`Pass`
34	Does clicking the reset date ranges button cause the system to reset the date ranges correctly?	The begin and end date should be set to the extremes (earliest start date and latest end date) of the current data displayed on the graph such that all the data for selected users and groups is visible.	The date ranges were reset correctly to be the far extremes as expected. Verified by increasing the range by 1 day on each side to see no communications before or after.	`Pass`
35	Does clicking the reset chart button cause the system to remove all lines on the graph?	The system should remove all lines on the graph and uncheck any checkboxes on the interface which should be affected by this such as the groups being displayed.	The system correctly removed all lines from the graph and deselected any checkboxes visible on the interface which had to be.	`Pass`
36	Does adding a group member to a group already being displayed on the graph cause the graph to be re-drawn to reflect the change?	The graph should be re-drawn with the new member's combined e-mail message rates being added onto the old line's values.	The system correctly redrew the graph adding on the values of the new group member to the old line.	`Fail`
37	Does removing a group member from a group already being displayed on the graph cause the graph to be re-drawn to reflect the change?	The graph should be re-drawn with the old member's combined e-mail message rates being removed from the old line's values.	The system correctly redrew the graph removing the values of the old group member from the old line.	`Fail`

[16] However, when running this test case, a bug occurred where any displayed group had their checkboxes unchecked while the line for that group remained on the graph.

[17] Consider adding delays after each user input to allow the system time to display the graph.

[18] Implement correct user input checking to prevent this from happening.

38	When switching between employees using the combo box, are the states of each employee correctly conserved? *i.e.,* if employee 1 has incoming e-mails checked, when switching to employee 2 and back does employee 1's incoming e-mail checkbox remain checked?	The system should correctly conserve the state of each employee's data being displayed on the graph so when they are chosen again through the combo box, they have settings which match what is displayed on the graph and groups list.	The system correctly conserves states as described in the test description field.	`Pass`
39	Does switching between options on timescale correctly affect the graph?	When an option is chosen (*e.g.* day, week, month, year), the graph displays the data with each point representing the chosen time unit.	Functions as expected.	`Pass`

C.4 Community detection

40	Is it easy to see the graph?	The graph should be clear and easy to see.	The graph is easy to see on the screen.	`Pass`
41	Does the zoom functionality work correctly?	The graph should zoom in and out as the analyst desires.	The zoom functionality works as intended.	`Pass`
42	Can different nodes be distinguished from each other?	Nodes should be easy to differentiate.	The nodes are easy to distinguish between.	`Pass`
43	Can different groups be distinguished from each other?	Each group should be a different colour making it clear to the analyst the different groups.	Each of the groups have a different colour and have no edges between groups, making it easy to distinguish the different groups on the graph.	`Pass`
44	Are employees on the whitelist clickable?	The analyst should be able to click on any e-mail on the whitelist.	Employees on the whitelist can be clicked.	`Pass`
45	When an e-mail address on the whitelist is selected, are the e-mails of everyone in that group displayed in the box below?	A list of e-mails, including the e-mail of the selected employee, should be displayed in the box below the whitelist. All e-mails should belong to employees in the same group.	When clicking on an employee, a list of e-mails is displayed in the box below of all group members.	`Pass`
46	Do the groups displayed in the textbox, when clicking on an employee, match those generated by the graph.	The employees in the group displayed in the checkbox should have a grouping on the graph where all nodes are present in the list.	The list of e-mails in the box match those in the group displayed by the graph.	`Pass`

C.5 Generating a corporate hierarchy

Due to the feature not being implemented, tests for the hierarchy have been removed.

