An exploratory study on the automated classification of email importance using the Enron email corpus

James Dowdall

Bachelor of Science in Computer Science with Honours
The University of Bath
April 2014

This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:

Development of an automated classifier of email importance using the Enron email corpus

Submitted by: James Dowdall

COPYRIGHT

Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the author unless otherwise specified below, in accordance with the University of Bath's policy on intellectual property (see http://www.bath.ac.uk/ordinances/22.pdf).
This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:

Abstract

This project provides an introductory stepping stone to the field of Email Classification using Natural Language Processing and Machine Learning, in the context of Email Importance. The project performs a large amount of research and investigation, touching on work in Spam Detection and many other relevant areas. It provides an overview of the common processes, tools and datasets used by the community. The project designs, implements and publishes an open source suite of tools for parsing one of the most popular datasets, performing several of the processing methods. This processed data is then used as input for the WEKA tool, used to create two classifiers of email importance. To conclude, the project discusses a large amount of potential future work, to improve on or add to the suite provided here.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Simon Jones for his frequent advice and support over the course of the project, specifically his help with WEKA and for providing the initial LIWC data script. Stephen Payne for supervising the project and William Cohen for hosting the Enron dataset for the whole academic community.

Chapter 1

Introduction

1.1 Problem Description

Email overload is not a new nor poorly understood issue, the first studies into the issue began in the mid 90s (Whittaker and Sidner, 1996; Balter, 1997), and yet almost 20 years later the problem is still very much present throughout businesses worldwide.

The problem has evolved to the point that being able to manage and prioritise email has become a vital business skill. Being able to balance email management with other work activities is required to be effective in the workplace, but can place quite significant stress on employees (Dabbish and Kraut, 2006).

While email overload itself is a psychological phenomenon, it is dependent on a number of physical factors. The stress caused is influenced by the sheer the volume of email received, the role email plays in a person's job and the method of email management, to name a few (Dabbish and Kraut, 2006).

Some of these factors are an intrinsic characteristic of a given job role, for example professionals spend 35-50% of their day communicating whereas managers spend 50-80% (Mintzberg, 1973). However some of the other factors are more directly addressable; the method managing email specifically has been subject to much scrutiny over the years and yet one dominant method has not yet emerged.

While there are many methods controlling how and when to check email, the methods this paper is going to concentrate on are those concerning how email is organised. The traditional approach is manual folder categorisation and filing, explored by Whittaker and Sidner (Whittaker and Sidner, 1996).

This method is a very resource intensive manual process that has a whole suite of problems, naming folders is cognitively difficult and the filing structure may not even reflect a future data access need (Whittaker and Sidner, 1996). A common improvement on manual filing is the introduction of automated filtering using rules. While automated rules cut down on the manual sorting, these rules are quite inflexible and while effective at removing repetitive emails in a standardised format, these cannot handle large volumes of varied email (Manco, Masciari, Ruffolo and Tagarelli, 2002).

There has been some development on this topic, with most work attempting to use existing folders to predict future classification, using various labelled datasets or pre-existing folder structures (Mock, 2001; Chan, Koprinska and Poon, 2004; Bekkerman, McCallum and Huang, 2005). It becomes increasingly obvious that these methods are entirely reliant on a pre-existing, labelled dataset to function, and this is just not a realistic assumption to make in the common business environment.

Indeed, this begins to suggest a paradox of sorts, these methods work best on well categorised and labelled datasets, and yet as Whittaker and Sidner showed in 1996, and Fisher, Brush, Gleave and Smith revisited in 2006, over 70% of users did not maintain the sort of well filed inboxes that these methods require to function effectively (Whittaker and Sidner, 1996; Fisher, Brush, Gleave and Smith, 2006).

In effect, this work was helps users who already file their email well, to file their email better, while giving little help to the users already suffering from email overload. A lot of the research has focussed on categorising emails, however that may be at the expense of losing sight of the higher level goal that folders served - a method of organising a users email inbox.

To clarify, take the example of a typical manager suffering from email overload. If we assume that they are in the 32% of users that don't implement any structured filing system (Fisher et al., 2006), which is not a large stretch considering the higher email volume associated with the manager role (Whittaker and Sidner, 1996), we can see that almost all of the approaches above are not particularly helpful.

Extending this example, suppose that the manager has a 15 minute gap between meetings, how do they go about tackling the large volume of email in their inbox? There is not enough time to read through all the emails, where should they start? Without more information there is not a particularly good answer to this problem, yet an automated classifier could help greatly in this situation.

This sort of situation is just one example of why email folder classification is just one facet of the email overload problem. Even if the manager had the folder categorisation, that information may not aid them in choosing which emails to deal with first. In essence, the manager doesn't need to know which folder the email should be filed into, instead they need a classifier of the perceived email importance, allowing them to deal with the most important emails first.

