E-MAIL ANALYSIS FOR INVESTIGATORS:

TECHNIQUES AND IMPLEMENTATION

ADAM SZPORER

A THESIS

IN

THE DEPARTMENT

OF

CONCORDIA INSTITUTE FOR INFORMATION SYSTEMS ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF APPLIED SCIENCE IN

INFORMATION SYSTEMS SECURITY

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

FEBRUARY 2012

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Adam Szporer**

Entitled: **E-mail Analysis for Investigators: Techniques and Implementation**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science in Information Systems Security

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. S. Li, Chair

_____ Dr. M. Debbabi, Supervisor

_____ Dr. B. Fung, CIISE Examiner

_____ Dr. O. Ormandjieva, External Examiner

Approved _____

Chair of Department or Graduate Program Director

_____ 20 ____ _____

Dr. Robin Drew, Dean

Faculty of Engineering and Computer Science

Abstract

E-mail Analysis for Investigators: Techniques and Implementation

Adam Szporer

E-mail is a common form of communication in regular use today. As such, it is a normal part of investigating a person or a crime. At present, there are many tools to perform bulk analysis and basic searching, but our research advances the state of the art by applying text mining and unsupervised learning techniques to automate the e-mail analysis process. Our key goals are to group similar e-mails together and to identify the concepts (subjects of discussion) of those e-mail groups. We present several new methods to increase the grouping accuracy: e-mail domain analysis and word pair analysis. We also present a technique for concept analysis. These goals are achieved by integrating our research with the capabilities of Weka, an open-source machine learning suite, and WordNet, a lexical database of the English language. We apply this research to the publicly available Enron e-mail dataset. We verify the results by examining the comparative advantage of each new technique.

Acknowledgments

It is with great pleasure that I thank my supervisor, Prof. Dr. Mourad Debbabi, for his guidance and motivation these last few years. Working with him has helped to shape a very important part of my life and I am very grateful for the opportunities, challenges, and experiences that followed. Additional thanks go to Prof. Dr. Hakim Lounis who co-supervised part of my research.

A very special thanks to all of my colleagues and friends at Concordia, especially Ann Fry and James Grabowski, who have provided me with immeasurable counsel.

Even more thanks go to all of my family and friends for their support and understanding during the course of my studies, particularly Mom & Dad, Ryan, Matthew, Hugo, and Shosh.

And, lastly, I would like to thank you, the reader. Without you there wouldn't have been much point in writing a thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

From designing a better spam filter to automatically classifying thousands of messages, e-mail analysis is a growing area of research. Focusing on digital investigations, this thesis advances the state of the art with a new approach, and verifies that approach against a known corpus of e-mail.

1.1 Motivations

E-mail is an ubiquitous form of communication that may be used for a variety of purposes, such as:

- Personal communication
- Group communication
- Financial transactions and e-commerce

- Electronic crimes (e.g., spreading of malicious software or phishing)
- Traditional crimes (e.g., communicating with accomplices or a victim)

In 2002, a total of 31 billion e-mails were sent daily, totaling over 600 petabytes. The same report projected that in 2006 half of those would be unsolicited commercial e-mails (spam) [22].

There are several reasons to analyze e-mail, such as:

- Criminal investigation by law enforcement
- e-Discovery for a civil lawsuit
- Internal corporate investigations into fraud or abuse

Digital communication has become a normal part of everyday life. As such, we can expect to find a digital component of any modern investigation. Existing research into text mining and clustering has not focused on e-mails, nor has research into concept analysis. Given the large volume of information exchanged using this medium, we judge it useful and pertinent to develop better approaches for e-mail analysis.

1.2 Problem Statement

While small amounts of e-mail–tens or hundreds of messages–are comparatively easy for a human investigator to analyze, this task becomes increasingly difficult with for larger quantities. Furthermore, the state of the art in text mining does not focus on e-mails, nor does the state of the art in concept analysis. Digital investigations involving electronic mail

can benefit from research in other fields. We would like to develop automated techniques for analyzing large quantities of e-mail with little or no operator intervention.

Therefore, we present the following questions that this thesis will answer:

- What is the current research in text mining, clustering, and concept analysis?
- What is the current research in e-mail analysis?
- How can we advance the state of the art in e-mail analysis using existing research?
- What new approaches can we develop for e-mail analysis?
- How can these approaches be implemented in software?

The following features are desirable in a comprehensive e-mail analysis tool:

- Ability to inspect e-mail archives and related information by browsing through multiple data sources;
- Developing data mining models to classify e-mails into well-established categories, or to cluster them according to unknown relationships;
- Computing dependencies between users and cliques of users using social networking techniques;
- Searching through e-mails using keywords;
- Extracting *cyber DNA* which will be used for authorship analysis;
- Analyzing the concepts in an e-mail to identify the subject matter of the e-mails being exchanged.

Our research focuses on clustering of e-mails, concept analysis, and a graphical interface. Through e-mail clustering our software is able to group similar messages. This software algorithm associates related e-mails together in order to present coherent groups to an investigator. Concept analysis of the clusters allows our software to identify the subject matter being discussed and to extract keywords that will help an investigator choose which groups to examine more closely. Lastly, a graphical user interface is an essential element for investigators. It will allow easy navigation through the evidence that has been collected in a natural way that requires little additional training.

1.3 Thesis Contribution

The main contributions of this thesis are summarized as follows:

- The application of text mining techniques to e-mail: We apply text mining techniques that already exist for document classification to e-mail messages.
- E-mail domain analysis: We present a new method of analyzing e-mail addresses to help group related content.
- Word pair analysis: We present a new method of e-mail content analysis that reduces the computational overhead required for analysis with little change in the results.

Additionally, we verify our approach by applying it to the Enron e-mail corpus, which is a publicly-available collection of e-mail, and we present an example graphical interface that shows how we envision information being presented to a digital investigator.

1.4 Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2 describes the format of an e-mail and provides the information necessary for the reader to understand technical concepts such as the e-mail header format, the e-mail body format, and MIME extenstions.

Chapter 3 reviews relevant research in the areas of text-mining, document pre-procesing, clustering, concept analysis, authorship analysis, and metrics for measuring success.

Chapter 4 explains the elements from the state of the art that we retain for use in our software, and our new innovations that push the state of the art forward. It is subdivided into the e-mail analysis process, data extraction, pre-processing, clustering, and concept analysis. We present our new techniques of e-mail domain analysis and word pair analysis. We also present our approach for concept analysis.

Chapter 5 provides technical implementation details for the approach described in the fourth chapter, and examples of some algorithms. It also presents the proof-of-concept graphical interface.

Chapter 6 details the results of applying our approach to a selected part of the publicly-available Enron e-mail corpus. It examines the effect some of our innovations have on accuracy.

Chapter 7 concludes the thesis and discusses as future work some ideas that are not implemented yet.

Chapter 2

Background

In this chapter, we present the structured format of an e-mail message, and show how e-mails are transferred from sender to recipient. Moreover, we discuss the MIME extensions to e-mail messages and how this relates to data extraction. Finally, we explain the concept of *inverse weighting*, which is frequently used in text mining and data analysis.

2.1 E-mail Overview

An e-mail message is a sequence of bytes in a specific format designed to convey information from one entity to another using the Simple Mail Transport Protocol (SMTP), generally over the internet or a private network using internet protocols. An e-mail message is analogous to a conventional letter sent by regular mail (the post office):

- A regular letter is sent inside an envelope.

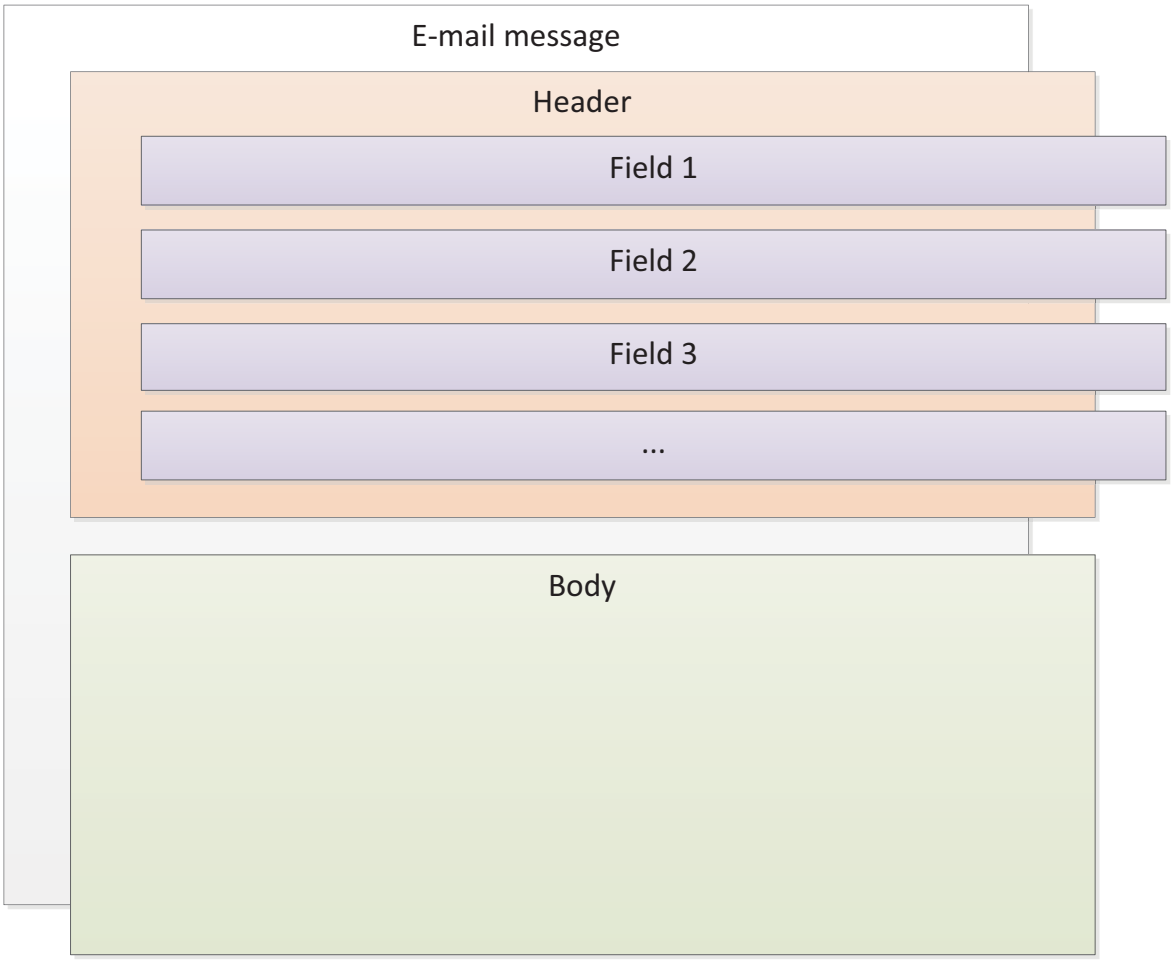

Figure 1: E-mail Message Parts

- The envelope contains information in a well-defined format that ensures proper delivery of the message.

- The envelope is annotated as it passes through the postal system.

Like a letter, an e-mail message has two parts: a *header* and a *body*, as shown in Figure 1. The header follows a well-defined format to ensure proper delivery while the body is free-form and intended for the recipient.

The format of an e-mail message was originally defined by a *request for comments* entitled *Standard for the Format of ARPA Internet Text Messages* (commonly known as RFC 822 [8]), published in August 1982. The most recent incarnation is entitled *Internet Message Format*, numbered RFC 5322 [7].

2.2 E-mail Flow

In order to understand how e-mail flows from senders to recipients, we explain two important concepts: the Mail User Agent (MUA) and Mail Transfer Agent (MTA). These were originally defined in the e-mail specification and are essential to understanding how e-mail works.

Mail User Agent (MUA) is a a client application. This application communicates with a remote transport agent to send and receive e-mail. A mail user agent is frequently called a *mail client*.

Mail Transfer Agent (MTA) is a server-side application responsible for transporting e-mails to their destination. A mail transfer agent is frequently called a *mail server*.

In the early days of the internet, it was possible for a message to pass through many different mail transfer agents on its way from sender to recipient. Each transport agent would add a little bit of information to show the route that the message took through the internet. This behaviour is suggested by the e-mail specification, but not required. There exist many agents that act “silently”, such as certain junk mail filters.

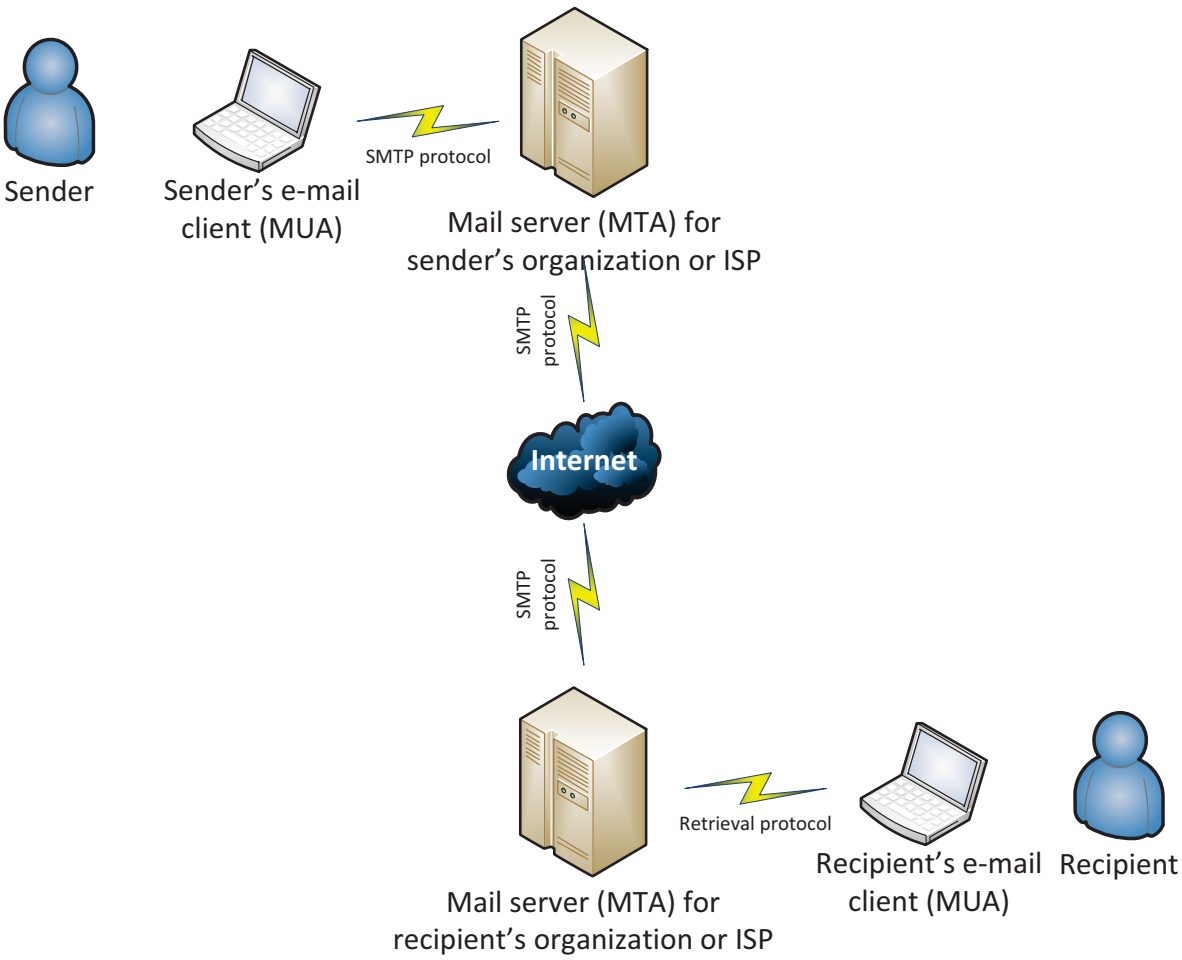

Figure 2: E-mail Flow

By comparison, in today's robust internet where all public mail transfer agents are generally available full-time, e-mail messages frequently pass directly from the sender's internet-facing MTA to the recipient's internet-facing MTA, as shown in Figure 2.

New features, such as anti-spam and anti-virus scanning, are frequently implemented as mail transfer agents. In this way, they are integrated into the e-mail flow as shown in Figure 3.

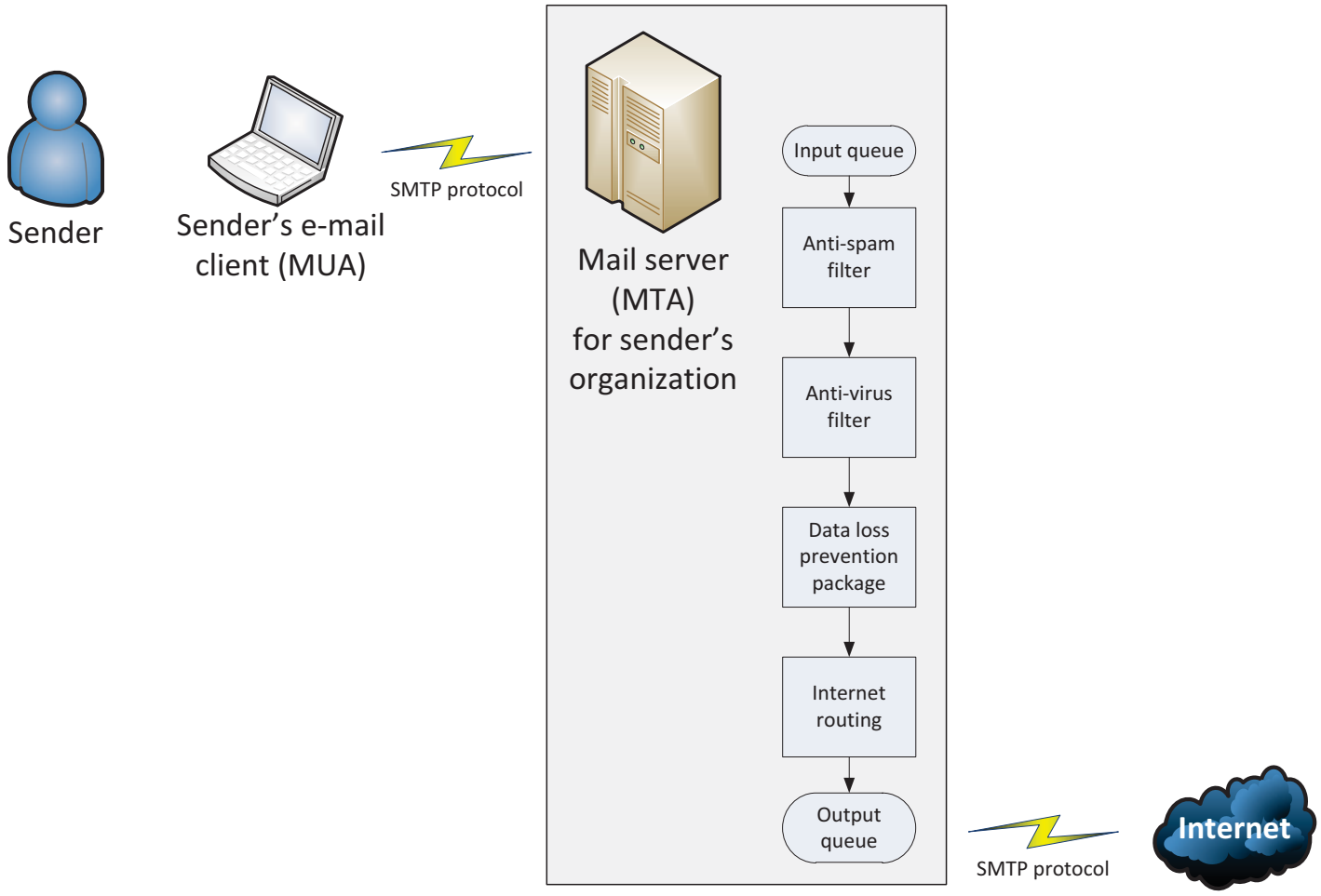

Figure 3: Multiple Mail Transfer Agents

2.3 E-mail Header

The header contains metadata about the e-mail. Among other information, it describes the:

- Sender of the message
- Recipient of the message
- Creation date
- Message route

The e-mail header is organized into *fields*. Each header field contains a *field name* and a *field body*. Some field names can appear more than once, and others must be unique.