C.6 Ranking Enron employees

47	Can you select employees in the combo box?	The system should allow you to select employees from the combo box.	Employees in the combo box are selectable.	`Pass`
48	When you select an employee from the combo box does it display their employee position and raw rank score?	The system should display the employee position and raw rank score of the employee selected below the combo box.	The employee's position and raw rank is displayed below the combo box as expected.	`Pass`

49	When an employee is selected from the combo box, is the graph re-drawn to show their node in the graph?	The graph should be re-drawn and the employee selected should have their node and any edges connected to it highlighted green.	The employee selected is highlighted in the graph on the right.	`Pass`
50	Are the list of employees in the combo box ordered in descending order of position?	The combo box should list the employees in descending order of position such that the employee with rank 1 should be the first option.	The employees are ordered in descending order of ranking such that employee position 1 is the first option of the combo box.	`Pass`

C.7 Data cleansing and data insertion

51	Does the database contain any e-mails which have no recipient?	The database should return null when querying for e-mails where recipient is blank as it is a primary key.	The database contained no e-mails with no recipient.	`Pass`
52	Does the database contain any e-mails where the timestamp is in `1980` or after `2040`?	The database should not contain any e-mails where the timestamp of the e-mail is before `1980` or `2040` as this indicates the timestamp in the original dataset was damaged or not available.	There are no e-mails in the database that have a timestamp of either `1980` or after `2040`.	`Pass`
53	Does the database contain any duplicate e-mails?	The database should contain no e-mail duplicates as all message contents were hashed using `MD5` and duplicate e-mails were prevented from being added by the `SQL` database.	There are no duplicate e-mails in the database.	`Pass`

This page intentionally left blank

Appendix D (Semi-)Complete Program Listings

Listing C.1.1: AliasGroupObject

```
public class AliasGroupObject {

  private String emailAddress;
  private String shortEmailAddress;
  private int groupNumber;

  public AliasGroupObject(String emailAddress, int groupNumber) {
    this.emailAddress = emailAddress;
    this.groupNumber = groupNumber;
    this.shortEmailAddress = emailAddress.split("@", 2)[0];
  }

  public String getShortEmailAddress() {
    return shortEmailAddress;
  }

  public void setShortEmailAddress(String shortEmailAddress) {
    this.shortEmailAddress = shortEmailAddress;
  }

  public String getEmailAddress() {
    return emailAddress;
  }

  public void setEmailAddress(String emailAddress) {
    this.emailAddress = emailAddress;
  }

  public int getGroupNumber() {
    return groupNumber;
  }

  public void setGroupNumber(int groupNumber) {
    this.groupNumber = groupNumber;
  }

  @Override
  public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + ((emailAddress == null) ? 0 : emailAddress.hashCode());
    result = prime * result + groupNumber;
    return result;
  }

  @Override
  public boolean equals(Object obj) {
    if (this == obj)
      return true;
    if (obj == null)
      return false;
    if (getClass() != obj.getClass())
      return false;
    AliasGroupObject other = (AliasGroupObject) obj;
    if (emailAddress == null) {
      if (other.emailAddress != null)
        return false;
    } else if (!emailAddress.equals(other.emailAddress))
      return false;
    if (groupNumber != other.groupNumber)
      return false;
    return true;
  }
```

```
  @Override
  public String toString() {
    return emailAddress;
  }

}
```