Importance is ultimately a tricky concept to strictly classify, as at its core importance is a subjective measure. While importance as a whole is certainly objective, assumptions can be made to identify some email types that will likely have different levels of importance; for example by definition, we can assume that all users will find spam email unimportant (Spamhaus, 1998). These concepts are explored in far greater depth in my Literature Review (section 2.2).

There has been some valuable work in this field which could certainly be correlated with importance, Carvalho and Cohen performed some work classifying "speech acts" in emails, which certainly hold some promise in terms of correlation with importance (Carvalho and Cohen, 2005). Dabbish, Kraut, Fussell and Kiesler performed some very relevant work predicting action received emails, which again holds promise (Dabbish, Kraut, Fussell and Kiesler, 2005; Dabbish, Kraut, Fussell and Kiesler, 2004). As above, a far more in-depth investigation into definitions of importance occurs in Literature Review section 2.2.

The most effective way of solving this problem is the development of an automated email classifier. Classifiers are made up of a number of standardised components as below:

- Representative Dataset
- Test hypothesis/definition
- Dataset parser and processor

- Set of analysis tools
- Machine learning tool

These elements themselves will vary depending on the intended usage, but the structure remains constant.

When considering initial definitions of importance, it becomes immediately obvious that this is a vast area of research, which is certainly not addressable in just one piece of work. However, while hypotheses change, the other elements are far more static; initial work identified a large amount of studies using the same machine learning tools and datasets for example.

1.2 Project Objectives

In order to foster more work in this area this paper bills itself as an exploratory study. By that the paper intends to introduce a user to the issue, discussing a large amount of the existing research in this field, moving on to discussing specific tools and datasets commonly used.

Once this initial introduction has been completed, a suite of tools for interacting with the most common elements will be provided. Specifically this will take the form of a number of scripts for manipulating and parsing datasets. This work will all be provided in a freely accessible online repository, developed in an open source manner.

In essence, this paper serves as a stepping stone into the wider research field of Natural Language Processing and the concepts of Email Importance and Email Overload.

To realise this vision, this set of of overarching project objectives have been formed:

1. Carry out a thorough survey of existing work, specifically focussing on the various datasets, tools and methods used in other work.
2. Identify an appropriate email dataset.
3. Investigate a variety of potential definitions to inform my definition of email importance.
4. Identify a number of potential features based on these initial definitions.
5. Design and Implement a tool for parsing and processing the given dataset.
6. Produce a classifier based on the previously determined features to produce a statistical model of a given emails importance.
7. Host completed work online for use in future studies.

1.3 Document Structure

The following chapters describe the design and implementation of the components defined above. These are organised as follows:

2 Literature Review - This chapter explores the wealth of existing work in this area, paying particular attention to previously used datasets, features potentially correlated with importance, tools for text analysis and machine learning tools. This chapter is explicitly limited in scope, it investigates at the high level. Any elements requiring more in depth analysis are investigated in the Technical Investigation chapter.

3 Technical Investigation - This chapter is a continuation of the Literature Review. Any components that require more in-depth technical analysis are discussed here. This chapter forms a bridge between the exploratory and high level Literature Review and the low level and exacting requirements specification.

At the end of this chapter I perform a research summary, summarising my Literature Review and Technical Investigations. Here I will make informed decisions on what to use going forward in my project based on my research and investigation.

4 Requirements - This chapter covers the definition of the system requirements. This chapter will depend heavily on the decisions made in the research summary performed in chapter 3.

The requirements will also leave enough flexibility in the system to allow easy system re-design in response to initial classification results. This chapter will also discuss the development technologies and tools, important to the exploratory part of the project.

5 Design - This chapter is intended to define the High Level Architecture of the system, decomposing the system into its modular components and their interfaces. The Design will also include a justification of decisions and map the Requirements to their relevant Design parts.

6 Implementation - This chapter identifies the key challenges in system implementation, justifying important design choices and paying particular attention to non-functional requirements. Any deviations from the Design will be fully defined and documented here.

7 Analysis - This chapter covers the processing performed on the dataset after the initial dataset has been parsed. This includes work reducing the features, normalising data, removing outliers and so on.

This chapter provides the results of the classification, highlighting feature correlation and overall accuracy. A summary of the success of the work will also be provided.

8 Future Work - This chapter reflects on my work, describing potential improvements and suggesting some directions for future research.

9 Conclusion - This chapter discusses the overall significance of my work, and how successfully the project has met its aims.