A header field is comprised of a field name, followed by a colon, followed by a field body, and terminated with the carriage return and line feed characters (US-ASCII characters 13 and 10, respectively). For each field name, the corresponding field body may be unstructured text terminating with a carriage return and line feed, or may follow a specific format [7]. For example:

- Date fields follow the format *Mon, 24 Aug 2010 16:30:00 -0400*
- E-mail address fields follow the format *"John Doe" <jdoe@example.net>*

As an e-mail passes from computer to computer on its way to the recipient, each node (by convention) adds its own information to the top of the header. It is therefore important to note that the header is in *reverse* chronological order.

Here is an example of an e-mail header (some user names have been altered):

```
X-Spam-Checker-Version: SpamAssassin 3.2.3 (2007-08-08) Exchange
  SpamAssassin Sink (www.christopherlewis.com) 1.2.81 on owl.szporer.ca
X-Spam-Level:
X-Spam-Status: No, score=-3.3 required=6.3 tests=AWL,BAYES_00
  shortcircuit=noautolearn=ham version=3.2.3
Received: from owl.szporer.ca ([10.2.1.6]) by owl.szporer.ca with
  Microsoft SMTPSVC(5.0.2195.6713); Wed, 20 Feb 2008 11:30:01 -0500
Delivered-To: example.net-adam@example.net
Received: from mail.servak.net [38.113.161.116] by owl.szporer.ca
  with POP3 (fetchmail-6.3.8 polling mail.servak.net account
  adam-pop@example.net) for <ajs@szporer.ca> (single-drop);
  Wed, 20 Feb 2008 11:30:01 -0500 (EST)
Received: (qmail 29681 invoked from network); 20 Feb 2008 16:25:23 -0000
Received: from anis.telecom.uqam.ca (132.208.250.6)  by new2.411.ca
  with SMTP; 20 Feb 2008 16:25:23 -0000
Received: from anis.telecom.uqam.ca (anis.telecom.uqam.ca
  [132.208.250.6]) by sortant.uqam.ca (8.13.8/8.12.1) with SMTP id
  m1KGJmjw002588 for <adam@example.net>; Wed, 20 Feb
  2008 11:20:06 -0500 (EST)
Received: from antivirus.uqam.ca ([127.0.0.1]) by anis.telecom.uqam.ca
  (SAVSMTP 3.1.1.32) with SMTP id M2008022011200604695 for
  <adam@example.net>; Wed, 20 Feb 2008 11:20:06 -0500
Received: from [132.208.137.54] ([132.208.137.54]) by intrant.uqam.ca
  (8.13.8/8.12.2/uqam-filtres) with ESMTP id m1KGIq5T026232 for
  <adam@example.net>; Wed, 20 Feb 2008 11:18:54 -0500 (EST)
X-UQAM-Spam-Filter: Filtre-Uqam re:  abuse@uqam.ca
Content-Class: urn:content-classes:message
Importance: normal
Priority: normal
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1914
Message-ID: <47BC52E6.5050303@uqam.ca>
Date: Wed, 20 Feb 2008 11:18:46 -0500
From: "John Doe" < j_doe@uqam.ca>
User-Agent: Thunderbird 2.0.0.9 (Windows/20071031)
MIME-Version: 1.0
To: "Adam Szporer" <adam@example.net>
Subject: Re: Meeting on Friday
References: <4390CD2C52C0474CB69F17C2A4B3452F081E0E@owl.szporer.ca>
In-reply-to: <4390CD2C52C0474CB69F17C2A4B3452F081E0E@owl.szporer.ca>
Content-Type: text/plain;
format=flowed;
charset="ISO-8859-1"
Content-Transfer-Encoding: quoted-printable
X-MIME-Autoconverted: from 8bit to quoted-printable by
  antivirus.uqam.ca id m1KGIq5T026232
Return-Path: <j_doe@uqam.ca>
X-OriginalArrivalTime: 20 Feb 2008 16:30:01.0707 (UTC)
  FILETIME=[D74B6BB0:01C873DD]
```

We now show you the same header, with annotated comments:

```
X-Spam-Checker-Version: SpamAssassin 3.2.3 (2007-08-08) Exchange
  SpamAssassin Sink (www.christopherlewis.com) 1.2.81 on owl.szporer.ca
X-Spam-Level:
X-Spam-Status: No, score=-3.3 required=6.3 tests=AWL,BAYES_00
  shortcircuit=noautolearn=ham version=3.2.3
```

The following (most recent) portion of the e-mail header is added by an anti-SPAM filter on the recipient's mail server.

```
Received: from owl.szporer.ca ([10.2.1.6]) by owl.szporer.ca with
  Microsoft SMTPSVC(5.0.2195.6713); Wed, 20 Feb 2008 11:30:01 -0500
Delivered-To: example.net-adam@example.net
Received: from mail.servak.net [38.113.161.116] by owl.szporer.ca
  with POP3 (fetchmail-6.3.8 polling mail.servak.net account
  adam-pop@example.net) for <ajs@szporer.ca> (single-drop);
  Wed, 20 Feb 2008 11:30:01 -0500 (EST)
```

The following portion of the e-mail header is added when the message is retrieved from the recipient's server and delivered to his inbox.

```
Received: (qmail 29681 invoked from network); 20 Feb 2008 16:25:23 -0000
Received: from anis.telecom.uqam.ca (132.208.250.6)  by new2.411.ca
  with SMTP; 20 Feb 2008 16:25:23 -0000
Received: from anis.telecom.uqam.ca (anis.telecom.uqam.ca
  [132.208.250.6]) by sortant.uqam.ca (8.13.8/8.12.1) with SMTP id
  m1KGJmjw002588 for <adam@example.net>; Wed, 20 Feb
  2008 11:20:06 -0500 (EST)
```

The following header lines represent the actual transfer of the e-mail from the sender's mail server to the recipient's mail server. These first (oldest) lines show the delivery of the message from the sender's mail client to the sender's mail server. Note that in this case, the sender's mail server has performed anti-virus checking.

```
Received: from antivirus.uqam.ca ([127.0.0.1]) by anis.telecom.uqam.ca
  (SAVSMTP 3.1.1.32) with SMTP id M2008022011200604695 for
  <adam@example.net>; Wed, 20 Feb 2008 11:20:06 -0500
Received: from [132.208.137.54] ([132.208.137.54]) by intrant.uqam.ca
  (8.13.8/8.12.2/uqam-filtres) with ESMTP id m1KGIq5T026232 for
  <adam@example.net>; Wed, 20 Feb 2008 11:18:54 -0500 (EST)
X-UQAM-Spam-Filter: Filtre-Uqam re:  abuse@uqam.ca
```

This initial header segment is generated by the sender's mail client and contains the information required for delivery by the sender's mail server.

```
Content-Class: urn:content-classes:message
Importance: normal
Priority: normal
X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1914
Message-ID: <47BC52E6.5050303@uqam.ca>
Date: Wed, 20 Feb 2008 11:18:46 -0500
From: "John Doe" < j_doe@uqam.ca>
User-Agent: Thunderbird 2.0.0.9 (Windows/20071031)
MIME-Version: 1.0
To: "Adam Szporer" <adam@example.net>
Subject: Re: Meeting on Friday
References: <4390CD2C52C0474CB69F17C2A4B3452F081E0E@owl.szporer.ca>
```

```
In-reply-to: <4390CD2C52C0474CB69F17C2A4B3452F081E0E@owl.szporer.ca>
Content-Type: text/plain;
format=flowed;
charset="ISO-8859-1"
Content-Transfer-Encoding: quoted-printable
X-MIME-Autoconverted: from 8bit to quoted-printable by
  antivirus.uqam.ca id m1KGIq5T026232
Return-Path: <j_doe@uqam.ca>
X-OriginalArrivalTime: 20 Feb 2008 16:30:01.0707 (UTC)
  FILETIME=[D74B6BB0:01C873DD]
```

2.4 E-mail Body

The *Internet Message Format* standard does not specify a format for the body, as it is outside the scope of that document. In fact, many e-mails are sent with free-form bodies. There are certain limitations to this historical scheme:

- Each body character can only be one out of 127 possibilities (the US-ASCII standard). This is not sufficient for non-English languages.
- There is no standard way to incorporate attachments to an e-mail.
- Advanced formatting is not possible (different fonts, colours, styles).

The changing nature of e-mail necessitated an extension to the standard. This was accomplished in 1996 in the form of the *Multipurpose Internet Mail Extensions* [10].

The *Multipurpose Internet Mail Extensions* (MIME) [10] impose a backward-compatible structure on the e-mail body. Rather than being free-form, it is divided into multiple parts. Each part has an associated *content type*, which is encapsulated inside the original message body. Using MIME, one can even encapsulate another e-mail message (as in the case of forwarding a mail). These extensions to the internet mail standard allow many new capabilities:

- Textual message bodies in many character sets
- Different formats for non-textual message bodies
- Multi-part message bodies

This encapsulation scheme is illustrated in Figure 4. For example, a message may contain a colourful HTML part, a plain-text part (for clients that do not support HTML messages), and a file attachment. Such a message would be encoded in three parts. Parts are tagged with a *MIME type* to help mail clients correctly interpret the content. Examples of textual MIME types are "text/plain" and "text/html". A stub message is left at the beginning of the body so that a mail client that does not understand MIME will present the user with a friendly explanation.

2.5 Inverse Weighting ($tf * idf$)

Inverse weighting is a statistical method of ranking terms (such as words) in a collection of documents. An example of inverse weighting is the *term frequency — inverse document frequency* method, or $tf * idf$ [30]. As with all inverse weighting schemes, frequently-used words are assigned a low weight and infrequently-used words are assigned a high weight.

Consider a collection of *terms* (words) T in a collection of documents D. The *term frequency* (tf) of a particular term t in a document d is given by:

$$tf(t, d) = \text{number of times } t \text{ occurs in } d$$

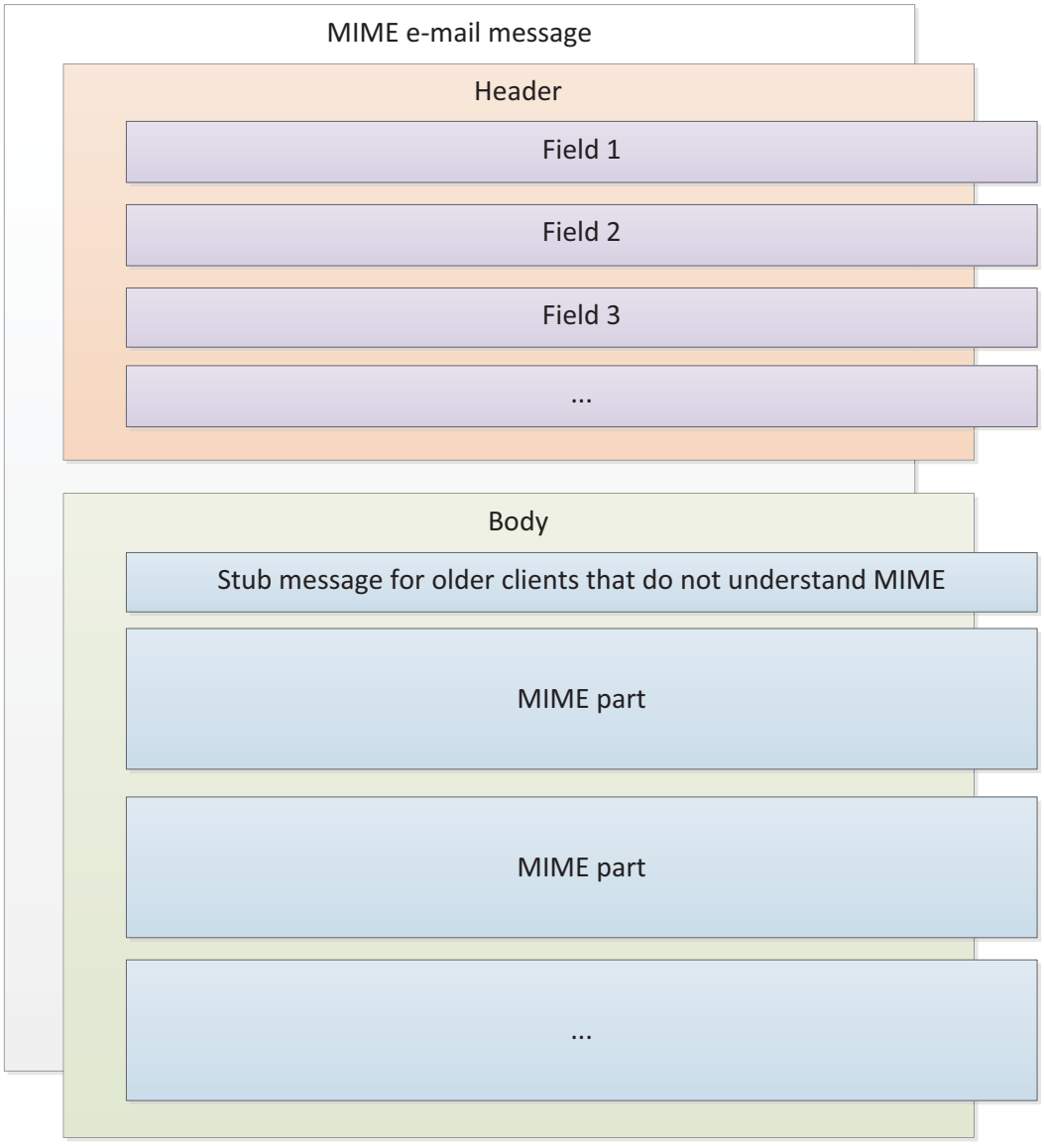

Figure 4: MIME Message Parts

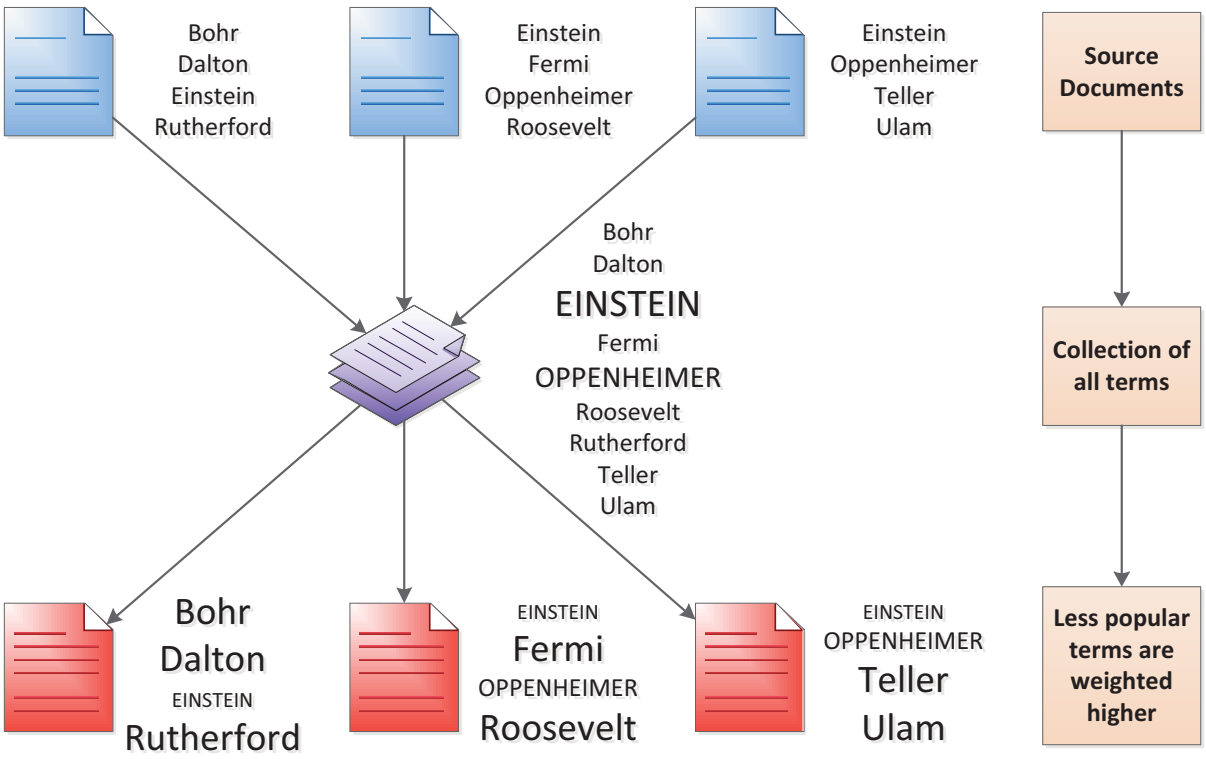

Figure 5: Inverse Weighting Example

The *inverse document frequency* (idf) of a particular term t is given by:

$$idf(t) = log\frac{count(D)}{count(\text{documents containing } t)}$$

Therefore, the inverse weighting $tf * idf$ gives a high value to a term that occurs often in a particular document, but is rare in the set of documents. Additionally, a frequently used word in a particular document has a low value if it is also frequently used in the set. This scheme allows us to "pick out" important words. The principle of inverse weighting is shown in Figure 5.

2.6 Summary

This chapter has provided information about the format of e-mails, how they are sent through the internet, and the way they are annotated. It has also explained the concept of inverse weighting, which will be of paramount importance to our research.

Chapter 3

Literature Review

Text mining is an active area of academic research. Our goal in this chapter is to thoroughly analyze the existing state of the art and identify the current research that is applicable to e-mail analysis. This literature review focuses on the text mining process, document pre-processing, clustering, concept analysis, authorship analysis, and, finally, ways to measure success.

3.1 Text Mining Process

The text mining process focuses on the individual steps used to turn raw data into useful knowledge. The body of published research uses many variations based on the same principles, such as extraction, pre-processing, clustering, and analysis. This section presents several visions of the text mining process.

Bichindaritz and Akkineni [4] have applied concept mining techniques to medical literature. Their process is divided into six steps:

1. *Data extraction:* Extracts the document as text from its source data storage location.

2. *Syntactic analysis:* Extracts basic lexical information from the text.

3. *Semantic analysis:* Identifies *trigger phrases* in the document.

4. *Relationship selector:* Selects the concepts of each trigger phrase.

5. *Ranking:* Ranks the concepts (relationships).

6. *Self-evaluation:* Automatically compares the results with manual classification for testing purposes.