Listing C.1.2: AliasIdentifier

```
public class AliasIdentifier {

  DatabaseConnector conn;
  ArrayList<String> whitelistList;
  ArrayList<Set<AliasGroupObject>> listOfAliasSets;

  public AliasIdentifier(DatabaseConnector conn, ArrayList<String> whitelistList) {
    this.conn = conn;
    this.whitelistList = whitelistList;
    listOfAliasSets = new ArrayList<Set<AliasGroupObject>>();
  }

  public void identifyAliases() {
    System.out.println("Starting to identify aliases");
    long start = System.nanoTime();
    NormalizedLevenshtein normalizedLevenshtein = new NormalizedLevenshtein();
    ArrayList<AliasGroupObject> emailAddresses = new ArrayList<AliasGroupObject>();
    for (String email : whitelistList) {
      emailAddresses.add(new AliasGroupObject(email, 0));
    }
    listOfAliasSets = new ArrayList<Set<AliasGroupObject>>();
    int numGroups = 0;
    for (int i = 0; i < emailAddresses.size(); i++) {
      for (int j = i+1; j < emailAddresses.size(); j++) {
        if (!(emailAddresses.get(i).getGroupNumber() != 0 && emailAddresses.get(j).getGroupNumber() != 0)) {
          double distance = normalizedLevenshtein.distance(emailAddresses.get(i).getShortEmailAddress(),
              emailAddresses.get(j).getShortEmailAddress());
          if ((1-distance) > 0.75) {
            AliasGroupObject iObject = emailAddresses.get(i);
            AliasGroupObject jObject = emailAddresses.get(j);
            if (iObject.getGroupNumber() == 0 && jObject.getGroupNumber() != 0) {
              iObject.setGroupNumber(jObject.getGroupNumber());
              listOfAliasSets.get(jObject.getGroupNumber() - 1).add(iObject);
            } else {
              if (iObject.getGroupNumber() != 0 && jObject.getGroupNumber() == 0) {
                jObject.setGroupNumber(iObject.getGroupNumber());
                listOfAliasSets.get(iObject.getGroupNumber() - 1).add(jObject);
              } else {
                Set<AliasGroupObject> newSet = new HashSet<AliasGroupObject>();
                numGroups += 1;
                iObject.setGroupNumber(numGroups);
                jObject.setGroupNumber(numGroups);
                newSet.add(iObject);
                newSet.add(jObject);
                listOfAliasSets.add(numGroups - 1, newSet);
              }
            }
          }
        }
      }
    }
    long end = System.nanoTime();
  }

  public Accordion createAliasTab() {
    Accordion accordion = new Accordion();
    TitledPane[] titledPaneArray = new TitledPane[listOfAliasSets.size()];
    for (int i = 0; i < titledPaneArray.length; i++) {
```

```
      titledPaneArray[i] = new TitledPane();
      Set<AliasGroupObject> set = listOfAliasSets.get(i);
      VBox vbox = new VBox();
      for (AliasGroupObject object : set) {
        vbox.getChildren().add(new Label(object.getEmailAddress()));
      }
      titledPaneArray[i].setText("Alias " + (i+1));
      titledPaneArray[i].setContent(vbox);
    }
    accordion.getPanes().addAll(titledPaneArray);
    return accordion;
  }

}
```