Chapter 2

Literature Review

2.1 Overview

To re-iterate my introduction, the role of the Literature Review is as a counterpart to the Technical Investigations (Chapter 3). Together these two chapters will explore the research areas defined below, with the main goal of informing my further work. I also intend to begin formulating some initial hypotheses and definitions to use when performing my analysis.

The scope of the Literature Review itself is explicitly limited in depth. Where the Literature Review identifies an element of interest with sufficient technical depth, the Literature Review will provide a high level overview and a reference, while the Technical Investigation provides the finer grain depth study and discussion.

I have identified these areas to initially focus my research in:

- Existing research in the field of Importance
- Email classification datasets
- Features identified in previous work
- Analysis tools used in previous research

Each of these areas are key components of my project, and by extension - key components of the classifier. Given that the project is intended to be exploratory understanding existing research is very important to ensure that the project provides a well grounded baseline that others can work from.

After performing my Literature Review and Technical Investigations I perform a summary of these two sections, identifying the existing work that is relevant to my project that I will use to support my work.

2.2 Importance

As mentioned in the introduction, Importance is the central concept for this project. However, at its core Importance is a subjective measure; every person will have their own thoughts and feelings on what is important. However, that is not to say that every persons preferences will vary wildly. Indeed it is my belief that at least in the context of email, these preferences will be tightly aligned, a view supported by Dabbish et al. (Dabbish et al., 2004; Dabbish et al., 2005).

2.2.1 Subjectivity in Importance

The research community has skirted around this issue in a manner. Rather than defining importance as the main goals of their studies, the studies focus more on potential measures

of importance. For example, Carvalho and Cohen perform work on predicting email speech acts, inferring what the purpose of an email is from its content (Carvalho and Cohen, 2005). It is an easy stretch to correlate an emails purpose with its importance. Indeed of the 8 different email purposes defined by Carvalho and Cohen, one could even go as far as to allow the user to prioritise these purposes, avoiding the problem of subjectivity altogether, in exchange for a small amount of manual overhead.

Dabbish et al. continues in a similar vein, predicting action on email (Dabbish et al., 2004). Again the correlation with importance is obvious and this paper goes as far as to explicitly mention it as one characteristic correlated with reply prediction. Interestingly, this paper states that while most important emails are replied to, it is not the case that all emails replied to are particularly important.

This has many implications, most importantly showing that importance is not the only feature that is correlated with email reply. The ease of reply is one that is explicitly mentioned, which does seem to make sense when considered in the context of email overload; if an email is very easily dealt with it can be immediately discarded, no longer contributing to email overload.

Other papers deal with subjectivity in a different manner. Kilander dealt with subjectivity by identifying common features in a large user survey. This approach confirms my claim that most users have similar metrics for what defines an important email, with at least 20% of users agreeing on 8 separate features (Kilander, 1996) (discussed further in **??**).

This paper, along with the follow up work creating a classifier for usenet news (Kilander, Fåhræus and Palme, 1997), is especially relevant to my work. Kilander et al.s work documents a classifier to "Select all interesting messages not marked as read, and order by importance." This goal entirely encompasses my goal, so while the datasets are different, the techniques used are very relevant to my study. Given this relationship to my work, I explore this in greater depth in section 3.3 of my Technical Investigation.

2.3 Spam

When discussing importance in the context of email it is difficult not to consider research into spam. In many ways, spam can be considered the inverse of importance. Where a user may want to deal with an important email first, they almost certainly would want to deal with spam email last, if at all. It is prudent then, to look into research on spam to explore whether methods used in identifying spam emails are equally applicable to determining important emails.

Email spam has been around almost as long as the Internet itself, the first mention of "the Junk Mail Problem" coming in 1975 (Postel, 1975), just 6 years after the implementation of ARPANET, with the first Spam Email being sent by Gary Theurk in 1978 (Hedley, 2006).

The universally accepted definition of spam is email that is both bulk and unsolicited (Spamhaus, 1998); specifically if both conditions 1 AND 2 hold:

1. "The recipient's personal identity and context are irrelevant because the message is equally applicable to many other potential recipients."
2. "The recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent."

2.3.1 Spam Detection

There are an absolute plethora of methods for detecting spam, from high level DNS blacklists down to simple block lists, far too many for me to discuss here. However, in the context of this project I am only interested in methods that have some relevance to natural language processing, so am able to cut this large list down. The main method of interest is that of Hybrid Spam Filtering, a method that makes extensive use of machine learning techniques.

As described by Guzella and Caminhas, the usual approach is to use a fairly static technique of training a machine learning classifier on a pre-labelled dataset of spam emails (Guzella and Caminhas, 2009). This technique is better than many other methods, however the predictive power and success of this method is ultimately dependant on the quality of the labeled learning set.