Tseng et al. [42] have applied text mining techniques to patent analysis. Their approach supports the following steps:

1. *Text segmentation:* Divides a structured patent document into its constituent parts.

2. *Text summarization:* Ranks sentences by their relative importance using a specific algorithm. Based on the summary length desired, lower-ranked sentences are discarded.

3. *Stopwords and stemming:* Removes predefined stopwords that do not add semantic value to a sentence. Stemming reduces verbs to a root form for easier analysis.

4. *Keyword and phrase extraction:* Identifies multiple word phrases.

5. *Term association:* Identifies of relationships among terms.

6. *Topic clustering:* Clusters related documents.

7. *Topic mapping:* Organizes topics (either hierarchically or in a map structure)

To summarize, each step in the text mining process can be thought of as a module with defined input and output specifications. This allows for steps to be added, removed, or changed independently. As shown in Figure 6, one module does not need to know how another works; rather, it need only know what form of input to accept and what form of output to produce. This is advantageous in software design in that changes to one part are transparent to all other parts.

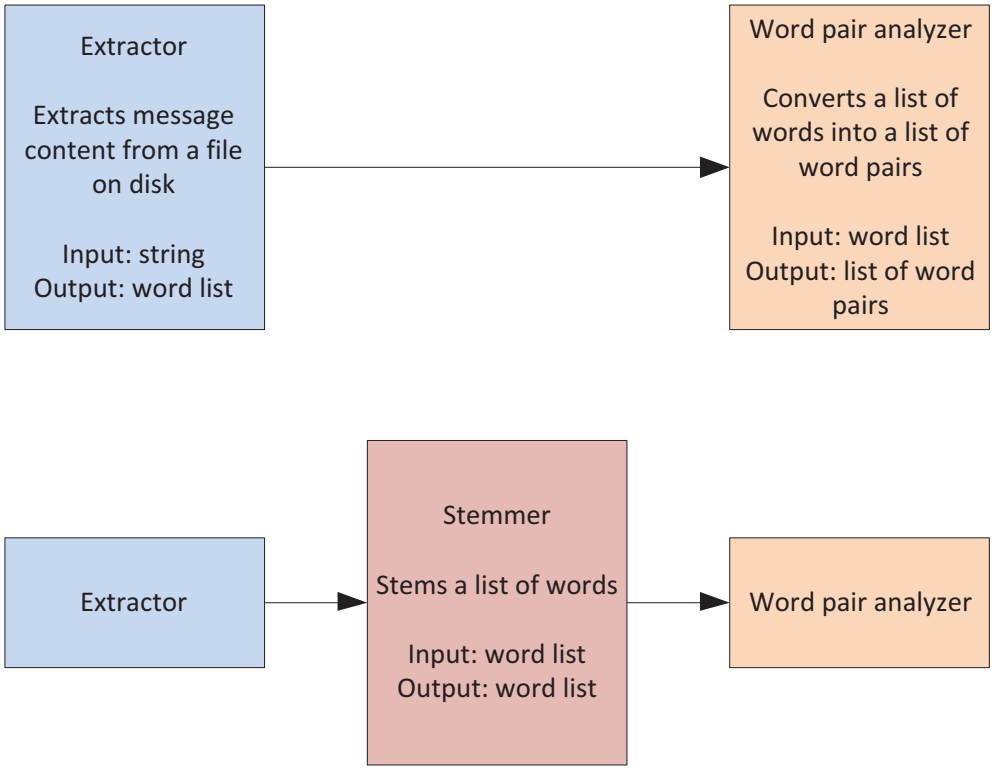

Figure 6: Modular Software Approach

3.2 Document Pre-processing

Pre-processing describes how to turn raw data into units suitable for further analysis. For example, a source e-mail must be extracted from a mailbox. Techniques such as *stop words* and *stemming* (explained later in this chapter) are often used to remove non-relevant information and to reduce the computational effort required. *Document vectors* are a common method used to represent the information in each source data file. Simple syntactic analysis and advanced natural language processing techniques are also employed. Common steps include:

- Removing common words (*stop words*) that might negatively affect the concept analysis (for example, "the", "it", "their", etc.)
- Consolidating terms that span multiple words (for example, "Concordia University")
- Stemming words to their root form (for example, "running" and "runs" will both become "run" for the purposes of our analysis)
- Identifying sentence structure and the grammatical functions of each word

Consider the following natural language sentence:

"The student attends Concordia University."

Pre-processing might reduce this sentence to three tokens:

- student (unchanged)
- attend (stemmed to its root form)

- Concordia University (condensed into one token)

This allows for more efficient computation (five terms have been consolidated into only three).

Dunja Mladenic and Marko Grobelnik's [24] work provides valuable insight into document pre-processing and representation. They have discussed the use of stop lists, pruning methods for low-frequency features, and word sequences.

Lee and Yang [20] have evaluated several techniques (both supervised and unsupervised) for multilingual text classification. Using Chinese characters as an example, the pre-processing phase employs a *segmentation* function to create better functional units. Their preprocessor extracts the names of organizations, people, locations, dates and times, monetary expressions, and percentage expressions.

Zhang [47] has examined several classification strategies, from weakly supervised to strongly supervised learning. To accomplish this, he has used a five-step data treatment process:

1. Extract document entities from the source material.

2. Segment the text into sentences.

3. Parse the sentences into syntactic trees.

4. Convert the trees into a flattened format.

5. Compute features and generate a document vector from the flattened tree.

Khan and Card [30] have applied textual analysis to web pages and then classified them using unsupervised techniques. Their text analysis is performed in the following phases:

1. Identify the individual words in a collection of documents.

2. Eliminate stop words (using a list of 250 words).

3. Stem the remaining words to their lexical root.

4. Weigh the remaining words (using the inverse document frequency method).

5. Classify the document (several classifiers are examined in their research).

Sinka and Corne [36] have used the following approach in their analysis of unsupervised clustering techniques:

1. Extract all words that appear once in the document.

2. Remove all stop words (using Van Rijsbergen's list).

3. Stem words to their lexical root.

4. Record the frequency of each word in a particular document.

Scharl and Bauer [33] have outlined the following process for text analysis of web-based documents:

1. Keyword analysis

    (a) Convert the source document into plain text.

(b) Segment the plain text into units for processing (for example, removing punctuation marks).

(c) Identify and group identical units.

2. Clustering term frequencies.

3. Correspondence analysis.

To summarize, the removal of stop words and stemming are commonly-used steps in pre-processing. In many cases this positively influences the outcome, and in all cases it reduces the computational overhead (time and space). Frequency counting is often employed to produce a workable document vector.

3.3 Clustering

Clustering (or *categorization*, or *classification*) is the grouping of related documents together based on a mathematical model. There are different types of clustering techniques, for example:

- Cluster membership: each document belongs to exactly one cluster, or can belong to multiple clusters.

- Supervision: clustering with operator input (supervised), or clustering without operator input (unsupervised).

- Training: clustering results are based on prior information (training data), or are solely determined by the content to be clustered (no training data).

- Organization: clusters may be organized hierarchically or not.

Our specific area of research suggests the use of an unsupervised algorithm that does not require operator input; however, we will also examine supervised algorithms in this literature review.

Yang [46] has performed a comparative evaluation of several text classifiers:

- Linear Least Squares Fit (LLSF) [45, p. 409]
- k-Nearest Neighbor (kNN) [45, p. 78]
- WORD [46]

He has concluded that kNN and LLSF, which are learning algorithms, outperform WORD, a non-learning algorithm.

Yang and Lee [19] have applied text mining and concept analysis techniques to the World Wide Web, using self-organizing maps (SOM). Their document processing algorithms are relevant to other text mining problems (such as e-mail). Their approach uses the term-frequency weighting method (tf) because no clear experimental advantage is determined for $tf * idf$.

Wei et al. [9] have proposed a collaborative filter-based technique for personal document clustering. By combining knowledge of multiple users' preferences (as *non-content information*), they have attempted to increase clustering effectiveness. They have used the $tf * idf$ method to represent the contents of a document.

Saraçoglu et al. [32] have applied a fuzzy clustering approach to finding similar documents. Their work focuses on determining a similarity measure based on document features. They have used the $tf * idf$ method to extract document features.

Sakurai and Suyama [31] have combined supervised and unsupervised learning in the context of a routing telephone calls in a customer support centre. Using e-mail subjects and bodies, their method classifies the text of an e-mail and extracts key concepts. They have used a fuzzy inductive learning algorithm with a training data set to acquire a basic concept relation dictionary. This produces a decision tree. During the course of their study, their method has not achieved a high enough precision for use in the customer centre; however, their e-mail mining techniques have provided valuable insight into our research.

Pons-Porrata et al. [27] have developed a hierarchical clustering algorithm to discover the topics of textual documents. This algorithm does not require the number of eventual clusters to be known beforehand, and is well suited to incremental processing (such as web-based applications).

Li et al. [21] have proposed two algorithms for clustering, based on frequent word sequences and frequent word meaning sequences. Their work leverages WordNet to reduce word sequences to word meaning sequences.

Beebe and Clark [3] have applied clustering techniques to increase the effectiveness of information retrieval for the purpose of forensic investigations. Their selection of Kohonen's Self-Organizing Map [16] is principally driven by its $O(n)$ time requirement.

Shahnaz et al. [35] have developed a methodology to cluster textual documents into non-overlapping clusters using what they call nonnegative matrix factorization. This unsupervised approach achieves an accuracy varying between of 96 % for two categories, to 55 % for twenty categories.

Adeva and Atxa [2] have applied text mining techniques to detect intrusions in web applications. Using the web application logs as input data, they have developed techniques to identify anomalous behaviour. They have compared the results of three different categorization techniques (Bayes, k-Nearest Neighbours, and Rocchio).

De Vel [5] has developed a technique for classifying files based on their byte-level structure. This can help to identify file types where no metadata is available (such as in MIME attachments).

Tan [39] has worked to improve the k-Nearest Neighbor (kNN) classification method. With the premise that kNN is biased because of a presumption of evenly distributed training data, he has proposed a refinement. Using an F_1 measure of performance, his DragPush classifier increases accuracy by several percentage points (to over 90 % in optimal cases).

Sinka and Corne [36] have investigated unsupervised clustering techniques with a publicly available dataset. To establish a baseline reference, they have used the standard k-means clustering [45, pp. 137–138]. They have presented the following observations:

- Removing stop words is always beneficial.
- Stemming is only slightly beneficial, and in some cases reduces accuracy slightly. In all cases, it provides practical benefits in processing speed and working memory.

- A longer document feature vector is helpful, but in cases where the categories are dissimilar, a shorter document feature vector is also effective.

Masson and Denœux [23] have proposed a variation of the c-means clustering algorithm. Unlike standard clustering which partitions a collection of items into disjoint groups, a fuzzy clustering algorithm allows an item to belong to multiple groups with a different degree of membership. They have applied this technique to medical images with meaningful results.

Tjhi and Chen [41] have proposed an algorithm for automatic categorization of large document collections. Their method, called possibilistic fuzzy co-clustering, demonstrates robustness and efficiency compared to existing algorithms (such as fuzzy c-means).

The *Waikato Environment for Knowledge Analysis* (Weka) [12, 45] is an open-source framework that provides a rich API for data mining and analysis. The Weka environment already implements several clustering algorithms that can be used "out-of-the-box", such as the expectation-maximization classifier. This unsupervised classifier performs well when the number of groups is not known beforehand [25].

To summarize, there are many clustering options available, as shown in Table 1.

3.4 Concept Analysis

Concept analysis (or *summarization*) is one of the final stages in e-mail analysis. It takes as input a refined, usable, document representation (such as a document vector), and provides useful knowledge—the content of a natural language document. Natural language

Table 1: Clustering Methods

Method	**Type**	**Implemented in Weka**
c-Means	Unsupervised	No
Collaborative Filter	Unsupervised	No
DragPush	Supervised	No
Expectation-Maximization	Unsupervised	Yes
Fuzzy Inductive Learning	Supervised	No
Hierarchical	Unsupervised	No
k-Nearest Neighbour	Supervised	No
k-Means	Unsupervised	Yes
Linear Least-Squares Fit	Supervised	No
Non-negative Matrix Factorization	Unsupervised	No
Self-organizing Maps	Unsupervised	No
WORD	Unsupervised	No

techniques are applied, as well as semantic analysis. Concept analysis techniques are frequently based on dictionary models (such as WordNet). Some techniques produce a set of disjoint concepts, while others produce a tree of hierarchically related concepts.

Weng and Liu [44] have applied text mining and concept analysis techniques to propose e-mail reply templates in a customer service environment. They have evaluated decision trees, Bayesian classification, and the k-Nearest Neighbour method. They further define concepts as groups of associated terms:

$$\text{concept} = (\text{term}_1, \text{term}_2, \ldots, \text{term}_n)$$

Adami et al. [1] have worked to cluster documents into a hierarchy of concepts. Their approach generates a taxonomy based on unsupervised clustering, followed by expert revision.

Sánchez and Moreno [37] have worked to build a concept knowledge base that is not based on hierarchical relationships. Their technique of Automated Ontology Learning allows a model to be built using non-taxonomic relationships. The two parts of their technique are the detection of relationships and the assignation of a semantic meaning to the relationship. Using WordNet, they have augmented the knowledge base with verified relationships.

Perrin and Petry [26] have applied concept analysis to psychiatric evaluation reports. They have defined a *collocational expression* as a relevant textual feature and provided an algorithm to extract them in an unsupervised environment.

Köhler et al. [15] have used existing semantic databases such as WordNet, SUMO and OpenCyc to improve search results for the World Wide Web. They have focused on word sense disambiguation techniques, which fall into these categories:

- Supervised analysis
- Unsupervised analysis
- Dictionary-based analysis
- Statistics-based analysis
- Combinations of the above methods

One of the techniques they have developed is called ontological indexing. This attempts to define the *context* of a *concept* by analyzing the relationships between the surrounding text and:

- synonyms of the concept
- *subconcepts* of the concept (more specific terms)
- *superconcepts* of the concept (more general terms)
- synonyms of the subconcepts of the concept
- synonyms of the superconcepts of the concept

To summarize, the WordNet database is an excellent tool that groups words into a hierarchy. We make extensive use of it in our concept analysis.

3.5 Authorship Analysis

Authorship analysis is an active area of research in the community. In this section, we present some of the techniques being developed. Much like a person leaves DNA traces behind, research shows that we also leave a "signature" in our writing styles. There are many developments in the field of e-mail analysis that seek to identify the author of an unsigned e-mail with impressive results.

De Vel et al. [6] have used document and language features to identify the gender and the language background of e-mail authors, via a Support Vector Machine (SVM) algorithm. They have classified authorship by male or female, and by English as a first language or English as a second language. With e-mails at least 200 words long their technique is able to achieve over 70 % accuracy.

Iqbal et al. [14] have worked in a related area to concept analysis, employing *writeprints* to help investigators with anonymous e-mails. They extract many document features to profile the author, such as:

- Lines in an e-mail
- Sentence count
- Paragraph count
- Presence/absence of greetings
- Has tab as separators between paragraphs
- Has blank line between paragraphs
- Presence/absence of separator between paragraphs
- Average paragraph length in terms of characters
- Average paragraph length in terms of words
- Average paragraph length in terms of sentences
- Use e-mail as signature
- Use telephone as signature
- Use URL as signature

To summarize, authorship identification is an important part of e-mail analysis. As it is outside the scope of our research, we discuss possible ways to integrate it as future work.

3.6 Measuring Success

This section examines some metrics used and developed by others in the field. Adopting these metrics will allow our results to be comparable with the existing body of research.

Several authors, including Yang [46], have used recall and precision to measure success in clustering:

$$\text{recall} = \frac{\text{correctly classified}}{\text{total classified}}$$

$$\text{precision} = \frac{\text{correctly classified in set}}{\text{total in set}}$$

Yang [46] and Tan [39] have used an F_1 measure of performance:

$$F_1 = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}}$$

Adami et al. [1] have defined two useful metrics, accuracy and coverage (analogous to recall and precision). The accuracy (A) is calculated as the number of correctly classified documents out of the number of classified documents, and the coverage (C) of a set is calculated as the number of correctly classified documents out of the number of documents that should be in the set:

$$A = \frac{\text{correctly classified}}{\text{total classified}}$$

$$C = \frac{\text{correctly classified in set}}{\text{total in set}}$$

Tan [39] has worked to improve the k-Nearest Neighbor (kNN) classification method. With the premise that kNN is biased because of a presumption of evenly distributed training data,

he has proposed a refinement called *DragPush*. Using an F_1 measure of performance, his DragPush classifier increases accuracy by several percent (to over 90 % in optimal cases).

$$F_1 = \frac{2 * \text{recall} * \text{precision}}{\text{recall} + \text{precision}}$$

Using the same presumption of inductive bias due to uneven training data distribution, Tan [40] has also refined the centroid text classifier. Using the same F_1 measure of performance, he has also increased accuracy by several percent. Tan has also researched the same improvements to the neighbour-weighted k-Nearest Neighbour classifier [38].

Hsiao and Chang [13] have proposed an adaptive learning classifier for separating legitimate and spam e-mails. Designed to run as a continuous process, they have compensated for the imbalance in the distribution of training data and for changes over time. They have measured accuracy using the following formula:

$$\text{accuracy} = \frac{A + D}{A + B + C + D}$$

where:

- A is the number of messages correctly classified as ham,
- B is the number of messages incorrectly classified as ham,
- C is the number of messages incorrectly rejected as spam, and
- D is the number of messages correctly rejected as spam.

Sinka and Corne [36] have provided the following algorithm for measuring the success of an unsupervised clustering algorithm in a trial dataset:

- Identify the category X that has the highest number of matches with the label L. Mark these matches as "correct".
- Mark as "incorrect" the documents with label L that are not in category X.
- Mark as "incorrect" the documents in category X that do not have label L.
- Mark label L as processed.
- Repeat until all labels are processed.

Iqbal et al. [14] have used the Enron e-mail corpus to evaluate their *writeprint* algorithm. They have randomly selected a number of employees (from three to ten), and for each employee they have randomly selected a number of that employee's e-mails (from 10 to 100). For 40 messages per author, they have achieved a success rate of 90 %.

To summarize, *recall*, *precision*, and F_1 are commonly employed measures of success. The Enron E-mail Dataset [29] is a publicly available corpus that is easily used for e-mail analysis testing.

3.7 Summary

This chapter has presented the key concepts in text mining and e-mail analysis that we use for our research, as shown in chapter 4.

Chapter 4

New Approach for E-mail Analysis

The goal of our approach is to improve existing techniques for e-mail analysis and to develop new ones. These techniques are tailored to the problem statement—how to help forensic investigators process a set of uncategorized e-mails. Our approach performs the following actions:

- Grouping related e-mails with high accuracy.
- Identifying the high-level concepts of each group.

Moreover, and to demonstrate this new approach, we develop a software tool with a graphical interface.

This chapter presents our contribution, which builds upon the state of the art described in chapter 3. It also provides the reader with essential information needed to understand our software implementation described in chapter 5.

This chapter is organized into five sections, representing five core areas of research: the e-mail analysis process, data extraction, pre-processing, clustering, and concept analysis. The extraction section shows how we convert e-mail messages in their original form into a workable form and how we parse the header information (including our *e-mail domain analysis* technique). The pre-processing section focuses on the initial textual processing of the message body, to prepare for clustering (including our *word pair analysis* technique). The clustering section shows how messages are reduced into *features* and represented as vectors to be clustered. Lastly, the concept analysis section describes how we use WordNet to identify the subjects of discussion in an e-mail message.