Listing C.2: MessageRatesScene

```
public class MessageRatesScene {

  DatabaseConnector conn;
  ArrayList<String> whitelistList;
  LineChart<Date, Number> lineChart;
  DateAxis xAxis;
  NumberAxis yAxis;
  String selectedGroup;
  HashMap<String, MessageRateGroup> emailsInGroupHashMap;
  HashMap<String, EmailGroupObject> seriesForEmailHashMap;
  ListView<GroupListViewObject> groupsList;
  Rectangle2D screenSize;
  String timeFormat;
  int typeOfData;

  public MessageRatesScene(DatabaseConnector conn, ArrayList<String> whitelistList) {
    this.conn = conn;
    this.timeFormat = "%Y-%m-%W";
    this.typeOfData = Calendar.WEEK_OF_YEAR;
    this.whitelistList = whitelistList;
    this.xAxis = new DateAxis();
    this.yAxis = new NumberAxis();
    this.lineChart = new LineChart<Date, Number>(xAxis, yAxis) {
      { setLegendSide(Side.RIGHT); }
    };
    screenSize = Screen.getPrimary().getVisualBounds();
    lineChart.setTitle("Message rates by person");
    Rectangle2D screen = Screen.getPrimary().getVisualBounds();
    lineChart.setMinHeight(2*screen.getHeight()/3);
    lineChart.setPrefWidth(screen.getWidth());
    emailsInGroupHashMap  = new HashMap<String, MessageRateGroup>();
    seriesForEmailHashMap = new HashMap<String, EmailGroupObject>();
  }

  public VBox graphSceneCreator() {
    for (String emailAddress : whitelistList) {
      seriesForEmailHashMap.put(emailAddress, new EmailGroupObject(emailAddress));
    }
    ...
    HBox userInteractionBox = new HBox(20);
    userInteractionBox.setAlignment(Pos.CENTER);
    userInteractionBox.getChildren().addAll(userToSelectBox, customGroupGrid, userAddToGraphBox, dateRangeBox)
        ;

    VBox overallBox = new VBox();
    overallBox.getChildren().addAll(lineChart, userInteractionBox);
    return overallBox;
  }

  public GroupListViewObject createNewListViewObject(final String emailAddress) {
    final GroupListViewObject item = new GroupListViewObject(emailAddress, false);
```

```
    item.onProperty().addListener(new ChangeListener<Boolean>() {
      public void changed(ObservableValue<? extends Boolean> ov, Boolean old_val, Boolean new_val) {
        MessageRateGroup group = emailsInGroupHashMap.get(emailAddress);
        if (new_val) {
          ObservableList<String> emailAddresses = group.getEmailList();
          if (!(emailAddresses.isEmpty()) || emailAddress == null) {
            ObservableList<XYChart.Data<Date, Number>> seriesData = conn.selectIncomingAndOutgoingEmails(
                emailAddresses, timeFormat);
            ObservableList<XYChart.Data<Date, Number>> paddedData = padSeriesData(seriesData, typeOfData);
            XYChart.Series<Date, Number> newSeries = new XYChart.Series<Date, Number>(emailAddress, paddedData
                );
            group.setSeries(newSeries);
            lineChart.getData().add(newSeries);
          } else {
            System.err.println("There are no email addresses in the group");
            item.setOn(false);
            groupsList.refresh();
          }
        } else {
          lineChart.getData().remove(group.getSeries());
        }
      }
    });
    return item;
  }

  public ObservableList<XYChart.Data<Date, Number>> padSeriesData(ObservableList<XYChart.Data<Date, Number>>
      seriesData, int typeOfData) {
    ObservableList<XYChart.Data<Date, Number>> paddedSeriesData = FXCollections.observableArrayList();
    for (int i = 0; i<seriesData.size(); i++) {
      Date date = seriesData.get(i).getXValue();
      if (i == 0) {
        System.out.println("Adding first point");
        paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, -1, typeOfData), 0));
      } else {
        if (!paddedSeriesData.get(paddedSeriesData.size()-1).getXValue().equals(changeDate(date, -1,
            typeOfData))) {
          paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, -1, typeOfData), 0));
        }
      }
      paddedSeriesData.add(seriesData.get(i));
      if (i == (seriesData.size() -1)) {
        System.out.println("Adding last point");
        paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, 1, typeOfData), 0));
      } else {
        if (!seriesData.get(i+1).getXValue().equals(changeDate(date, 1, typeOfData))) {
          paddedSeriesData.add(new XYChart.Data<Date, Number>(changeDate(date, 1, typeOfData), 0));
        }
      }
    }
    System.out.println(paddedSeriesData);
    return paddedSeriesData;
  }
  ...
}
```