As Spam is a very fast moving area, it is possible that a method like this could be out of date as soon as it has been generated. Indeed, this method will require the training sets to be almost constantly updated, which is not entirely ideal for a solution implemented at the user level. The study does acknowledge that this is one of the key problems with this approach, and discusses some solutions. However, the field of email importance evolves much slower so this becomes a non-issue, and does not need to be investigated.

In terms of feature identification, this paper makes repeated reference to the commonly used Bag-of-words model (also known as the Vector Space Model), identifying its popularity for this sort of work. This approach is equally applicable to both Spam detection and Importance Analysis, and will require further investigation.

This work is very useful when considering different methods of feature selection, identifying Information Gain as by far the most popular method for selecting features (being used in 26 works, as opposed to the next most popular method being used in 2), a very telling result that I can use in my work.

One way in which the spammers can stay ahead of the new spam detection tools is by obfuscating their messages, using different character sets, hidden formatting (such as tables) and unusual punctuation to enable their spam messages to avoid detection by the traditional bag-of-words approach. As a result, the pre-processing and textual analysis tools in the spam detection field are far more sophisticated by necessity, and many of the lessons learned are equally applicable to the work I am performing.

The paper by Blanzieri and Bryl explores some interesting work in addressing one of the problems also highlighted in the Guzella and Caminhas paper, the problem of the reliance on

the initial training set (Blanzieri and Bryl, 2008). The solutions concern various methods of performing co-training, that is performing semi-supervised learning on a small initial dataset, using those findings to train the classifier further. This could be potentially useful as it allows the use of smaller datasets without the usual disadvantages.

When discussing data classification methods, both papers investigate and consider many different methods, but both advocate heavily in favour of the Naive Bayes method of classification. Earlier papers by Pantel and Lin and Dumais, Platt, Heckerman and Sahami also performed a similar comparison and also came to the conclusion (Pantel and Lin, 1998; Dumais, Platt, Heckerman and Sahami, 1998). The very popular work by Graham also advocates the use of Naive Bayes, further cementing Naive Bayes as the preferred classification method.

Blanzieri and Bryl also independently re-iterates many of the points covered by Guzella and Caminhas, such as the use of the bag-of-words model in many other works, and the proliferation of the Naive Bayes model throughout the research area as the machine learning model of choices. These two papers independently giving similar conclusions gives me more confidence in using their work.

My initial research into Spam has been incredibly successful. The field of Spam detection is very fast moving, focussed and motivated by the constant competition between spammers and the developers of spam detection tools. As a result, there is a large amount of research and a culture of competing approaches against each other in the interests of determining the best methods.

The field of email importance can share much of this work, as a machine learning classifier is the core solution to both problems. Specifically, this work has highlighted the Bag-of-words model as a method of generating features, the Information Gain method for selecting features, and the Naive Bayes method for performing the classification. I explore these methods further in my Technical Investigation under the LIWC (section 3.5) and WEKA (section 3.6) sections respectively.

2.4 Email Analysis Datasets

My research above into Spam Detection made it clear just how important a good dataset is for use as the initial training set. It was inevitable then that the lack of a standardised data set would pose significant hindrance for the email classification field. There were a number of datasets to support spam detection, however it was much harder to collect large collections of work emails, for obvious privacy reasons.

This made progress difficult, as with no way for researchers in email classification and linguistic analysis to directly compare methods on the same data, there was no way of determining which methods were objectively better. Indeed, Klimt and Yang state that "most studies [used incredibly small personal email] collections of people working on the projects", not ideal by any means. The small existing datasets could also cause problems,

as they could be unknowingly influenced by non-standard features, potentially skewing results.

This issue was resolved by the introduction of the Enron Dataset in 2004 by Klimt and Yang (Klimt and Yang, 2004). This provided a large, open dataset that could be freely used by researchers to solve many of the issues identified above. The dataset was revolutionary due to its scale and depth, but it would not become apparent until years later just how unique the Enron Dataset was.

The dataset was unique as it was released at a time when privacy on the internet not as prominent as it would later become. Later analysis by the EDRM project would reveal over 10,000 pieces of 'PII" (personally identifiable information) including credit card numbers and dates of birth in the dataset (EDRM Project, 2010). It is a widely held belief that because of this, another dataset like Enron will not be released.

2.5 The Enron Email Corpus

The Enron Dataset was originally collected by FERC (Federal Energy Regulatory Committee) as part of their investigation into the collapse of Enron. They collected the emails of over 150 users, including a large number of managers, consisting of over 500,000 emails in total (Klimt and Yang, 2004).

Since its introduction the corpus has evolved from the original introduced by Klimt and Yang. The dataset has been processed and now exists in many different formats with different modifications, yet all stem from this original.