4.1 E-mail Analysis Process

After reviewing other proposals that have discussed the general text mining process (as it pertains to medical literature [2] and patent analysis [42]), we observe common trends:

- Getting the data (*extraction*)
- Parsing the data into a useful form (*preprocessing*)
- Performing complex analysis on the data (such as clustering or concept analysis)

By applying the existing body of research to our problem, we decide to use the following process for e-mail analysis:

1. Extraction
2. Pre-processing

3. Clustering

4. Concept analysis

There are some commonalities and differences between e-mails and other classes of documents that are important to discuss: structure and metadata, and length and vocabulary. These are examined in the following sections.

4.1.1 Structure and Metadata

As discussed in chapter 2, e-mails have two parts: a highly structured header, and a generally free-form body. Our data extraction techniques take into account routing information found in e-mail headers, e-mail addresses, and multipart messages. We design a completely new way of analyzing e-mail addresses using the Domain Name System (DNS), discussed later in this chapter.

4.1.2 Length and Vocabulary

Unlike other types of documents, e-mails are generally short and use a reduced vocabulary. We adopt and develop techniques (such as $tf * idf$ weighting, and word-pair analysis) to compensate for this difficulty.

4.2 Extraction

Extraction, or *data mining*, is the process of extracting useful information from a given source material. In our case, these are e-mail messages. The body of an e-mail contains

the bulk of the content, but by analyzing the header we can extract important data such as: email addresses, source and intermediary IP addresses, and mail user agent information. We use the open source toolkit mime4j [28] to extract e-mail content. This toolkit is an Apache James project that provides a rich environment for working with raw e-mail content.

Our principal innovation in this area is called *e-mail domain analysis*. We begin by analyzing the header of an e-mail message to harvest useful information about senders and recipients, as defined by the e-mail specification [7]. The header of an e-mail may contain sender addresses in the following fields:

- from
- reply-to
- resent-from
- sender

The header of an e-mail may also contain recipient addresses in the following fields:

- to
- cc
- bcc
- resent-to
- resent-cc
- resent-bcc

Each valid address is extracted and stored. We can now apply e-mail domain analysis to each one. This process replaces the domain part of the e-mail address with a new one that is based on the mail exchanger (MX) server registered in DNS. For example, when an e-mail is sent to a fictitious address "j_doe@encs.concordia.ca", the sending server performs a DNS query on "encs.concordia.ca" for all of the MX records. These indicate, with an order

of preference, the actual host name of a server that is able to receive e-mail for the domain. The sending server can then begin communication with the destination server to transmit the message for delivery.

In the case of "encs.concordia.ca", the first mail server has the DNS name "oldperseverance.encs.concordia.ca". Our technique would then convert the e-mail address "j_doe@encs.concordia.ca" to "j_doe@@oldperseverance.encs.concordia.ca". Note that this double-at-sign notation (user@@domain) is how we represent e-mail addresses that are converted using our domain analysis technique. This technique provides two advantages in certain cases:

- Domain analysis can eliminate duplicate addresses.
- Domain analysis can show links that are not previously visible.

The following sections discuss each of these advantages.

4.2.1 Eliminating Duplicate Addresses

Consider a fictitious Concordia student "John Doe" in the ENCS faculty whose user name is "j_doe". This person may have several equally valid aliases for the same mail drop:

- j_doe@encs.concordia.ca
- j_doe@cs.concordia.ca
- j_doe@ciise.concordia.ca

It is desirable for these e-mail addresses to be recognized as aliases of the same user. Through our e-mail domain analysis, we discover that the mail server for those three domains is the same: "oldperseverance.encs.concordia.ca". As such, the three different addresses will be converted to "j_doe@@oldperseverance.encs.concordia.ca". We are now able to see that it is, in fact, the same user.

4.2.2 Showing New Links

Consider two fictitious companies, "Acme General Contractors" and "Superlative Engineering & Design Inc.", with the respective e-mail domains "acme.com" and "superdesign.com". In this example, the two corporations are actually owned and operated by the same people. In fact, all of their information technology infrastructure is shared. In a fictitious DNS query, the mail exchanger for both "acme.com' and "superdesign.com" is "mail2.acme.net". Our e-mail domain analysis technique reveals all e-mail addresses belonging to either "Acme General Contractors" or "Superlative Engineering & Design Inc." to be part of the same organization. This scenario is presented in Figure 7.

4.3 Pre-processing

A list of common words to exclude (*stop words*), and *stemming* words to their lexical root are often-used techniques in data mining [30, 36]. Another important technique is inverse weighting (such as $tf * idf$, as described in chapter 2). The theory behind this weighting is that the more *often* a word appears, the less important it is.

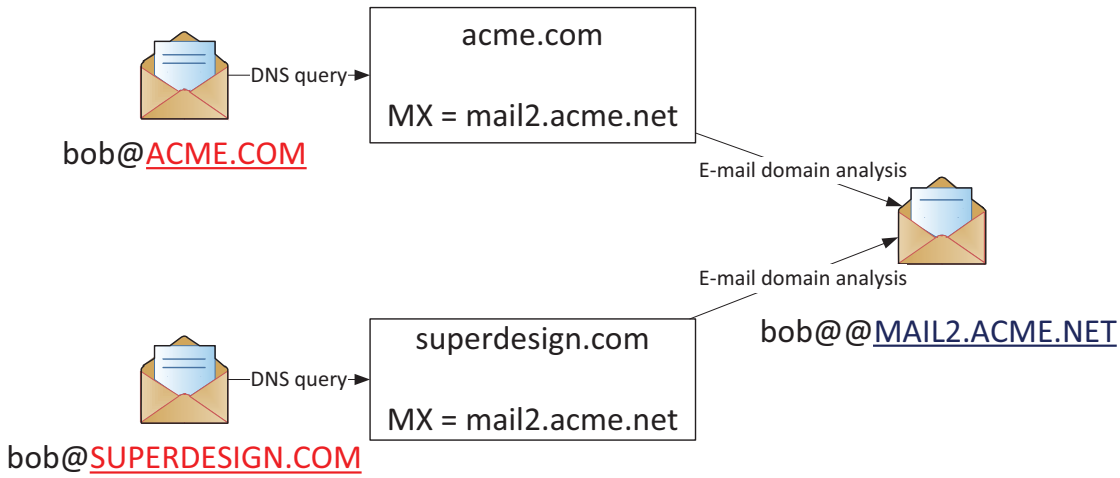

Figure 7: E-mail Domain Analysis

We use several methods to help reduce raw e-mails to a form that is relevant for further analysis (*pre-processing*):

- Extraction of the textual body parts of an e-mail message;
- Removing the signature block of a message (that is delineated with two hyphens);
- Removing short lines at the beginning and the end of a message;
- Identifying each word's English language part of speech, keeping only common nouns and verbs;
- Stemming each word to its root form if possible;
- Analyzing word pairs to increase accuracy.

In the following sections we elaborate on each of these methods.

4.3.1 Part-of-speech Tagging

A prerequisite for our natural language analysis is to determine the function of each word in a sentence. Consider the two sentences:

"Are you sitting on the chair?"

"Please chair the meeting."

Both of these sentences use the word "chair", but the uses are entirely unrelated. The first "chair" is a noun, while the second "chair" is a verb. Applying part-of-speech (POS) tagging allows our software to know if the word "chair" is a noun or a verb.

The Penn Treebank [18] is a database for classifying parts of speech into various categories. We determine that common nouns and verbs provide a good foundation for concept analysis. These are represented by the following Penn Treebank tags [17]:

- *NN*: noun, singular or mass
- *NNS*: noun, plural
- *VB*: verb, base form
- *VBD*: verb, past tense
- *VBG*: verb, gerund/present participle
- *VBN*: verb, past participle
- *VBP*: verb, non-third person singular present
- *VBZ*: verb, third person singular present

We reduce this list of classifications to either “noun” or “verb” for our analysis. By excluding other words, it is easy to filter less relevant content and leave only the most important parts of the message. As an example, consider the following sentence:

“The quick brown fox jumps over the lazy dog.”

Extracting only common nouns and verbs yields the abridged version below:

{ fox, jumps, dog }

4.3.2 Stemming

Once the most relevant words have been extracted, it is desirable to reduce each word to a base form. Consider the following two sentences:

“The quick brown fox jumps over the lazy dog.”

“The quick brown foxes jump over the lazy dogs.”

Intuitively, these sentences describe similar situations; however, using only the part-of-speech tagging gives us the following sets:

{ fox, jumps, dog }

{ foxes, jump, dogs }

To a computer, these two sets are completely different. Using the Java WordNet Interface [34] we are able to stem the words to a root form. Using this algorithm, the computer can identify the equivalence:

- foxes → fox
- jumps → jump
- dogs → dog

This helps us cluster the e-mail messages by reducing the number of distinct words that need to be analyzed.

4.3.3 Word Pair Analysis

A common method for natural language processing is *bigram analysis*. This technique considers letter pairs as analysis units. It is also a successful way to reverse simple letter-swapping cryptography, as the statistical occurrence of each bigram pair is generally known (or can be computed) for the source language. Building on this principle, we develop a *word pair* analysis technique. We focus on word pairs instead of individual words. By filtering out word pairs that occur only once, we are left with a reduced, higher quality set of terms for analysis. This is confirmed by our experimental observations.

4.3.4 Example

Consider the following paragraph from an e-mail in the Enron dataset: [29]

> "This is an alert that FERC will be more critical in the future of negotiated rate deals. The orders discussed below dealt with buyout provisions in negotiated rate deals that Tennessee, ANR and Columbia had done. These provisions had previously been permitted as negotiated rates, but FERC has decided that there

is significant potential for undue discrimination and the recourse rates do not provide an adequate alternative. Thus, anything in a negotiated rate deal that is not strictly rate-related will most likely be subject to intense scrutiny."

This reduces to the following set of common nouns and verbs:

{is, alert, be, future, negotiated, rate, deals, orders, discussed, dealt, buyout, provisions, negotiated, rate, deals, had, done, provisions, had, been, permitted, negotiated, rates, decided, is, potential, discrimination, recourse, rates, do, provide, alternative, anything, negotiated, rate, deal, is, be, scrutiny}

Stemming each word to its root form provides the following set of stemmed common nouns and verbs:

{be, alert, be, future, negotiate, rate, deal, order, discuss, deal, buyout, provision, negotiate, rate, deal, have, do, provision, have, be, permit, negotiate, rate, decide, be, potential, discrimination, recourse, rate, do, provide, alternative, anything, negotiate, rate, deal, be, be, scrutiny}

Using word pair analysis, we have the following set:

- {negotiate, rate} **(4 times)**
- {rate, deal} **(3 times)**
- {be, alert}
- {alert, be}
- {be, future}
- {future, negotiate}
- {deal, order}
- {order, discuss}

- {discuss, deal}
- {deal, buyout}
- {buyout, provision}
- {provision, negotiate}
- {deal, have}
- {have, do}
- {do, provision}
- {provision, have}
- {have, be}
- {be, permit}
- {permit, negotiate}
- {rate, decide}
- {decide, be}
- {be, potential}
- {potential, discrimination}
- {discrimination, recourse}
- {recourse, rate}
- {rate, do}
- {do, provide}
- {provide, alternative}
- {alternative, anything}
- {anything, negotiate}
- {deal, be}
- {be, be}
- {be, scrutiny}

Lastly, we eliminate all word pairs that occur only once, leaving:

- negotiate, rate (4 times)
- rate, deal (3 times)

The entire text has been reduced to two features for clustering. By using word pair analysis over a collection of e-mail messages, popular combinations become more frequent

and are therefore retained.

It is possible that some messages in the collection will not have any word pairs that are retained. We consider this a symptom of a message that is unlikely to be accurately clustered due to insufficient content. In our analysis of 1082 messages, we are able to cluster 58 additional messages by not removing singleton word pairs.

4.4 Clustering

Clustering is the process of aggregating distinct e-mails into groups. Much as a collection of papers remains just that until they are organized into meaningful piles or folders, a collection of e-mails is no different. Presented with five, ten, or twenty messages, an investigator can read each one and group them appropriately; however, an investigator may be faced with fifty e-mails, one hundred e-mails, or more. Computer software can provide valuable assistance to an investigator. There are two fundamental groups of techniques used to cluster messages: supervised, and unsupervised.

Supervised learning requires a training set consisting of e-mails (or more generally, instances) that have already been clustered. The test set is clustered based on the existing knowledge gained from the training set. Supervised learning can be quite accurate, depending on the quality of the training set. Unfortunately, in a real-life environment, a training set is not always available; therefore, the application of supervised learning techniques is more limited. Unsupervised learning techniques do not use a training set, but are also less accurate. The primary advantage to using unsupervised learning is that no training set is

Table 2: Example of Message Features

Message	Address 1	Address 2	Word Pair 1	Word Pair 2
1	1	1	0	2
2	1	1	0	1
3	0	1	1	0
4	0	1	3	2
5	1	0	0	4
6	1	0	0	1
7	0	1	2	0
8	1	1	0	3
9	0	1	4	0
10	1	0	0	2

required.

Once the text of an e-mail has been mined for useful word pairs it is time to group e-mails together based on their content and metadata. Building upon the general techniques of text mining, we use other parts of the e-mail message to increase the accuracy of the clustering techniques beyond what can be achieved by analyzing the body content alone. The clustering of e-mails is accomplished in the following three steps:

1. E-mail address extraction and domain analysis.

2. Word pair analysis for each e-mail message.

3. Clustering of e-mails based on the e-mail addresses and word pair features.

Each e-mail message is reduced to a set, or vector, of *features*. The features that are used as input to the clustering algorithm include the word pairs that occur in the message, e-mail addresses that occur in the header, and e-mail domains that occur in the header. The collection of messages and features can be visualized as a matrix, as shown in Table 2.

It is this generic matrix of features that is used to cluster the messages. In the case of our implementation, we use the expectation-maximization algorithm [25], implemented by Weka. This algorithm allows each message to belong to exactly one cluster.

This example shows how an e-mail message is reduced to word pair features, e-mail address features, and domain features for analysis. Consider this example message:

```
From: victor.lamadrid@enron.com
To: robert.allwein@enron.com, airam.arteaga@enron.com, sunjay.arya@enron.com,
shanna.boudreaux@enron.com, tamara.carter@enron.com,
joann.collins@enron.com, stephanie.erwin@enron.com,
clarissa.garcia@enron.com, steve.gillespie@enron.com,
lia.halstead@enron.com, meredith.homco@enron.com,
kelly.loocke@enron.com, chris.ordway@enron.com,
michael.pritchard@enron.com, daniel.prudenti@enron.com,
robert.ramirez@enron.com, christina.sanchez@enron.com,
tracy.wood@enron.com
Subject: FW: New FERC Policy on Negotiated Rate Deals
Cc: chuck.ames@enron.com, f..brawner@enron.com, chris.germany@enron.com,
scott.goodell@enron.com, john.hodge@enron.com, f..keavey@enron.com,
brad.mckay@enron.com, jonathan.mckay@enron.com, scott.neal@enron.com,
w..pereira@enron.com, vladi.pimenov@enron.com, andrea.ring@enron.com,
leonidas.savvas@enron.com, maureen.smith@enron.com,
craig.taylor@enron.com, judy.townsend@enron.com,
victoria.versen@enron.com, frank.vickers@enron.com

fyi...

This is an alert that FERC will be more critical in the future of negotiated rate deals.
The orders discussed below dealt with buyout provisions in negotiated rate deals that
Tennessee, ANR and Columbia had done.  These provisions had previously been permitted
as negotiated rates, but FERC has decided that there is "significant potential for undue
discrimination" and the recourse rates do not provide an adequate alternative. Thus,
anything in a negotiated rate deal that is not strictly rate-related will most likely
be subject to intense scrutiny.
```

As shown in the previous section, the following word pair features are extracted from this message (remember that word pairs that occur less than twice are not included):

- negotiate, rate (4 times)
- rate, deal (3 times)

The following e-mail addresses and domains are extracted from this message (all e-mail domains have been converted to the MX host name for that domain):

- victor.lamadrid@enron.com
- robert.allwein@enron.com
- airam.arteaga@enron.com
- sunjay.arya@enron.com
- shanna.boudreaux@enron.com
- tamara.carter@enron.com
- joann.collins@enron.com
- stephanie.erwin@enron.com
- clarissa.garcia@enron.com
- steve.gillespie@enron.com
- lia.halstead@enron.com
- meredith.homco@enron.com
- kelly.loocke@enron.com
- chris.ordway@enron.com
- michael.pritchard@enron.com
- daniel.prudenti@enron.com
- robert.ramirez@enron.com
- christina.sanchez@enron.com
- tracy.wood@enron.com
- chuck.ames@enron.com
- f..brawner@enron.com
- chris.germany@enron.com
- scott.goodell@enron.com
- john.hodge@enron.com
- f..keavey@enron.com
- brad.mckay@enron.com
- jonathan.mckay@enron.com
- scott.neal@enron.com
- w..pereira@enron.com
- vladi.pimenov@enron.com
- andrea.ring@enron.com
- leonidas.savvas@enron.com
- maureen.smith@enron.com
- craig.taylor@enron.com

- judy.townsend@enron.com
- frank.vickers@enron.com
- victoria.versen@enron.com
- @@inbound.mailwise.com

This combined list of features e-mail addresses, domain parts, and word pairs, will be the input to the Weka clustering algorithm.

4.5 Concept Analysis

Automated concept analysis is an incredible and complex goal for investigators. It aims to extract the semantic meaning of a document with little or no human intuition. In a traditional physical investigation, an investigator (or a team of investigators) would need to manually sift through documents and identify concepts, noting which documents are relevant and which are not. This is also possible with digital evidence—but not entirely necessary. This burden can be taken off of the human investigators through automated analysis techniques.

We have already explored the preprocessing and clustering of e-mail messages. At this point an investigator has e-mails separated into logical groups. This allows a better division of labour (each group can be analyzed together, focusing the investigator's work on related messages at the same time). This section shows how we analyze each group as a whole and how we infer the concepts (topics) discussed.

The WordNet Project [43] is a hierarchical semantic database of relationships between words. To make an investigator's task easier we use WordNet to discover the concepts discussed in a collection of e-mails. This public database gives our software access to

a rich hierarchy of lexical relationships. WordNet provides definitions of words and the relationship between words. An example of how WordNet's hierarchy can be visualized is shown in Figure 8.

The basic unit of WordNet is the *synset*, which is a set of words sharing the same meaning. Words that have several meanings will appear in several synsets. In general, synsets are organized in a hierarchy as follows:

- more general concepts
- more specific concepts
- similar concepts

The analysis techniques that we use focus on nouns and verbs. The terms that apply are described in more detail below.

Nouns The following relations for noun synsets are relevant to our analysis:

- *hypernym*—a more general term. For example, *furniture* is a hypernym of *table*. A table is, by definition, a piece of furniture.
- *hyponym*—a more specific term. For example, *pen* is a hyponym of *writing implement*. A pen is, by definition, a writing implement.
- *sister terms*—terms sharing a common hypernym. For example, *pen* and *chalk* are sister terms because they share the hypernym *writing implement*.

All nouns are hyponyms of *entity*, the "root" noun. WordNet calls this the *unique beginner*.