Listing C.3: GroupingScene

```
public class GroupingScene {

  private HBox hBox;
  private DatabaseConnector conn;
  private ArrayList<String> whitelistList;
  private Hashtable<Integer, Set<String>> groupToClusterHashTable;
  private Hashtable<String, Integer> emailToGroupHashTable;
  private Color[] colours;

  public GroupingScene(DatabaseConnector conn, ArrayList<String> whitelistList) {
```

```
  this.conn = conn;
  this.whitelistList = whitelistList;
  emailToGroupHashTable = new Hashtable<String, Integer>();
  groupToClusterHashTable = new Hashtable<Integer, Set<String>>();
  hBox = new HBox(5);
}

public void makeGraph() {
  SwingNode swingNode = new SwingNode();
  createAndSetSwingContent(swingNode);
  hBox.getChildren().addAll(createWhitelistView(), swingNode);
}

public void createAndSetSwingContent(final SwingNode swingNode) {
  SwingUtilities.invokeLater(new Runnable() {
    @Override
    public void run() {
      Graph<String, EdgeObject> graphToDisplayClusters = new SparseMultigraph<String, EdgeObject>();
      ArrayList<EdgeObject> clusterVisualizationEdges = conn.getAllEdgesForClusteringDisplay(whitelistList);
      LinkedList<Set<String>> voltageClusters = conn.getUserGroups();
      /* Adds all vertices to hashtable together with group number for fast retrieval */
      int j = 0, maxCount = 0;
      for (Iterator<Set<String>> cIt = voltageClusters.iterator(); cIt.hasNext(); j++) {
        Set<String> vertices = cIt.next();
        for (Iterator<String> iterator = vertices.iterator(); iterator.hasNext();) {
          String string = iterator.next();
          System.out.println(string);
          emailToGroupHashTable.put(string, j+1);
        }
        for (EdgeObject edge : clusterVisualizationEdges) {
          if (edge.getReceiver() != null) {
            if (vertices.contains(edge.getSender()) && (vertices.contains(edge.getReceiver()))) {
              if (edge.getSize() > maxCount) maxCount = edge.getSize();
              graphToDisplayClusters.addEdge(edge, edge.getSender(), edge.getReceiver(), EdgeType.DIRECTED);
            }
          }
        }
        groupToClusterHashTable.put(j+1, vertices);
      }
      ...
    }
  });
}

public void generateGroups() {
  Graph<String, EdgeObject> graphToIdentifyClusters = new SparseMultigraph<String, EdgeObject>();
  for (String user : whitelistList) {
    graphToIdentifyClusters.addVertex(user);
  }
  ArrayList<EdgeObject> clusterCreationEdges = conn.getAllEdgesForClusteringCreation(whitelistList);
  for (EdgeObject edge : clusterCreationEdges) {
    if (edge.getReceiver() != null) {
      graphToIdentifyClusters.addEdge(edge, edge.getSender(), edge.getReceiver(), EdgeType.DIRECTED);
    }
  }
  VoltageClusterer<String, EdgeObject> voltageClusterer = new VoltageClusterer<String, EdgeObject>(
      graphToIdentifyClusters, graphToIdentifyClusters.getVertexCount()/10);
  LinkedList<Set<String>> voltageClusters = (LinkedList<Set<String>>) voltageClusterer.cluster(
      graphToIdentifyClusters.getVertexCount()/10);
  System.out.println("There␣are␣" + voltageClusters.size() + "␣clusters,␣and␣" + graphToIdentifyClusters.
      getEdgeCount() + "␣edges");
  conn.dropGroups();
  conn.commit();
  int groupNumber = 1;
  for (Iterator<Set<String>> cIt = voltageClusters.iterator(); cIt.hasNext();) {
    Set<String> vertices = cIt.next();
    System.out.println(vertices.toString());
    for (Iterator<String> iterator = vertices.iterator(); iterator.hasNext();) {
```

```
        String email = iterator.next();
        System.out.print("Group number: " + groupNumber + " Email: " + email + " ");
        System.out.println(conn.insertIntoGroupStatement(groupNumber, email));
      }
      groupNumber++;
    }
    conn.commit();
  }
  ...
}
```