The original CALO dataset is the closest to the original data collected by FERC (Klimt and Yang, 2004). This dataset provides the data in a file structure with little processing applied to the data, and also removes the attachment data from the dataset. It is worth noting that there has been very little maintenance on this dataset beyond removal of emails where redaction requests have been received. I explore this dataset in detail in Technical Investigation section 3.1.1.

Existing in parallel with the CALO corpus is the dataset provided by the EDRM project (EDRM Project, 2010). This dataset has been actively maintained by the EDRM project and while there is a large amount of overlap between the EDRM and CALO dataset, there exist some subtle differences.

This project was where the large amount of personal information in the dataset was identified, and as a result presents a cleansed version of the dataset with these removed. This dataset also differs from the CALO dataset in that it also has all the attachments, resulting in a much larger size. This dataset is explored in section 3.1.6 of the Technical Investigation.

There are also a number of other smaller corpora that have been created through various projects. The ANLP labelled dataset is one such example, created by the Applied Natural

Processing Language Processing project at the University of California, Berkeley.

This small corpus consists of 1700 manually labelled emails, which offers some different information to the raw data provided by the other datasets. I explore this further in section 3.1.7 of the Technical Investigation.

2.6 Email Classification Features

While Klimt and Yang introduced the dataset that made true comparative work possible, there has been other work that considers features important in emails. I will provide a high level overview of the features here, discussing the explicit features themselves in section of my Technical Investigation.

Indeed, many of the features discussed by Klimt and Yang were first identified in (Kilander, 1996). Admittedly these initial features were little more than the results of a survey requesting users to speculate on features they would use to classify emails, however when combined with Lantz's similar work (Lantz, 1995), using Usenet articles; these began to develop into a strong set of initial features.

By combining studies using two different communication methods, Kilander et al. could be far more confident about the common features that were identified, documented in (Kilander et al., 1997).

Manco et al. built on these two by beginning to actively refine this initial set of features (Manco et al., 2002). Manco et al. began by splitting the features into two broad categories; *Structured* and *Unstructured* features.

2.6.1 Structured Features

Structured features encompass a large number of different, specific pieces of data. These features could be considered metadata, and can be further subdivided into *Categorical Data* and *Numeric Data*. In the context of email, these features are generally metadata, either physical metadata like the Sender, or metadata that needs to be generated, like the number of recipients. It is worth saying that these groups are not exclusive, and it is possible for a feature to be both Categorical and Numeric Data.

Categorical Data

Categorical Data is generally associated with the email header, rather than the body and content of the message. Fields such as "CC:" and "From" are prime examples of Categorical Data. These fields are very useful for classification as they are easily extracted from each individual email, and are strong indicators of classification.

Indeed, Kilander's studies showed showed that users ranked the sender as the second highest

feature of an email for classification purposes (Kilander, 1996), giving excellent evidence as to the importance of Categorical Data.

Numeric Data

Numeric Data is as the name suggests, numerical data that is derived from the data in some way (Manco et al., 2002). This data has varying levels of usefulness, indeed Klimt and Yang even go as far as to state *"So far, every test has found that these features contribute little towards email classification."* I do not fully support this statement, as while I agree that the Numeric data examples mentioned by Klimt and Yang (message size, number of recipients and character counts) are poor features, I believe that this does not necessarily extrapolate to all numeric data.

Indeed Manco et al. explicitly mentions the number of recipients and number of messages received from the same sender, both of which I believe have potential as features. I will tackle this further when discussing individual features in section 3.3 of my Technical Investigation.

2.6.2 Unstructured Features

Unstructured features are typically far less numerous than Structured features, however that does not suggest they are any less useful. On the contrary, Unstructured Data includes far more raw data than any other individual features, so opens itself up for different methods of data analysis. Specifically, Unstructured Data is made up of the textual contents of the email, gathered from the Subject and Body fields.

There are several methods of processing this data, the Bag-of-Words method (also known as the Vector Space Model or a Unigram), N-Gram models (both character and word) and tf-idf (term frequency-inverse document frequency) (Guzella and Caminhas, 2009).

The Bag-of-Words model is relatively simplistic, taking the whole text set and counting the number of occurrences of each word, which is then typically compared with a dictionary to impart more complicated meaning. N-gram takes a sliding window of n-length, and uses it to generate tokens, in character n-gram this takes n characters, word n-gram generates tokens of n words. N-gram then uses these tokens and attempts to predict the next values.

Tf-idf is slightly more complicated, this is a measure of how significant a word is in a given text. The tf-idf score increases proportionally to the number of times it occurs. However, as some words are common than others, this value is then reduced by the frequency of the word in the overall corpus, giving a value for the significance for that value in the individual text.

I discuss these methods in more depth when discussing feature selection in section 3.3 of my Technical Investigation.