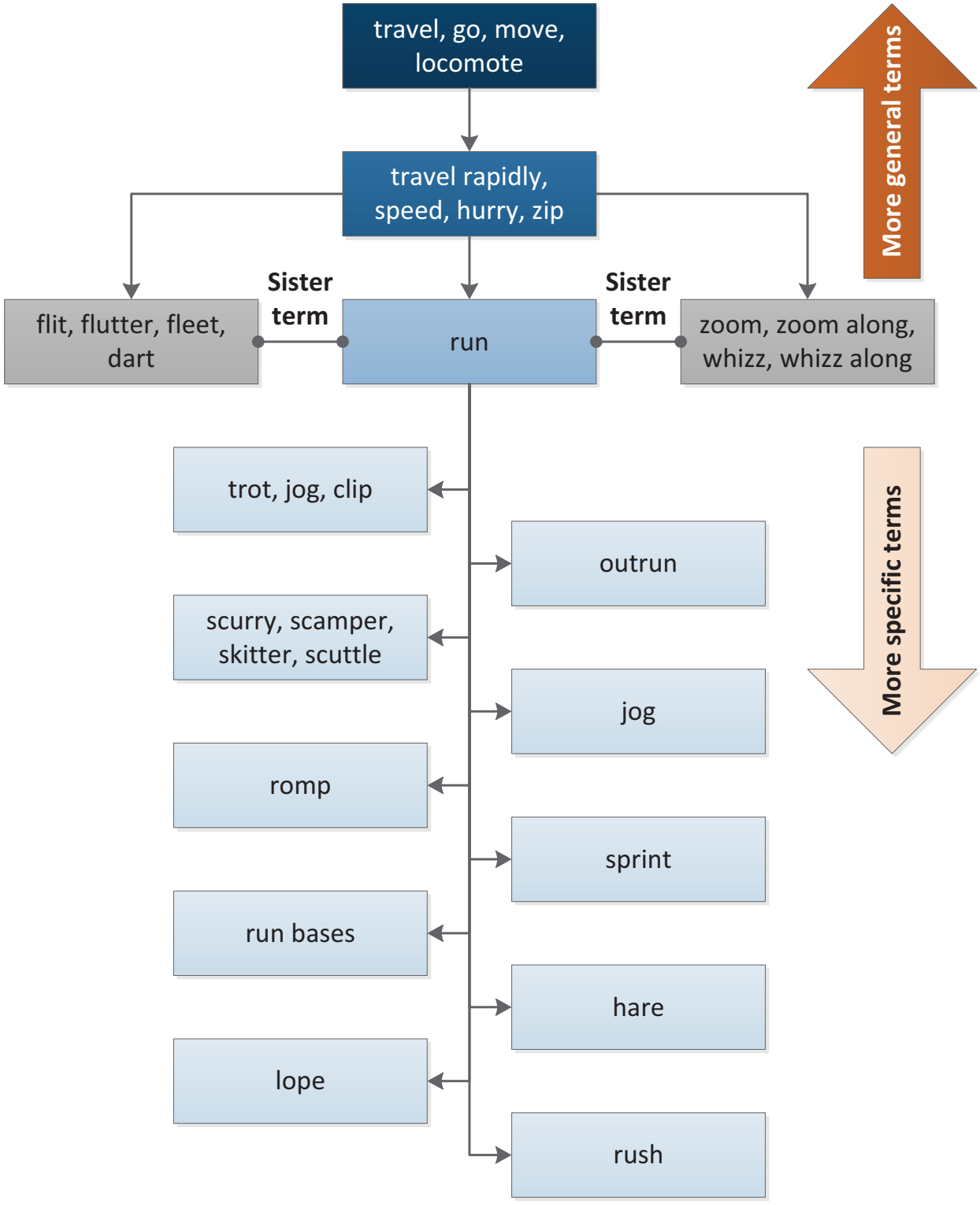

Figure 8: WordNet Relationship Example

Verbs The following relations for verb synsets are relevant to our analysis:

- *hypernym*—a more general term. For example, *travel rapidly* is a hypernym of *run*. Someone who is running is, by definition, traveling rapidly.

- *troponym* (or *hyponym*)—a more specific term. For example, *jog* is a troponym of *run*. Someone who is jogging is, by definition, running.

- *sister terms*—terms sharing a common hypernym. For example, *jog* and *scurry* are sister terms because they share the hypernym run.

The central feature of our concept analysis technique is the identification of common hypernyms. We hypothesize that as the number of words sharing a common hypernym increases, so does the likelihood that this hypernym is an important concept in the text being analyzed. Consider the example from Figure 8: if the words *jog*, *trot*, and *outrun* all appear in a document, it is logical to conclude that the document is about running, because *run* is a hypernym of each of those words. In the following pages we will explain the challenges of using WordNet for this purpose and our solutions.

A major problem to overcome with this technique is that one word may have many different semantic meanings. Consider a WordNet query for the word *jog*. No less than nine synsets are returned as a result:

- *jog (n)*—a sharp change in direction (e.g., "there was a jog in the road")

- *jog (n)*—a slow pace of running

- *jog (n)*—a slight push or shake

- *jog (v)*—continue talking or writing in a desultory manner (e.g., "this novel rambles on and jogs")
- *jog (v)*—even up the edges in a stack of paper, in printing
- *jog (v)*—run for exercise (e.g., "jog along the canal")
- *jog (v)*—run at a moderately swift pace
- *jog (v)*—give a slight push to
- *jog (v)*—stimulate to remember (e.g., "jog my memory")

Given only the word jog, with no other contextual information, it is impossible to know which of these nine meanings to use. Our solution to this problem is twofold:

- Part-of-speech tagging and common hypernym reduction
- Popularity

To illustrate our solutions, consider the following phrase:

"I am jogging with you. Don't trot, or I will outrun you."

4.5.1 Part-of-speech Tagging and Hypernym Reduction

First, the *Stanford Log-linear Part-of-speech Tagger* [11] used in the e-mail preprocessing provides us with the additional data of the type of word (i.e., noun or verb). This will eliminate some incorrect possibilities. Consider the word *jogging* from the example in the

previous section. Through stemming, this will be reduced to *jog*. Although the word *jog* has nine synset results, only six of them are verbs. Of those six, we identify their hypernyms:

- *jog (v)*—continue talking or writing in a desultory manner (e.g., "this novel rambles on and jogs") → *continue (v)*
- *jog (v)*—even up the edges in a stack of paper, in printing → *square (v)*
- *jog (v)*—run for exercise (e.g., "jog along the canal") → *run (v)*
- *jog (v)*—run at a moderately swift pace → *run (v)*
- *jog (v)*—give a slight push to → *nudge (v)*
- *jog (v)*—stimulate to remember (e.g., "jog my memory") → *provoke (v)*

The six results for *jog* have five unique hypernyms:

- *continue (v)*
- *square (v)*
- *run (v)*
- *nudge (v)*
- *provoke (v)*

4.5.2 Popularity

Continuing the *jog* example, we hypothesize that important concepts will have many hyponyms in the body text; therefore, we simply consider all of the synset results for the document. We expect that a "popular" hypernym will have many "hits" from different words.

Now consider the hypernyms of trot and outrun, from Figure 8:

- *jog (v)*—run at a moderately swift pace → *run (v)*
- *trot (v)*—ride at a trot → *ride horseback (v)*
- *trot (v)*—cause to trot (e.g., "she trotted the horse home") → *walk (v)*
- *outrun (v)*—run faster than (e.g., "in this race, I managed to outrun everybody else") → *run (v)*

The three verbs *jog*, *trot*, and *outrun* have a combined total of ten different meanings; however, they only have seven different hypernyms among them:

- *continue (v)*—1 occurence
- *square (v)*—1 occurence
- ***run (v)*—4 occurences**
- *nudge (v)*—1 occurence
- *provoke (v)*—1 occurence

- *ride horseback (v)*—1 occurence
- *walk (v)*—1 occurence

Of those seven different hypernyms, the verb *run* appears most often (by far). It is this logic that we use to deduce the phrase is about running.

4.6 Summary

We have retained several techniques from the body of existing research, such as the e-mail analysis process, the $tf * idf$ method of inverse weighting, part-of-speech tagging, and stemming. To this, we have added three new techniques: e-mail domain analysis, word pair analysis, and concept analysis with WordNet.

Chapter 5

Design and Implementation

This chapter shows how we construct a proof-of-concept application in Java that implements our research and demonstrates our clustering and concept analysis techniques. The software includes a graphical user interface; however, for the purpose of this proof-of-concept, some of our research (especially testing and verification algorithms) is implemented only in a console application. This chapter describes the core implementation of our research, and showcases the graphical user interface as a final section as an example of our vision of what sort of tool might be useful to investigators.

5.1 Concept Analysis Test Bench

The Concept Analysis Test Bench (CATB) is the program that applies our software algorithms to a collection of e-mails and outputs the results in a manner suitable for verification. It is a console application with no user interface.

The CATB performs several distinct operations, in sequence, to process the input e-mails and arrive at a result. The first set of operations are performed on each message individually. The second set of operations are performed on the collection of all messages. Finally, several verification steps are performed to measure the results. The individual message processing operations are:

1. Remove the opening and the closing lines of a message.
2. Tag each word in the message with its English-language part of speech (common nouns and verbs only).
3. Stem each word to its root.
4. Count the number of occurrences of each word pair.

Once each message has been processed individually, the collective message processing operations are:

1. Remove all word pairs that occur only once.
2. Remove any messages that no longer have any word pairs.
3. Build a Weka instance for each message.
4. Cluster the messages.
5. Verify the clusters.

Lastly, it performs concept analysis on each cluster.

5.2 Loading the E-mails

We use a singleton class *SlimEmailContainer* as a lightweight container for all of the e-mails to be analyzed. The SlimEmailContainer uses a Java *TreeMap* class that contains a collection of *EmailAnalysisObject* classes. This is the fundamental class that models an RFC 2822 e-mail message. We create a class *Scavenger* that converts a raw message into an EmailAnalysisObject.

Inside the EmailAnalysisObject, the mime4j library is used to parse each message [28]. If there are multiple textual parts (i.e., the message is MIME-encoded), they are joined together by concatenation. Only textual parts (MIME type "text/plain") are recognized. Other parts of messages, such as HTML-encoded parts (MIME type "text/html"), are discarded. A future version might handle these other parts; however, this is not an urgent feature since e-mail programs conventionally send a plain text part with an HTML part to support clients that do not display HTML. The e-mails are loaded from a hard-coded path. The subdirectory relative to the base path is kept as a reference for later verification. Algorithm 1 is used to parse message content.

5.3 Individual Message Processing

For each message in the newly-created SlimEmailContainer, a corresponding *ConceptEmail-Params* class is generated. This class stores information such as its unique ID number, the e-mail addresses in the message, the words in the message, and the cluster the message exists in (after clustering).

Algorithm 1 Scavenge Text From MIME Parts

```
Get message body in body
if body is an embedded message (e.g. forward) then
    Call this function recursively on the embedded message and return its output
else if body is of type Multipart (e.g. MIME) then
    for all parts of body do
        Call this function recursively for the current part and append the result to the output
    end for
else if body is of type Text (e.g. plain-text message or plain-text part of a MIME message)
then
    Return the textual parts of the message as a single string
else
    Return an empty string
end if
```

5.3.1 Remove Opening and Closing Lines

After extracting the plain text portions of the e-mail message, we attempt to remove certain portions of it that are not relevant to content analysis. These portions are:

- Signature block at the end
- Opening greetings, such as "Hi Mr. Jones,"
- Closing remarks, such as "Sincerely, / John Smith"

The *RemoveShortEnds* function of EmailAnalysisObject is used to remove lines with less than five words at the beginning (first line only) and at the end of a message (last two lines). This removes common phrases such as "Hello," and "Goodbye" that do not aid us in message clustering. The behaviour can be modified using the constants MAX_WORDS_-CUTOFF (default 5), MAX_LINES_TOP (default 1), MAX_LINES_BOTTOM (default 2). This is illustrated in Algorithm 2.

Algorithm 2 Removing Short Lines

MAX_WORDS_CUTOFF = 5
MAX_LINES_TOP = 1
MAX_LINES_BOTTOM = 2
top_marker = index of first line in message
$bottom\_marker$ = index of last line in message

Get message content in $content$
for first MAX_LINES_TOP lines in of $content$ **do**
  if $line$ has more words than MAX_WORDS_CUTOFF **then**
    Break from loop
  else
    Increment top_marker
  end if
end for
for last MAX_LINES_BOTTOM lines in of $content$, in reverse order **do**
  if $line$ has more words than MAX_WORDS_CUTOFF **then**
    Break from loop
  else
    Decrement $bottom\_marker$
  end if
end for
return All text between top_marker and $bottom\_marker$

The *RemoveSignature* function of EmailAnalysisObject (Algorithm 3) is used to remove all lines after a line containing only "--" (defined by the constant MARKER). This line is commonly used to begin the signature block at the end of a message.

Algorithm 3 Removing the Signature Block

```
MARKER = "--"

Get message content in content
for all lines in of content do
   if line begins with MARKER then
      return output
      end
   else
      Add line to output
   end if
end for
return output
```

5.3.2 Part-of-speech Tagging and Word Pair Computation

Part-of-speech (POS) tagging is accomplished using the Stanford Natural Language Processing Group's Log-linear software [11]. This software implements a Maximum Entropy POS tagger in Java with a rich API. The Stanford tagger also provides a WordToSentenceProcessor class useful for parsing a document. This class allows a document to be converted into an array of *Sentences* (also defined by the Stanford tagger). The Sentences can be directly used with the tagger class (MaxentTagger) to produce a list of TaggedWords (a class that contains a Word and its associated Tag). The tags generated conform to the format of the Penn Treebank [18].

Sentence by sentence, we now compute *word pairs* from the common nouns and verbs.

Each word is also stemmed to its root form using the *WordnetStemmer* [43], through the *SlimConceptAnalyzer* class we create, as shown in Algorithm 4.

Algorithm 4 Part-of-Speech Tagging and Word Pair Computation

```
Parse the document into sentences using the MaxentTagger
for all sentences do
  for all taggedwords in sentence do
    if word contains [: / @ '] then
      do nothing
    else if word.PartOfSpeech in [NOUN, VERB] then
      Stem word as stemmed_word
      if this is the first word in the sentence then
        do nothing
      else
        add the pair of the preceding word and this one to the wordpairlist
      end if
    end if
  end for
end for
```

We create two classes for the word pair computation: *WordPOS* which pairs a word with a part-of-speech tag, and *WordPOSPair* which implements a pairing of two WordPOS objects. Both classes implement a comparator. We also create a generic class *CountingMap* that extends the TreeMap class and provides for each object to be paired with an integer. When adding an object to the CountingMap, if an equal object already exists, its count is automatically increased by one. The CountingMap, unique to each e-mail, is kept in an array. Additionally, the entire CountingMap is added to a master CountingMap for all e-mails. This master CountingMap lets us later determine how frequently a word pair is across all messages.

5.4 Collective Message Processing

We now have a collection of word pairs that represent the essence of each message. These next operations operate on groups of messages, rather than individual ones.

5.4.1 Remove Singleton Word Pairs

Using the master CountingMap that is created during the word pair computation, we remove any word pairs that occur only one time. This can reduce the number of word pairs by an order of magnitude, depending on the message. Since the word pairs occurred only once, we judge that they will not aid in clustering. Any message that has no word pairs remaining (because it does not occur in any other message) will be considered non-clusterable.

5.4.2 Build Weka Instances

We need to construct a Weka *instance* for each message to be clustered. Our *InstanceBuilder* class will help by wrapping this part of Weka. The collection of Weka instances is simply a two-dimensional matrix. Each row represents an instance (an e-mail) and each column represents an attribute, or feature, of the e-mail. For each message, the attribute column is set to 1 if present or 0 otherwise. The columns of *attributes* represent:

- Each word pair in the collection of messages.
- All of the e-mail addresses in the message header.

5.4.3 Message Clustering

As mentioned in the previous chapter, we opt to integrate Weka rather than re-implement a clustering algorithm. Using the EM *classifier* (clustering algorithm) provided by Weka, we can group similar messages. Each e-mail is represented as a Weka instance. The analysis features in each instance are called attributes. The Weka API is used to cluster the instances using the clusterer of our choice. We use the expectation-maximization algorithm, which assigns a probability distribution to each e-mail that indicates the probability of it belonging to each of the clusters. The number of clusters is chosen by cross-validation [12, 45]. The features used for clustering are the word pairs extracted from the body and the addresses extracted from the header (after e-mail domain analysis).

5.5 Verification of Clusters

For research purposes, we use e-mails that are already grouped into folders (and presumed related). Consider the groups $G_1, G_2, \ldots, G_n$. Each processed message $m_1, m_2, \ldots, m_n$ belongs to exactly one group. After the clustering process, each message also belongs to exactly one cluster, $C_1, C_2, \ldots, C_n$. For each cluster C_i, we identify the source group G for each message m (or $G(m)$) in C_i. The source group having the most messages in C_i is determined to be the "correct group" for the cluster C_i, or $G(C_i)$, as detailed in Algorithm 5.

An e-mail is considered correctly clustered if its group $G(m)$ is equal to its cluster's correct group, $G(C(m))$. Otherwise, it is considered incorrectly clustered. This is calculated for each source group (Algorithm 6), each cluster (Algorithm 7), and for all messages

together (Algorithm 8).

Algorithm 5 Generation of Cluster Identities

Require: Clustered messages
  for all messages m_i in $clusteredmessages$ **do**
    $C = cluster(m_i)$
    $G = sourcegroup(m_i)$
    $C(G) + +$
  end for
  for all C_i in $clusters$ **do**
    for all G_i in $groups$ **do**
      if $C_i(G_i)$ is the highest **then**
        Name the cluster C_i with the name of G_i and an incremental counter
      end if
    end for
  end for

Algorithm 6 Verification of Groups

Require: Clustered messages and cluster identities
  for all G_i in $sourcegroups$ **do**
    for all messages m_i in group G_i **do**
      if m_i is not clustered **then**
        $G_i.unclustered + +$
      else if $C(m_i) = G_i$ **then**
        $G_i.correct + +$
      else
        $G_i.incorrect + +$
      end if
    end for
  end for

5.6 Concept Analysis

Now that the e-mails are clustered, we use the WordNet database to examine the individual words in each message and infer some of the semantic concepts in each cluster. This is accomplished in the following steps:

Algorithm 7 Verification of Clusters

Require: Clustered messages and cluster identities
  for all C_i in $clusters$ **do**
    for all messages m_i in cluster C_i **do**
      if $G(m_i) = C_i$ **then**
        $C_i.correct++$
      else
        $C_i.incorrect++$
      end if
    end for
  end for

Algorithm 8 Verification of All Messages

Require: Clustered messages and cluster identities
  for all m_i in $messages$ **do**
    if m_i is not clustered **then**
      $unclustered++$
    else if $G(m_i) = C(m_i)$ **then**
      $correct++$
    else
      $incorrect++$
    end if
  end for

1. Generate a list of hypernyms for each cluster.

2. Use a $tf * idf$ analysis to determine which hypernyms are relevant.

5.6.1 Generation of Hypernym Lists

We begin by creating a CountingMap for all of the WordPOS objects in each cluster. This will allow us to count the nouns and verbs in each cluster. For each WordPOS object in the entire cluster, we use *WordNet* to identify its *synset*. This is accomplished using the *GetSynset* function of our *SlimConceptAnalyzer* class. The synset identification begins by stemming the word with the WordNet stemmer (different than the one used previously in the clustering process). This stemmer can return multiple words, depending on the input value. Each value is checked in the WordNet database to see if it matches the part-of-speech tag that has been assigned during the clustering process. If the word is a match, it is added to a list of synsets. Generally there is no more than a single match.