Listing C.4: RankingScene

```
public class RankingScene {

  DatabaseConnector conn;
  ArrayList<String> whitelistList;
  Graph <String, GraphEdge> rankEmailGraph;
  Graph <String, String> singleEdgeGraph;
  HashMap<String, Double> emailRankHashMap;
  HBox rankingBox;
  MySwingNode mySwingNode;

  public RankingScene(DatabaseConnector conn, ArrayList<String> whitelistList) {
    this.conn = conn;
    this.whitelistList = whitelistList;
    emailRankHashMap = new HashMap<String, Double>();
  }

  public HBox getRankingBox() {
    return this.rankingBox;
  }

  private void setChosenVertex(String email) {
    mySwingNode.setChosenVertex(email);
    mySwingNode.changeSelectedVertex();
  }

  public void makeGraph() {
    VBox rankingVBox = rankingVBox();
    singleEdgeGraph = singleEdgeGraphCreator();
    mySwingNode = new MySwingNode(singleEdgeGraph, emailRankHashMap);
    createAndSetSwingContent(mySwingNode);
    rankingBox = new HBox(20);
    rankingBox.getChildren().addAll(rankingVBox, mySwingNode);
  }

  private VBox rankingVBox() {
    final Label employeePosition = new Label("Employee Position");
    employeePosition.getStyleClass().add("file-label");
    final Label employeePositionValue = new Label("-");
    employeePositionValue.getStyleClass().add("file-label");
    final Label rawRank = new Label("rawRank");
    rawRank.getStyleClass().add("file-label");
    final Label rawRankValue = new Label("-");
    rawRankValue.getStyleClass().add("file-label");

    VBox leftMenu = new VBox(20);
    ComboBox<String> rankingListView = new ComboBox<String>();
    rankingListView.setMinSize(300,20);
    rankingListView.setVisibleRowCount(40);
    final ObservableList<String> emailRankList = FXCollections.observableArrayList();
    rankingListView.setItems(emailRankList);
    rankingListView.setPromptText("Email address");
    final DecimalFormat df = new DecimalFormat("#.#####");

    emailRankList.addAll(conn.getUsersWithRanking());
    rankingListView.valueProperty().addListener(new ChangeListener<String>() {
```

```
    @Override
    public void changed(ObservableValue<? extends String> ov, String t, String newValue) {
      employeePositionValue.setText(Integer.toString(emailRankList.indexOf(newValue) + 1));
      rawRankValue.setText(df.format(emailRankHashMap.get(newValue)).toString());
      setChosenVertex(newValue);
    }
  });

  emailRankHashMap = conn.getRanking(whitelistList);
  leftMenu.getChildren().addAll(rankingListView, employeePosition, employeePositionValue, rawRank,
      rawRankValue);

  return leftMenu;
}

public void computeRankings() {
  rankEmailGraph = rankgraphCreator();
  BetweennessCentrality<String, GraphEdge> ranker = new BetweennessCentrality<String, GraphEdge>(
      rankEmailGraph, true, false);

  ranker.setRemoveRankScoresOnFinalize(false);
  ranker.evaluate();

  int vertexCount = rankEmailGraph.getVertexCount();
  double maxRanking = ranker.getRankings().get(0).rankScore;
  double minRanking = ranker.getRankings().get(vertexCount-1).rankScore;

  for (int j = 0; j < vertexCount; j++) {
    String emailAddress = (String)ranker.getRankings().get(j).getRanked();
    Double normalizedRank = normalizeRankings(ranker.getRankings().get(j).rankScore, maxRanking, minRanking)
        ;
    emailRankHashMap.put(emailAddress, normalizedRank);
    conn.updateRanking(emailAddress, normalizedRank);
  }
  conn.commit();
}
...
private Graph<String, String> singleEdgeGraphCreator() {
  Graph<String, String> emailGraph = new SparseMultigraph<String, String>();
  ArrayList<GraphEdge> edges = conn.rankingEdges(whitelistList);
  HashSet<GraphEdge> graphEdgeHashSet = new HashSet<GraphEdge>();
  for (int i = 0; i < edges.size(); i++) {
    graphEdgeHashSet.add(edges.get(i));
  }
  Iterator<GraphEdge> iterator = graphEdgeHashSet.iterator();
  for (String user : whitelistList) {
    emailGraph.addVertex(user);
  }
  while (iterator.hasNext()) {
    GraphEdge edge = iterator.next();
    emailGraph.addEdge(edge.getSenderRecipient(), edge.getSender(), edge.getRecipient());
  }
  return emailGraph;
}

public Double normalizeRankings(Double rank, Double maxRanking, Double minRanking) {
  return (rank - minRanking)/(maxRanking-minRanking); // Scaled ranking so min(rank)=0 and max(rank)=1
}

private Graph<String, GraphEdge> rankgraphCreator() {
  Graph<String, GraphEdge> emailGraph = new SparseMultigraph<String, GraphEdge>();
  ArrayList<GraphEdge> edges = conn.rankingEdges(whitelistList);
  for (String user : whitelistList) {
    emailGraph.addVertex(user);
  }
  boolean edgeAdded = false;
  for (int i = 0; i < edges.size(); i++) {
    GraphEdge currentEdge = edges.get(i);
```

```
      edgeAdded = emailGraph.addEdge(currentEdge, currentEdge.getSender(), currentEdge.getRecipient(),
          EdgeType.DIRECTED);
      if (!edgeAdded) System.out.println(currentEdge.getEmailRecipientHash());
    }
    return emailGraph;
  }

}
```