2.7 Pre-Processing Methods

In addition to performing dataset specific pre-processing (processing to adapt to unique peculiarities in the dataset, discussed further in section 3.2 of my Technical Investigation), there are a few pre-processing methods that are used commonly to increase the efficacy of Natural Language Processing tasks, specifically those used with the Bag of Words approach (Ahmed and Mithun, 2004).

I touched on these when discussing work in the Spam Detection field in section 2.3.1. This field has some sophisticated methods of pre-processing data, developed to combat increasingly complex spam obfuscation (Manco et al., 2002; Guzella and Caminhas, 2009). Stemming and the Removal of Stop-words improve the efficacy of methods such as the Bag-of-Words approach, giving each feature far more predictive accuracy and improving runtimes.

The specific implementation of these two methods is covered in section 3.2 of my technical investigation.

2.7.1 Stemming

Stemming is the "process of reducing the syntactical variants of the words to their root, mainly eliminating plurals, tenses, gerund forms, prefixes and suffixes." (Manco et al., 2002). As shown by Ahmed and Mithun, stemming results in a much cleaner set of terms to be analysed, and allow the NLP to be much more decisive (Ahmed and Mithun, 2004).

2.7.2 Removal of Stopwords

Stopwords are "words occurring with high frequency in the text of the messages. As a consequence, such terms are not relevant, as they do not discriminate among contents." (Manco et al., 2002) Common conversational terms such as *"and", "to" and "is"* are all examples of common stopwords. When considered, it is almost by definition that the frequency of these words in a document would have little correlation with Importance.

2.8 Relationship Data

When analysing emails we do not have to stop at the individual emails. Indeed, emails are rarely just individual messages, most are just the start of a discussion (referred to as a discussion thread, or thread). This has been touched upon in a number of papers, Klimt and Yang performed some initial threading analysis on the when introducing the Enron Dataset, showing that over 60% of the dataset appears to exist in threads (Klimt and Yang, 2004), however it does not go into any explicit detail on its methods.

Papers by Lewis and Knowles and Yeh and Harnly expand on different methods of threading

greatly, explicitly documenting different methods of generating threads, and comparing their accuracies to determine the most efficient (Lewis and Knowles, 1997; Yeh and Harnly, 2006).

Clearly there is a lot of data to be mined here, and I would hypothesise that messages in a thread are likely to be considered more important than those not, although this is likely to be dependent on a number of factors such as who started the thread, and how the user is related to the thread.

I discuss this in greater detail in section 3.4 of my Technical Investigation.

Chapter 3

Technical Investigation

3.1 Enron Datasets

As previously discussed, the Enron dataset is used throughout the academic community for a wide variety of purposes, and as a result there exist several different versions of the dataset, which I will investigate below.

3.1.1 CALO Corpus

This version of the email corpus (Gervasio and Kaelbling, 2009) is the closest thing that the academic community has to a standardised base data set. This dataset contains 619,446 messages belonging to 158 users (Klimt and Yang, 2004), however as identified by (Klimt and Yang, 2004), some of these are duplicates, and some exist in multiple places (for example, where the sender and recipient are both in the dataset).

However, this version is itself not the original data provided by the FERC (Federal Energy Regulatory Commission), it has had a small number of changes applied. This dataset has been processed by Gervasio and Kaelbling, performing the changes listed below:

- Removal of email records where redaction requests have been received
- Removal of email attachments
- Basic data processing on invalid email addresses

These changes do little to affect the overall integrity of the corpus. There have been two redaction requests for the email records of two employees, however given the number of users in the dataset, the loss of a small number should not have a large impact on the overall results.

This version of the corpus has also removed the specific email attachments, which could result in some loss of useful data. At first glance it appears that only the attachments themselves have been removed. The emails themselves still filename of the attachment, suggesting that there *was* a file attachment, which could have been enough to incorporate the presence of an attachment as a feature. However, upon closer inspection, the files are just the .pst representation of the email themselves, so this is not the case.

It is worth noting that the EDRM version of the dataset *does* have the individual attachments, which I investigate in section 3.1.6.

Finally, the data processing on the invalid email addresses is particularly helpful, converting emails to the format "*user@enron.com*", or "*no_address@enron.com*" where no recipient was found. This is useful in reducing the overhead on the data structure (discussed in section 3.1.2), and should have no negative impact on the data for the purpose I intend.

3.1.2 Data Structure & Representation

The structure of the data itself is also worth mentioning, as it will largely dictate how easy the dataset is to use. The data structure is depicted below in figure 3.1.