As described in Algorithm 9, we look up the word's *hypernym* using WordNet and store it in a CountingMap of word-hypernym pairs. We eliminate all hypernyms that occur 1 or 2 times, keeping those that occur 3 times or more.

5.6.2 Determination of Relevant Hypernyms

Using our *TfidfAnalyzer* class, we store the ranked hypernyms of each cluster. Our class provides a function *GenerateWeights* that implements the $tf * idf$ algorithm, as shown in Algorithm 10. We can now rank each term (hypernym) in the cluster and output an ordered

list (along with the $tf * idf$ value).

5.7 Graphical Interface

The graphical component of our software is a proof-of-concept showing how our algorithms and techniques can be applied in an actual software product. It allows a user to drag-and-drop source message files into the interface window (Figure 9). These are parsed and then displayed following the on-disk directory structure (in the case of multiple messages in individual files), or the internal folder structure of a mailbox file (such as the .mbox format). The user is also able to click on messages to see them rendered for easy viewing and navigation (Figure 10).

Using a publicly available IP to location database, and Google Maps, our software is able to generate a map based on the *Received* fields in an e-mail header, as shown in Figure 11.

Additionally, the software is able to analyze certain social networking parameters and display a graph containing a multitude of information. Figure 12 shows each e-mail address represented as a single node. Nodes that are associated with more messages are larger. The directed lines connecting each node shows the relative amount of communication in that direction (number of messages sent).

Algorithm 9 Generating a Hypernym Set

Require: Cluster of messages
  $ClusterWordList$ = new CountingMap
  for all m_i in $messages$ **do**
    Add the words of m_i to $CusterWordList$, with their Part-of-Speech tag (POS)
  end for
  $WordToHypernym$ = new CountingMap
  for all $word_i, POS(word_i)$ in $ClusterWordList$ **do**
    Get the WordNet synset for $word_i, POS(word_i)$ as $synsetID$
    Get the hypernym of $synsetID$ as $hypernymID = hypernym(word_i), word_i$
    Increment $WordToHyperym$ with $hypernym(word_i), word_i$
  end for
  Remove all elements of $WordToHypernym$ that occur less than 3 times

Algorithm 10 Generating Inverse Weights ($tf * idf$ analysis)

Require: Array of lists of ranked objects
  / Generate the document frequency for each object */*
  $DocumentFrequency$ = new TreeMap
  for all lists of ranked objects L_i in array A **do**
    for all object O_j in list L_i **do**
      increment $DocumentFrequency(O_j)$ by value $Rank(O_j)$
    end for
  end for
  / Compute the inverse document frequency (idf) */*
  $InverseDocumentFrequency$ = new TreeMap
  for all objects O_i in $DocumentFrequency$ **do**
    $InverseDocumentFrequency(O_i) = \frac{log_10(count(A)}{Rank(O_i)}$
  end for
  Store a reference to $InverseDocumentFrequency$ with each list L in array A for computing the $tf * idf$ value
  / The $tf * idf$ value is simply the term frequency in a particular document multiplied by that term's inverse document frequency */*

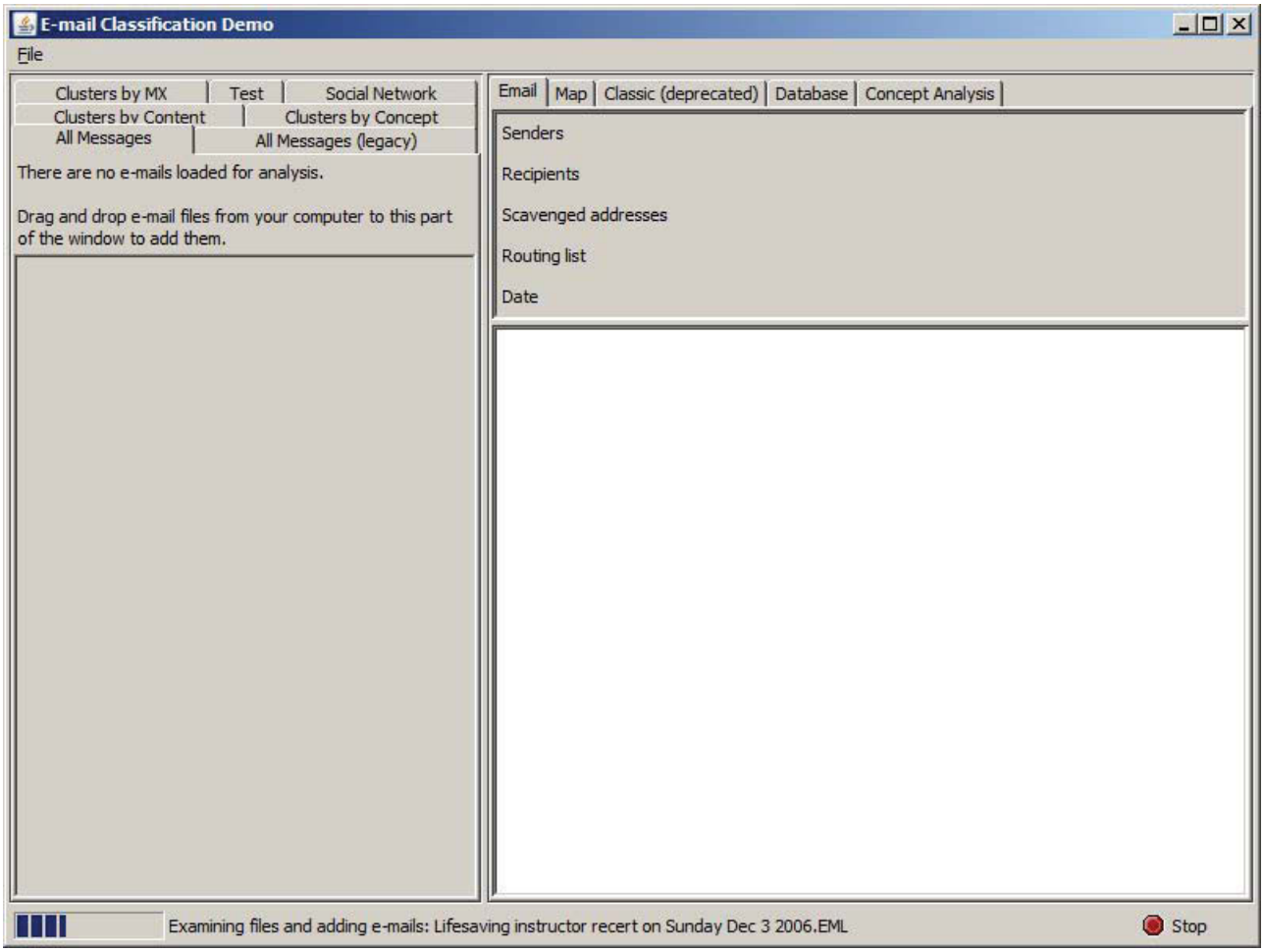

Figure 9: Adding E-mails

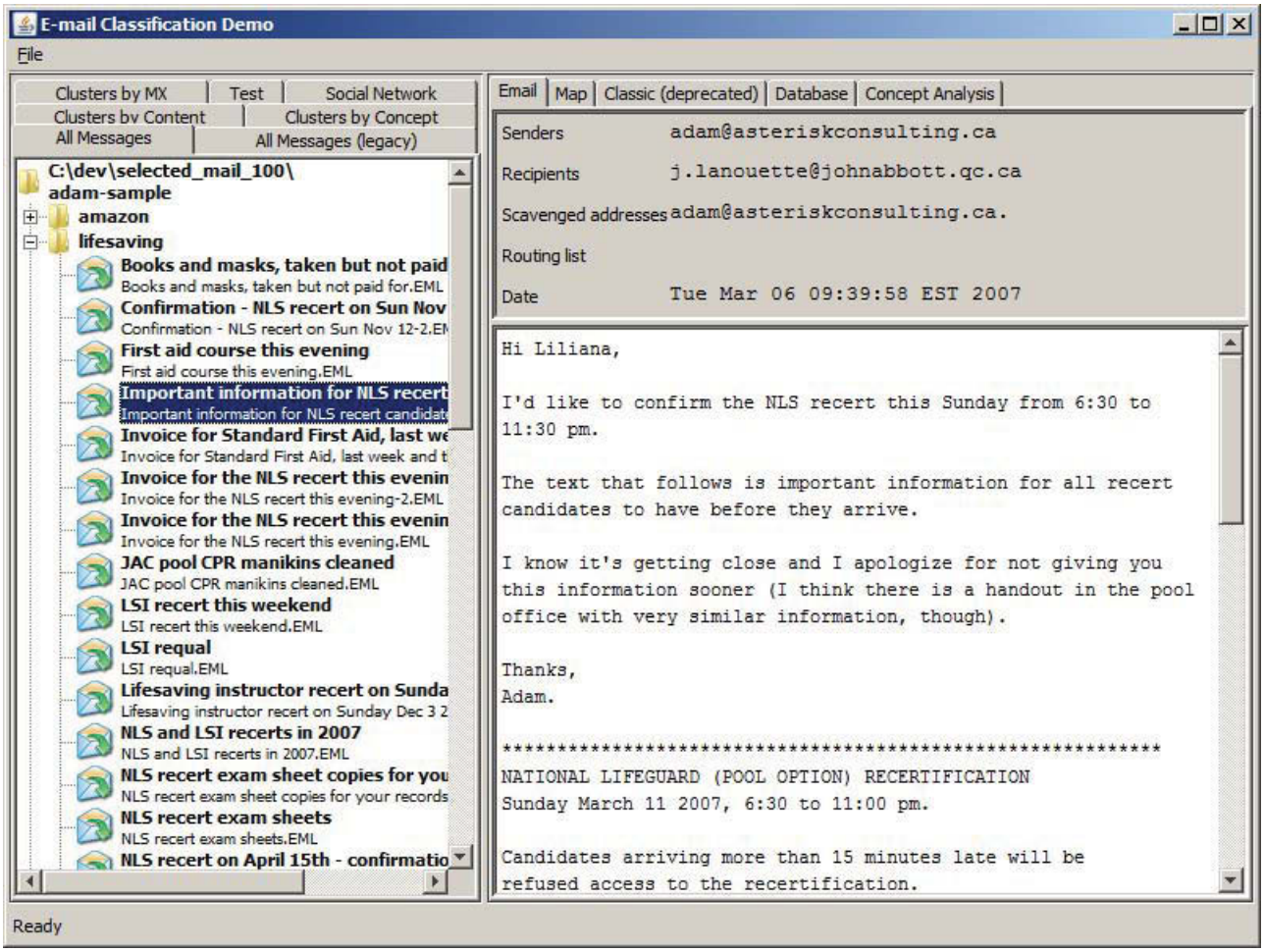

Figure 10: Message Viewer

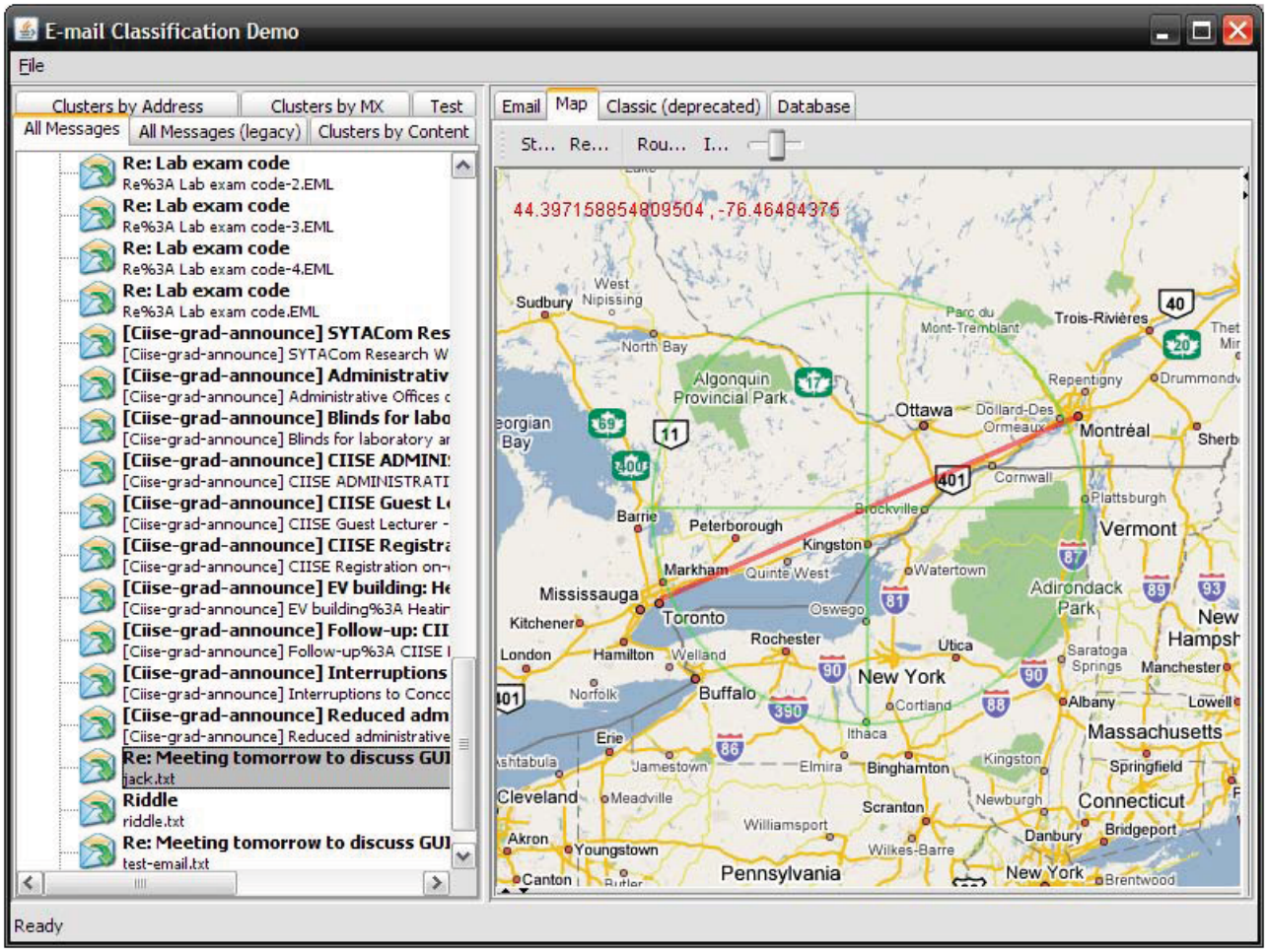

Figure 11: Geographic Map of Message Route

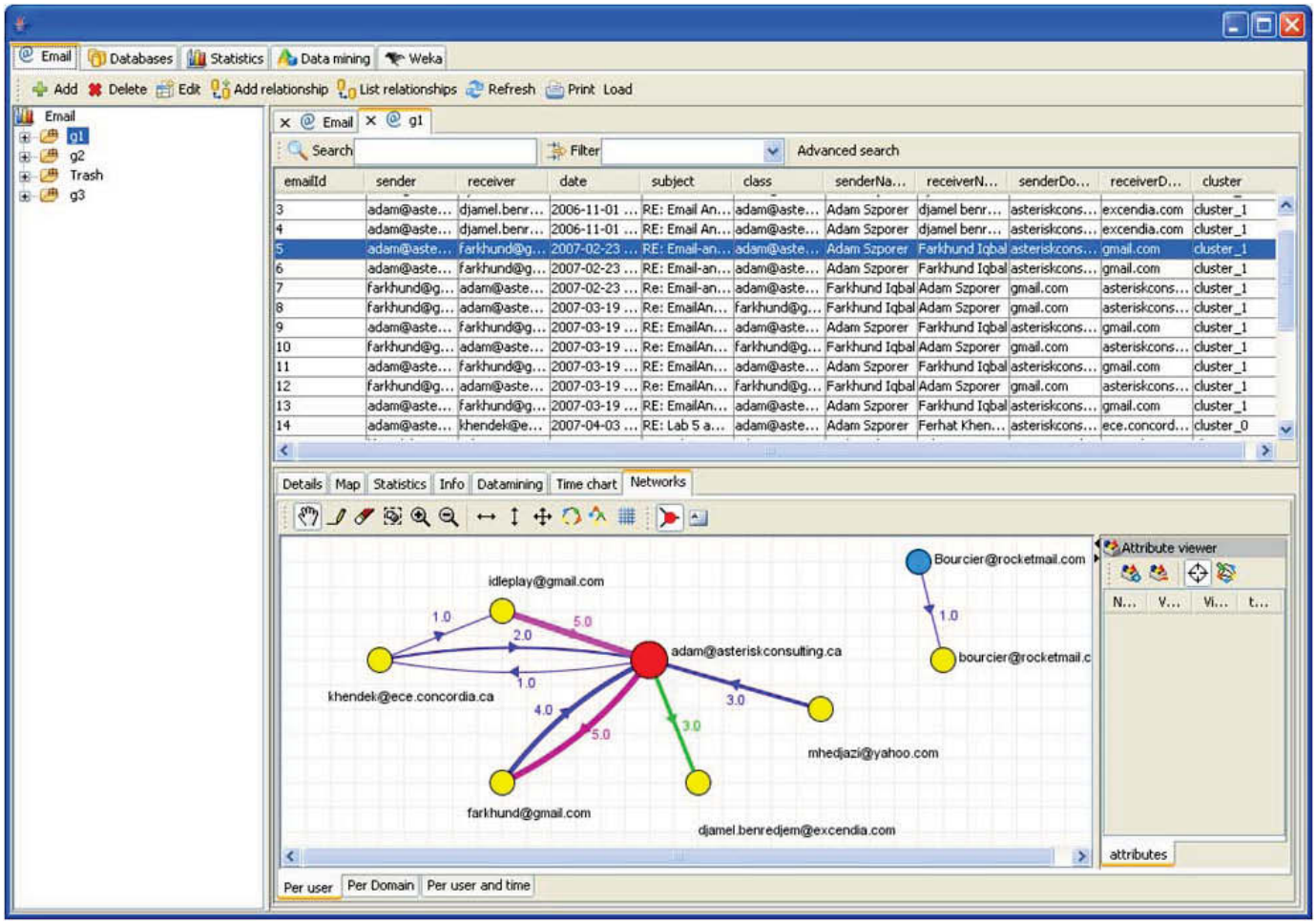

Figure 12: Social Network View

5.8 Summary

This chapter has shown our implementation of e-mail analysis, including detailed algorithms for data extraction, pre-processing, part-of-speech tagging, word pair computation, clustering, verification, and concept analysis. Screen captures demonstrate the utility of our graphical interface, highlighting drag-and-drop adding of e-mails, message browsing, geographic mapping of IP addresses, and the social network representation of an e-mail collection.

Chapter 6

Results

This chapter describes the methodology used to verify our techniques and the way we measure success. Later, it shows the objective results of our e-mail analysis method applied to the Enron e-mail corpus and presents a comparative analysis of our innovations to demonstrate how they increase accuracy. Lastly, this chapter presents the subjective results of our concept analysis technique.

6.1 Methodology

We select ten mailboxes from the publicly-available Enron e-mail corpus and analyze them using several variations of our algorithm. The different variations of our algorithm show the relative usefulness of our clustering methods. The ten mailboxes are:

- hernandez-j
- lavorato-j
- mcconnell-m
- mckay-j

- motley-m
- rodrique-r
- scholtes-d
- shackleton-s
- ward-k
- white-s

Additionally, we selected 111 e-mails hand-picked from the author's e-mail inbox. Each mailbox is structured in folders. This is advantageous because the users have already classified their e-mails. We simply use our algorithm on the mailbox contents and measure our success based on how many messages are classified in the same group as they are found. For performance reasons, we select between 25 and 200 messages from each mailbox. This also allows us to determine how clustering performance varies with mailbox size.

6.2 Measuring Success

For research purposes, we adopt the method used by Sinka and Corne to automatically measure the success of our unsupervised algorithm in a trial dataset [36]:

- Identify the category X that has the highest number of matches with the label L. Mark these matches as "correct".
- Mark as "incorrect" the documents with label L that are not in category X.
- Mark as "incorrect" the documents in category X that do not have label L.
- Mark label L as processed.

- Repeat until all labels are processed.

6.3 Clustering

This section shows the objective results of our extraction, pre-processing, and clustering techniques. First, our complete method is used against the test mailboxes. Then, a modified run of our method selectively excludes certain techniques. The effect of each technique is shown by comparing the modified runs to the complete run.

6.3.1 Complete Run

This run shows the complete implementation of our clustering algorithm, including the removal of opening and closing lines, stemming, and the removal of singleton word pairs. Table 3 shows the clustering results for our complete implementation. The following observations are noted:

- We analyze 11 mailboxes with a total of 1082 messages.
- An average of 17 % ($\sigma = 11\%$[1]) are not clusterable (due to insufficient content).
- Our techniques correctly cluster 80 % of the remaining messages ($\sigma = 6\%$).

Figure 13 shows a tight distribution in success rates, decreasing slightly as the total number of messages increases. Figure 14 shows that larger mailboxes tend to have more unclusterable messages (with a few outliers).

[1] Standard deviation

Table 3: Clustering Results (Complete Implementation)

Mailbox	Sample size	Unclusterable	Correctly clustered (n)	Correctly clustered (%)
adam-sample	111	7	81	78 %
hernandez-j	74	13	49	80 %
lavorato-j	111	13	73	74 %
mcconnell-m	28	4	21	88 %
mckay-j	182	72	92	84 %
motley-m	60	11	36	73 %
rodrique-r	97	20	64	83 %
scholtes-d	147	14	103	77 %
shakelton-s	108	0	90	83 %
ward-k	84	22	56	90 %
white-s	80	17	45	71 %

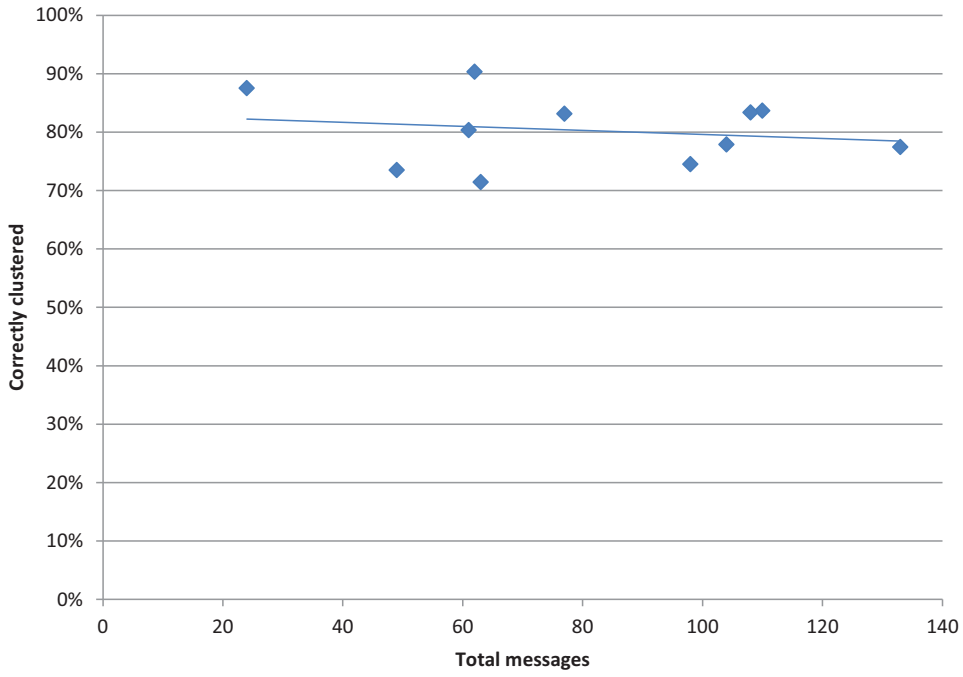

Figure 13: Correctly Clustered Messages (Full Run)

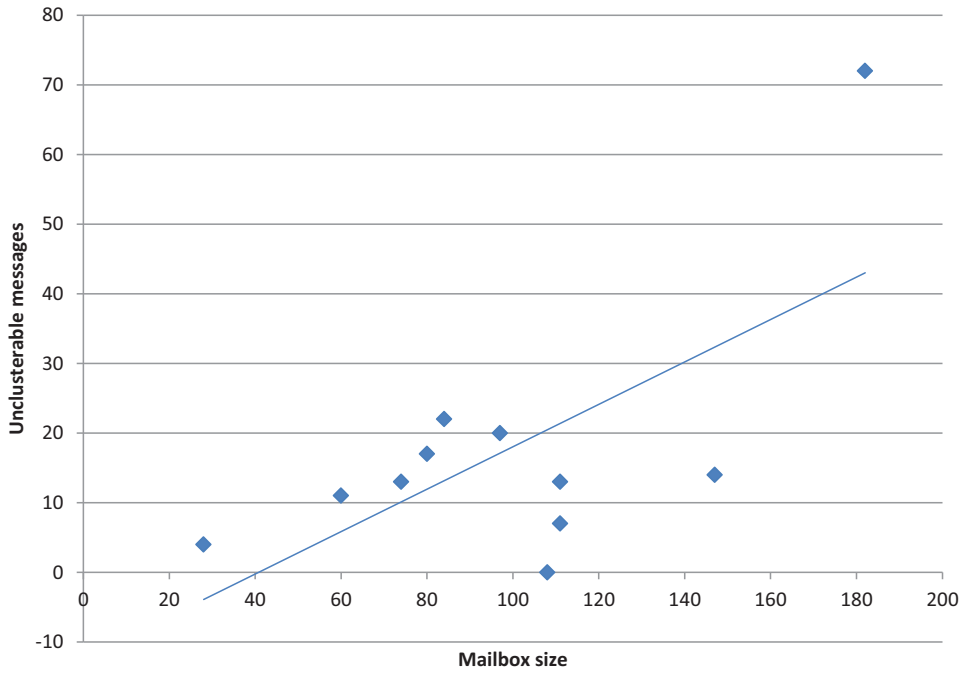

Figure 14: Unclusterable Messages (Full Run)

6.3.2 Modified Runs

The following three sections show modified versions of our analysis method, selectively excluding the removal of opening and closing lines, stemming, and the removal of singleton word pairs. Each modified run is compared to the complete run presented earlier in this chapter.

Without Removing Opening and Closing Lines

For this run, we do not remove short lines at the beginning and at the end of each message, nor do we remove any signature block that is detected. Table 4 shows the clustering results for this run.

Table 4: Clustering Results (Without Removal of Opening and Closing Lines)

Mailbox	**Sample size**	**Unclusterable**	**Correctly clustered (n)**	**Correctly clustered (%)**
adam-sample	111	3	46	43 %
hernandez-j	74	1	59	81 %
lavorato-j	111	6	81	77 %
mcconnell-m	28	0	19	68 %
mckay-j	182	45	116	85 %
motley-m	60	7	33	62 %
rodrique-r	97	9	68	77 %
scholtes-d	147	1	111	76 %
shakelton-s	108	0	90	83 %
ward-k	84	6	70	90 %
white-s	80	1	42	54 %

We observe the following:

- An average of 6 % ($\sigma = 7\%$) are not clusterable (due to insufficient content).
- Only 72 % of the remaining messages ($\sigma = 14\%$) are correctly clustered.

Note that the number of unclusterable messages decreases, because these messages have more content (short opening lines, closing lines, and signature blocks are not removed); however,

- the average number of correctly clustered messages decreases by 8 % and the standard deviation increased by 14 %.
- 3 mailboxes show an average improvement of 1 %.
- 7 mailboxes show an average decline of 13 %.

The results of this run show a marginal improvement in *some* cases, but clearly demonstrate an overall decline in accuracy and deviation, as shown in Figure 15.

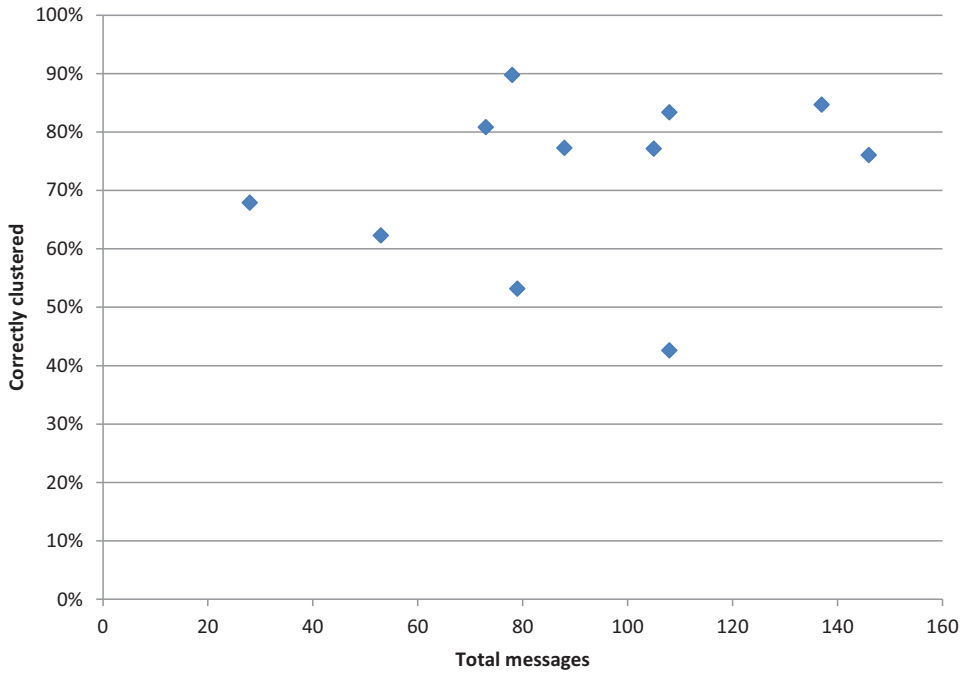

Figure 15: Correctly Clustered Messages (Without Removal of Opening and Closing Lines)

Without Stemming

For this run, we do not stem words to their lexical root. This increases the number of word pairs (some pairs with similar roots will now be counted separately). Table 5 shows the clustering results for this run. As you can see:

- An average of 18 % ($\sigma = 11\%$) are not clusterable (due to insufficient content).
- 80 % of the remaining messages ($\sigma = 9\%$) are correctly clustered.

Note that the number of unclusterable messages and the clustering accuracy remain approximately the same (although slightly worse than the full algorithm); however, stemming greatly reduces the number of features needed for analysis. The results of this run, as you

Table 5: Clustering Results (Without Stemming)

Mailbox	Sample size	Unclusterable	Correctly clustered (n)	Correctly clustered (%)
adam-sample	111	10	78	77 %
hernandez-j	74	15	49	83 %
lavorato-j	111	14	72	74 %
mcconnell-m	28	4	21	88 %
mckay-j	182	75	96	90 %
motley-m	60	13	29	62 %
rodrique-r	97	24	60	82 %
scholtes-d	147	15	102	77 %
shakelton-s	108	0	90	83 %
ward-k	84	22	56	90 %
white-s	80	17	45	71 %

can see in Figure 16, are similar to the full implementation, with a slightly larger deviation.

Without Removing Singleton Word Pairs

For this run, we do not remove infrequently occurring word pairs. Recall that the purpose of removing word pairs that occur only once is to reduce the number of features that do not aid clustering, since they occur only once. Table 6 shows the clustering results for this run. As you can see:

- An average of 11 % ($\sigma = 10\%$) are not clusterable (due to insufficient content).
- Only 64 % of the remaining messages ($\sigma = 15\%$) are correctly clustered.

Note that the number of unclusterable messages decreases (as in the first modified run), because these messages have more content (word pairs that occur only once); however,

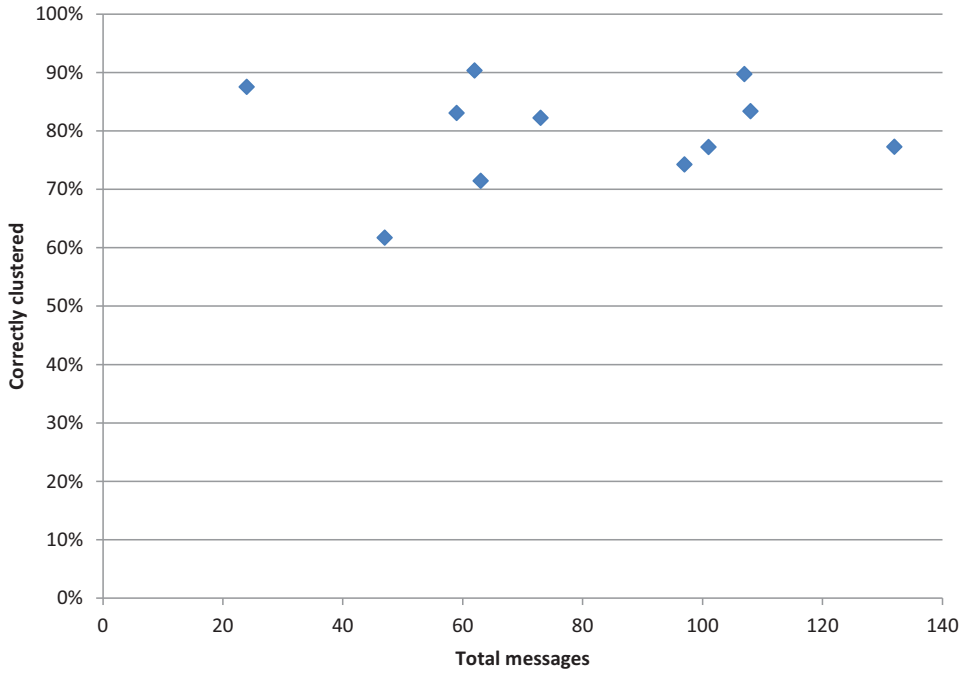

Figure 16: Correctly Clustered Messages (Without Stemming)

Table 6: Clustering Results (Without Removing Singleton Word Pairs)

Mailbox	**Sample size**	**Unclusterable**	**Correctly clustered (n)**	**Correctly clustered (%)**
adam-sample	111	3	58	54 %
hernandez-j	74	12	49	79 %
lavorato-j	111	4	68	64 %
mcconnell-m	28	1	10	37 %
mckay-j	182	57	98	78 %
motley-m	60	4	30	54 %
rodrique-r	97	16	37	46 %
scholtes-d	147	9	88	64 %
shakelton-s	108	0	78	72 %
ward-k	84	18	57	86 %
white-s	80	11	48	70 %

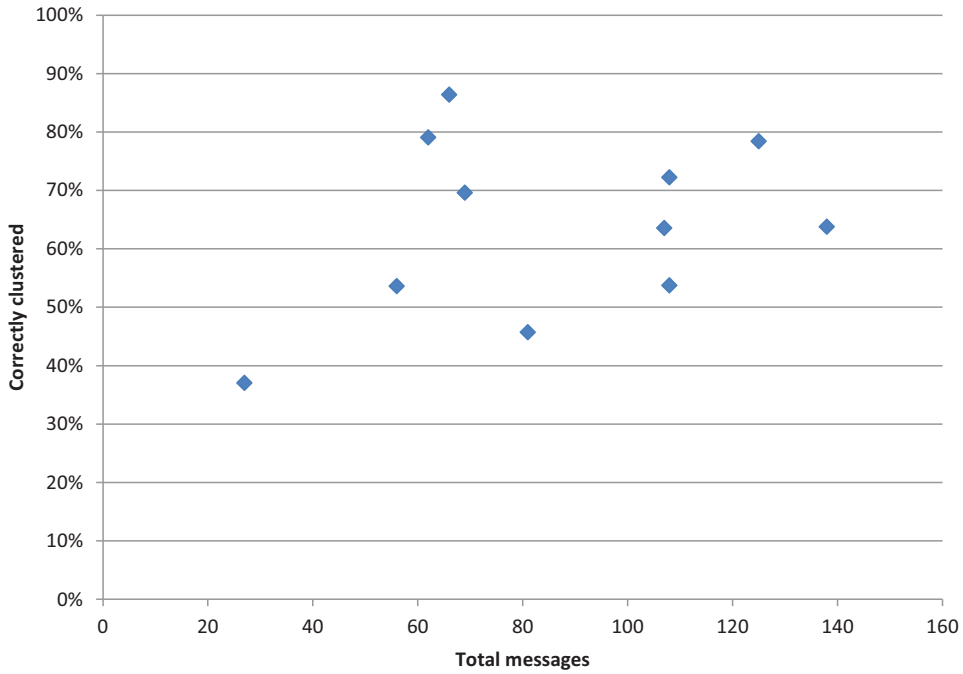

Figure 17: Correctly Clustered Messages (Without Removing Singleton Word Pairs)

- The average number of correctly clustered messages decreases by 16 % and the standard deviation increases by 10 %.
- No mailboxes show an improvement.

The results of this run show a clear benefit to removing singleton word pairs, as shown in Figure 17.

6.3.3 Analysis

The results of each of the four runs (full algorithm, without removal of opening and closing lines, without stemming, and without removing singleton word pairs) are shown together in Figure 18. Each mailbox has four vertical bars, representing the full run, and each of

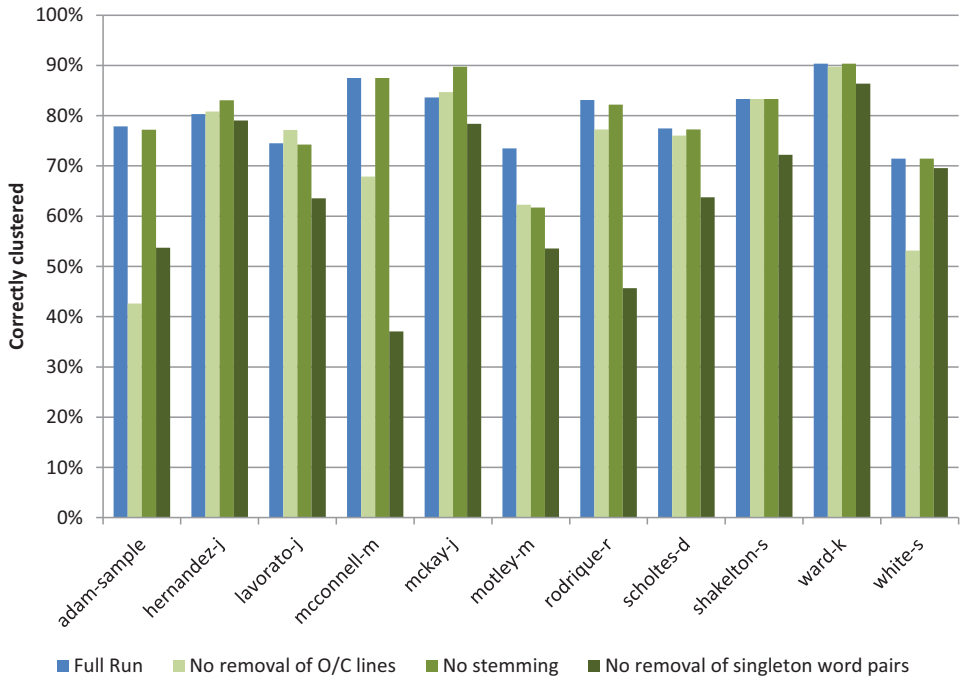

Figure 18: Clustering Runs (Comparison)

the modified runs. You can see that overall, the full algorithm is better than each of the modified runs in clustering accuracy.