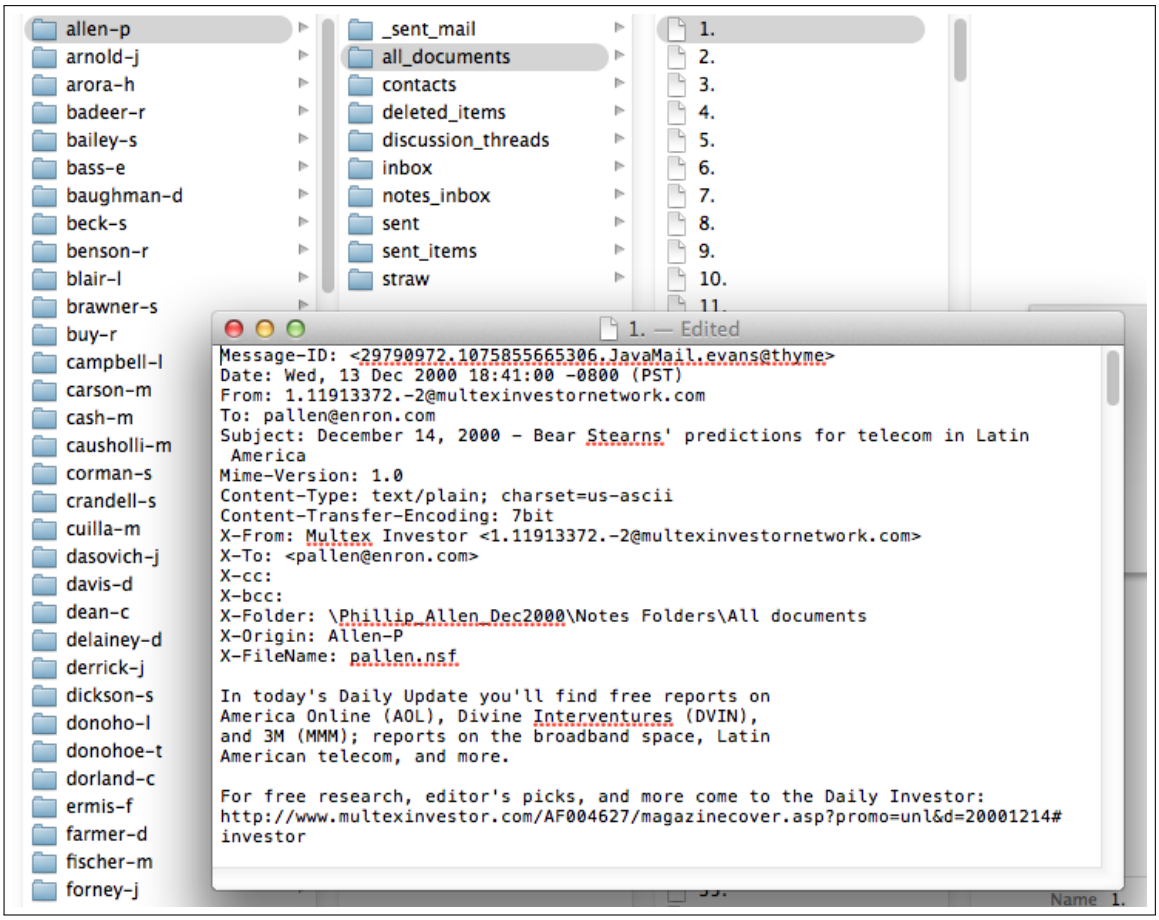

Figure 3.1: Data Structure of the CALO corpus

The structure is a fairly simple and intuitive folder based structure. The emails are organised in the structure User > User Defined Email Folders > Individual email records. The User is the reverse of the users Enron email address, with a dash as a forename/surname delimiter. For example, Phillip Allen (with email address *pallen@enron.com*), has his emails stored in the allen-p folder.

At the next level we have the User Defined Folders. These folders could be incredibly useful for me, as these folders are a very basic form of user data classification, and many I could reason about levels of importance with. While these folder names are not standardised, common themes in these folder names could be very useful.

There is a caveat to these folder names not being standardised; there are still a number

of folders that appear more frequently than others, as users independently use the same folders for common themes. Table 3.1 contains some analysis of the folder names in this dataset.

The emails themselves are plaintext files all named with a number from 1-N, with N being the number of emails in the folder. This is not ideal as it means that the unique identifier for a file is its full file path, rather than a unique filename, which while good for accessing the files (and also for storing the parent folders and user for the file), is not the most user friendly of approaches. A unique filename identifier could be more useful.