A different way to visualize the results is by comparing the modified runs of each mailbox (three bars for each mailbox) to the full run (which is the horizontal axis represented by 0 %). Figure 19 shows that the modified runs are nearly always worse than the full run.

6.4 Concept Analysis

As described in chapter 5, our concept analysis extracts relevant *hypernyms* from each cluster. To help investigators understand the concepts, our software produces a list of ranked concepts and a list of source words that generate them.

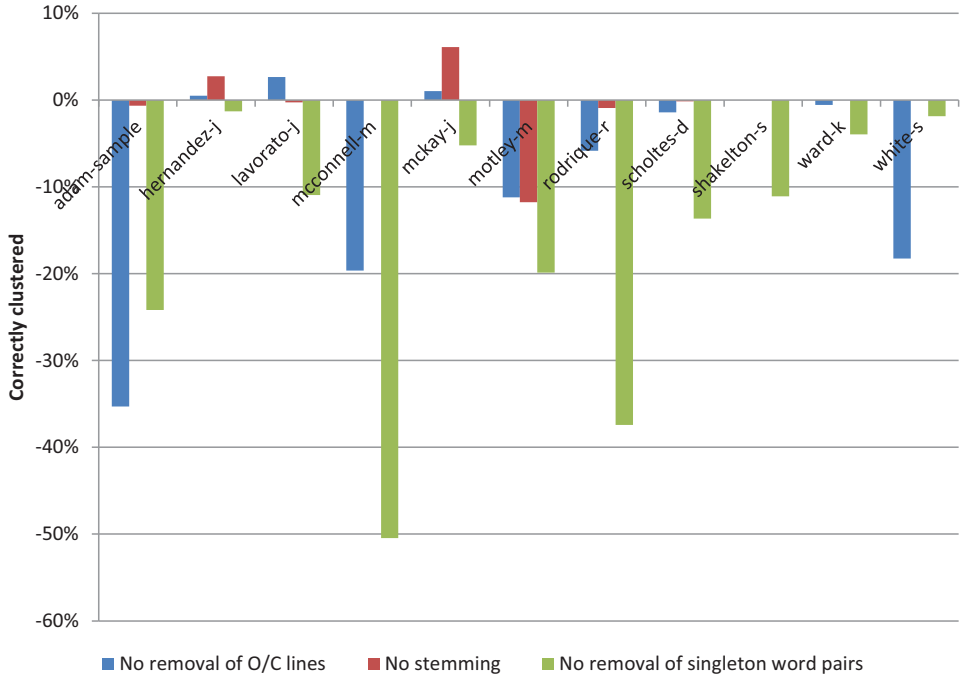

Figure 19: Clustering Runs (Differential Analysis)

6.4.1 Software Output

Here is an example of the output generated for the mailbox mackay-j:

```
***** CLUSTER "C-pipeline_info-1" *****
0.0000 SID-2579744-v: be,
Source: be(v), contain(v), need(v),
0.0291 SID-7092668-n: demand,
Source: call(n),
0.1194 SID-6510930-n: message, content, subject_matter, substance,
Source: direction(n), information(n),
***** CLUSTER "C-pipeline_info-2" *****
0.0000 SID-2579744-v: be,
Source: be(v), buy(v), continue(v), require(v), sell(v), terminate(v),
0.0044 SID-7092668-n: demand,
Source: call(n),
0.0059 SID-2344645-v: act, move,
Source: continue(v), use(v),
0.0101 SID-14914858-n: time_period, period_of_time, period,
Source: day(n), term(n),
0.0134 SID-403481-n: activity,
Source: demand(n), practice(n), provision(n), service(n),
0.0181 SID-108238-v: change,
Source: adjust(v), find(v), have(v),
0.0241 SID-123976-v: change, alter, modify,
Source: adjust(v), affect(v), terminate(v),
0.0318 SID-13469727-n: nautical_linear_unit,
Source: mile(n),
```

```
0.0424 SID-13427890-n: linear_unit,
Source: mile(n), point(n),
0.0482 SID-2190280-v: get, acquire,
Source: buy(v), find(v), have(v),
***** CLUSTER "C-retail-1" *****
0.0000 SID-2579744-v: be,
Source: be(v), buy(v), contain(v), go(v), need(v), require(v), sell(v), start(v),
take(v), want(v),
0.0023 SID-2344645-v: act, move,
Source: go(v), start(v), take(v),
0.0053 SID-403481-n: activity,
Source: help(n), role(n), use(n),
0.0095 SID-123976-v: change, alter, modify,
Source: exchange(v), get(v), think(v),
0.0123 SID-14914858-n: time_period, period_of_time, period,
Source: day(n), month(n), morning(n), year(n), yr(n),
0.0126 SID-20846-n: artifact, artefact,
Source: facility(n), sheet(n), thing(n),
0.0158 SID-1818343-v: travel, go, move, locomote,
Source: do(v), follow(v), return(v),
0.0166 SID-143724-v: change_state, turn,
Source: close(v), get(v), go(v),
0.0221 SID-108238-v: change,
Source: find(v), get(v), give(v), go(v), have(v), produce(v), take(v),
0.0222 SID-621052-v: think, cogitate, cerebrate,
Source: give(v), think(v),
0.0277 SID-1814387-v: move,
Source: close(v), give(v), start(v), take(v),
0.0284 SID-1602857-v: make, create,
Source: do(v), get(v), give(v), produce(v), reproduce(v), start(v), write(v),
0.0316 SID-2190280-v: get, acquire,
Source: buy(v), find(v), get(v), have(v), take(v),
0.0444 SID-732231-v: communicate, intercommunicate,
Source: get(v), give(v), sign(v), write(v),
***** CLUSTER "C-quotes-1" *****
0.0000 SID-2579744-v: be,
Source: be(v), contain(v), continue(v), need(v), start(v),
0.0143 SID-2344645-v: act, move,
Source: continue(v), create(v), start(v),
0.0351 SID-1602857-v: make, create,
Source: create(v), do(v), start(v),
0.0391 SID-14914858-n: time_period, period_of_time, period,
Source: night(n), time(n),
0.0457 SID-403481-n: activity,
Source: business(n), effort(n), help(n), help(v), location(n), market(n), use(n), work(n),
0.0617 SID-2452043-v: appoint, charge,
Source: authorize(v), create(v), name(v),
***** CLUSTER "C-pipeline_info-3" *****
0.0000 SID-2579744-v: be,
Source: be(v), continue(v),
0.0224 SID-2344645-v: act, move,
Source: continue(v),
0.1224 SID-1818343-v: travel, go, move, locomote,
Source: continue(v), follow(v),
```

In the case of this mailbox, we see that there are five clusters. The clustering algorithm splits one folder into multiple clusters, even though there are only three source folders.

The clusters are:

- retail-1
- quotes-1
- pipeline_info-1
- pipeline_info-2
- pipeline_info-3

Looking at the quotes-1 cluster as an example, we see that the two highest-ranked concepts are:

- WordNet synset 2452045, verb: appoint, charge
- WordNet synset 403481, noun: activity

We also see that the source words from the text that contribute to these top-ranked concepts are:

- business(n)
- effort(n)
- help(n)
- help(v)
- location(n)
- market(n)
- use(n)
- work(n)
- authorize(v)
- create(v)
- name(v)

Here is an excerpt from an e-mail in the quotes group:

```
jon,
i had no #'s yesterday because andy was out and adam left us to return to
london for good. hence no northeast markets. we  are helping the canadian
office each day and andy's back today at the end of the day i'll fill out as
much as possible.
john
```

6.4.2 Analysis

The results of our concept analysis are shown in Table 7. By examining the source e-mails, we observe that the concepts reflect their general nature.

Table 7: Concept Analysis

Mailbox	**Cluster**	**Top terms ($tf * idf$ value)**
adam-sample	vanier-1	gathering (0.0131), interact (0.0109)
	amazon-1	perceive (0.0211), acquire (0.0197), coming_together (0.0158), time_period (0.0110)
	lifesaving-1	remove (0.0137), communicate (0.0117)
hernandez-j	judy-1	assets (0.0071), content (0.0059), get (0.0056)
	daily_blessings-1	supply (0.0138), location (0.0115), get (0.0102)
	daily_blessings-2	have (0.0114), activity (0.0091)
lavorato-j	offsite-1	hit (0.0207), create_from_raw_material (0.0156), change (0.0153)
	personal-1	act (0.0108), use (0.0072)
	personal-2	communicate (0.0181), take_place (0.0135)
mcconnell-m	garcia_munte-1	make (0.0557), change (0.0477), travel (0.0318)
	art_committee-1	label (0.0235), point (0.0115)
	enhome-1	use (0.0447), point (0.0220)
mckay-j	pipeline_info-1	message (0.1194), demand (0.0291)
	pipeline_info-2	acquire (0.0482), linear_unit (0.0424), nautical_linear_unit (0.0318)
	retail-1	communicate (0.0444), acquire (0.0316), create (0.0284)
	quotes-1	appoint (0.0617), activiry (0.0457), time_period (0.0391)
	pipeline_info-3	travel (0.1224), act (0.0224)
motley-m	belden-1	hold (0.0241), move (0.0172), change (0.0155)
	risk_reports-2	protect (0.0560), product (0.0420)
	risk_reports-3	time_period (0.3010)
rodrique-r	dublin-1	get (0.2007), point (0.1003)
	dublin-2	get (0.2007), point (0.1003)
	canada-1	change (0.0483), time_period (0.0290)
	file-1	think (0.1193)
	file-2	give (0.1415), change (0.1084), compartment (0.1061)
scholtes-d	ferc-1	change (0.0054), time_period (0.0052), decrease (0.0040), activity (0.0035)
	transmission-1	touch (0.0651), change_state (0.0240)
	west_bank-1	alter (0.0053), time_period (0.0043), artifact (0.0036)
shakelton-s	organizational_changes-1	act (0.0195), create (0.0159), activity (0.0131)
	credit_watch-2	displace (0.0739)
	restricted_list-1	travel (0.0593), position (0.0446)
ward-k	citizens_utilities-1	alter (0.0000)
	citizens_utilities-2	change (0.0110), statement (0.0085), employ (0.0073)
white-s	legal-1	acquire (0.3010)
	legal-2	change (0.3010)
	cya-1	displace (0.0199), travel (0.0184), alter (0.0153)

6.5 Summary

The results shown in this chapter demonstrate that our approach to e-mail analysis is correct for the test data set. By comparing the results with and without the removal of short opening and closing lines, stemming, and word pair analysis, we have shown the advantage of each technique. Lastly, we have shown the results of our concept analysis method based on WordNet.

Chapter 7

Conclusion

This thesis has examined the current repertoire of techniques that exist for e-mail analysis, from the e-mail analysis process, to extraction, pre-processing, clustering, and concept analysis, through a thorough literature review. Our research has advanced the state of the art by creating new methods, such as e-mail domain analysis, and word pair analysis. Lastly, we have verified these new methods by applying them to an existing body of e-mails and presenting the results. We have created software to implement part of the e-mail analysis process, and we have demonstrated several innovations that will serve to aid further research in this field, by stimulating the development of even better techniques, and by inspiring developers who can bring these innovations into viable toolkits for investigators to use.

As future work, our approach can be applied to many more e-mail container formats, including proprietary ones such as Microsoft Exchange. It is also possible to mine e-mails

for attachments and let these aid the clustering and concept analysis process. Our proof-of-concept implementation, while functional, does not provide the performance necessary for real-world use. The algorithms, implemented in Java, can be re-implemented and optimized in C++ to improve execution time.

Bibliography

[1] Giordano Adami, Paolo Avesani, and Diego Sona. Clustering documents into a web directory for bootstrapping. *Data & Knowledge Engineering*, 54:301–325, 2005.

[2] Juan José Garcaí Adevaa and Juan Manuel Pikatza Atxa. Intrusion detection in web applications using text mining. *Engineering Applications of Artificial Intelligence*, 20:555–566, 2007.

[3] Nicole Lang Beebe and Jan Guynes Clark. Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results. *Digital Investigation*, 4S:S49–S54, 2007.

[4] Isabelle Bichindaritz and Sarada Akkineni. Concept mining for indexing medical literature. *Engineering Applications of Artificial Intelligence*, 19:411–417, 2006.

[5] Olivier de Vel. File classification using byte sub-stream kernels. *Digital Investigation*, 1:150–157, 2004.

[6] Olivier de Vel, Malcolm Corney, Alison Anderson, and George Mohay. Language and gender author cohort analysis.

[7] David H. Crocker (ed.). Internet Message Format. http://www.ietf.org/rfc/rfc5322.txt, October 2008. Accessed 2011-11-29.

[8] P. Resnick (ed.). Standard for the Format of ARPA Internet Text Messages. http://www.ietf.org/rfc/rfc822.txt, August 1982. Accessed 2011-11-29.

[9] C.-P. Wei et al. A collaborative filtering-based approach to personalized. *Decision Support Systems*, 2007.

[10] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. http://www.ietf.org/rfc/rfc2045.txt, November 1996. Accessed 2011-11-29.

[11] Stanford Natural Language Processing Group. Stanford Log-linear Part-of-speech Tagger. http://nlp.stanford.edu/software/tagger.shtml, 2011.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.

[13] Wen-Feng Hsiao and Te-Min Chang. An incremental cluster-based approach to spam filtering. *Expert Systems with Applications*, 34:1599–1608, 2008.

[14] Farkhund Iqbal, Hamad Binsalleeh, Benjamin C.M. Fung, and Mourad Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *Digital Investigation*, 2010.

[15] Jacob Köhler, Stephan Philippi, Michael Specht, and Alexander Rüegg. Ontology based text indexing and querying for the semantic web. *Knowledge-Based Systems*, 19:744–754, 2006.

[16] Teuvo Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[17] LINC Laboratory. Alphabetical list of part-of-speech tags used in the Penn Treebank Project. `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`, 2011. Accessed 2011-11-29.

[18] LINC Laboratory. Penn Treebank. `http://www.cis.upenn.edu/~treebank`, 2011. Accessed 2011-11-29.

[19] C. H. Lee and H.-C. Yang. A text mining approach on automatic generation of web directories and hierarchies. *Expert Systems with Applications*, 27:645–663, 2004.

[20] C. H. Lee and H.-C. Yang. Construction of supervised and unsupervised learning systems for multilingual text categorization. *Expert Systems with Applications*, 2008.

[21] Yanjun Li, Soon M. Chung, and John D. Holt. Text document clustering based on frequent word meaning sequences. *Data & Knowledge Engineering*, 64:381–404, 2008.

[22] Peter Lyman and Hal R. Varian. How Much Information 2003? `http://www2.sims.berkeley.edu/research/projects/how-much-info-2003`, 2003. Accessed 2008-02-15.

[23] Marie-Hélène Masson and T. DenIJux. Ecm: An evidential version of the fuzzy c-means algorithm. *Pattern Recognition*, 41:1384–1397, 2008.

[24] Dunja Mladenic and Marko Grobelnik. Feature selection on hierarchy of web documents. *Decision Support Systems*, 35:45–87, 2003.

[25] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, pages 47–60, November 1996.

[26] Patrick Perrin and Frederick E. Petry. Extraction and representation of contextual information for knowledge discovery in texts. *Information Sciences*, 151:125–152, 2003.

[27] Aurora Pons-Porrata, Rafael Berlanga-Llavori, and José Ruiz-Shulcloper. Topic discovery based on text mining techniques. *Information Processing & Management*, 42:752–768, 2007.

[28] Apache James Project. mime4j. `http://james.apache.org/mime4j/status.html`, 2011. Accessed 2011-11-29.

[29] CALO Project. Enron e-mail dataset. `http://www.cs.cmu.edu/~enron`, 2009. Accessed 2011-02-22.

[30] T.A. Runkler and J.C. Bezdek. Web mining with relational clustering. *International Journal of Approximate Reasoning*, 32:217–236, 2003.

[31] S. Sakurai and A. Suyama. An e-mail analysis method based on text mining techniques. *Applied Soft Computing*, 6:62–76, 2005.

[32] Ridvan Saraçoglu, Kemal Tütüncü, and Novruz Allahverdi. A fuzzy clustering approach for finding similar documents. *Expert Systems with Applications*, 33:600–605, 2007.

[33] Arno Scharl and Christian Bauer. Mining large samples of web-based corpora. *Knowledge-Based Systems*, 17:229–233, 2004.

[34] Computer Science and Artificial Intelligence Laboratory. MIT Java WordNet Interface. `http://projects.csail.mit.edu/jwi`, 2011.

[35] Farial Shahnaz, Michael W. Berry, V. Paul Pauca, and Robert J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42:373–386, 2006.

[36] Mark P. Sinka and David W. Corne. The banksearch web document dataset: investigating unsupervised clustering and category similarity. *Journal of Network and Computer Applications*, 28:129–146, 2004.

[37] David Sánchez and Antonio Moreno. Learning non-taxonomic relationships from web documents. *Data & Knowledge Engineering*, 64:600–623, 2008.

[38] S. Tan. Neighbor-weighted k-nearest neighbor for unbalanced text corpus. *Expert Systems with Applications*, 28:667–671, 2005.

[39] S. Tan. An effective refinement strategy for knn text classifier. *Expert Systems with Applications*, 30:290–298, 2006.

[40] S. Tan. An improved centroid classifier for text categorization. *Expert Systems with Applications*, 2007.

[41] William-Chandra Tjhi and Lihui Chen. Possibilistic fuzzy co-clustering of large document collections. *Pattern Recognition*, 40:3452–3466, 2007.

[42] Yuen-Hsien Tseng, Chi-Jen Lin, and Yu-I Lin. Text mining techniques for patent analysis. *Information Processing & Management*, 43:1216–1247, 2007.

[43] Princeton University. WordNet. `http://wordnet.princeton.edu/wordnet`, 2011. Accessed 2011-11-29.

[44] Sung-Shun Weng and Chih-Kai Liu. Using text classification and multiple concepts to answer e-mails. *Expert Systems with Applications*, 26:529–543, 2004.

[45] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, second edition, 2005.

[46] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90, 1999.

[47] Zhu Zhang. Mining relational data from text: From strictly supervised to weakly supervised learning. *Information Systems*, 33:300–314, 2007.