Folder Names	Frequency
inbox	137
sent_items	136
deleted_items	135
all_documents	110
discussion_threads	93
sent	90
notes_inbox	82
_sent_mail	80
calendar	72
contacts	48
personal	45
tasks	36
to_do	34

Table 3.1: Common User Defined folders in the Enron Dataset

The table shows the most common folder names in the data set. A large number of the folders appear to be automatically generated by email programs (a view shared by Klimt and Yang (Klimt and Yang, 2004)), however the fact that sent_items, sent and _sent_mail are all present suggests that perhaps there were a number of email programs in use, generating different automatic folders. This should be taken into account when reasoning about the origin of folders.

3.1.3 Further Analysis & Notable data points

Using the tools described in subsection 3.1.4 I performed some further analysis on the CALO dataset to begin identifying folders I believe may be useful for determining importance.

The folders of note in table 3.1 are the personal, tasks and to_do folders. While it is possible that that these folders could be automatically generated like some of the more common folders discussed above, I believe that their lower number of occurrences is indicative that this is not the case.

Tasks and to_do hold promise for having some level of correlation with importance, as an email containing some request for work is likely to be more important to a user than an

average email. It is also worth taking into account that some users may use these folders as methods for controlling workflow. If this is the case then it is possible that these folders contain an uncharacteristically low number of emails (as emails could be moved or deleted when actioned), making them less useful for classification. I will investigate this further in subsection 3.1.4.

The personal folder is another folder that looks to have been added manually. This folder could also be quite useful as it will likely be populated with emails that the user deems important. However, personal emails are also very subjective, and can be on any number of unrelated subjects. As a result, this folder may not be as useful for developing the classifier.

I also looked deeper into dataset, looking for any other obvious indicators of importance. The folders identified in the table stood out due to all containing the keyword important. However, further investigation showed they each occurred only once throughout the whole dataset, and only contained 35 emails between them.

Folder Name	Frequency	Email Count
importantstuff	1	4
important_information	1	4
important_info	1	9
importantemails	1	18

Table 3.2: Notable Folders

3.1.4 Credibility Investigation

After performing further search, it became increasingly apparent that the frequency with which a folder occurs was not enough to reason about the credibility of these folders. As a result I modified my analysis tools to record the number of emails in the folders accessed, with the results shown in table 3.3

3.1.5 Dataset preprocessing

When viewing Table 3.3, the sent folders stick out as an anomaly, as I would expect them to be the most used. To rectify this, created a script to pre-process the dataset, merging the sent message folders into a new separate folder “sent_mail”, giving the data structure shown in Table 3.4. I also performed the file renaming discussed above in section 3.1.2.

Folder Name	Frequency	Email Count
inbox	137	41521
sent_items	136	37931
deleted_items	135	50991
all_documents	110	128103
discussion_threads	93	58609
sent	90	58168
notes_inbox	82	36665
_sent_mail	80	30237
calendar	71	1754
contacts	48	437
personal	43	2642
tasks	36	102
to_do	34	332

Table 3.3: User Defined folders with email counts

Folder Name	Frequency	Email Count
sent_items	151	126336
inbox	137	41521
deleted_items	135	50991
all_documents	110	128103
discussion_threads	93	58609
notes_inbox	82	36665
calendar	71	1754
contacts	48	437
personal	43	2642
tasks	36	102
to_do	34	332

Table 3.4: Modified Folder Structure

3.1.6 EDRM dataset

As mentioned previously in section 2.4, the EDRM dataset is a cleansed version of the Enron Dataset (EDRM Project, 2010). Both the CALO and EDRM datasets share the same source data, meaning much of the email data is similar, yet there are some key differences with the EDRM dataset.

From a practical point of view, the Datasets are very different. The CALO dataset is 700MB in size, and stored in standard file structure (see section 3.1.2. The EDRM dataset on the other hand is over 18GB in size, with the emails stored by user in .PST (Personal Storage Table) format (Cassidy and Westwood-Hill, 2010).

The difference in size and accessibility is dramatic. The .PST format is a Microsoft propri-

etary file format (*Introduction to Outlook Data Files*, 2011) used to store "Messages and other Outlook items". This poses a major problem when trying to access the data, and while Microsoft does publish the specification for the .PST file type, and there are existing tools for interfacing with .PST filetypes, both add significant complexity.

The large size differential also poses a significant problem when attempting to handle the whole dataset. The 700MB CALO dataset is small enough that it can be processed entirely within RAM of a standard desktop computer, something not possible with the 18GB EDRM dataset. The publishers have attempted to at least address this problem, the dataset is provided in 130 separate files containing the .PST files per user, however this is still a considerable overhead to processing.

This dramatic increase in size is not without reason. This is due to the extra data held in the .PST files. Figure 3.2 shows an example of one of the .PST files loaded into Outlook. Notable are the attachments, contacts, threading notifications (indicating the message is in a thread, or "conversation") and message priorities. The attachments are all present as per the originals, being the main contributor to the vast file size. This data has all been lost in the CALO dataset.

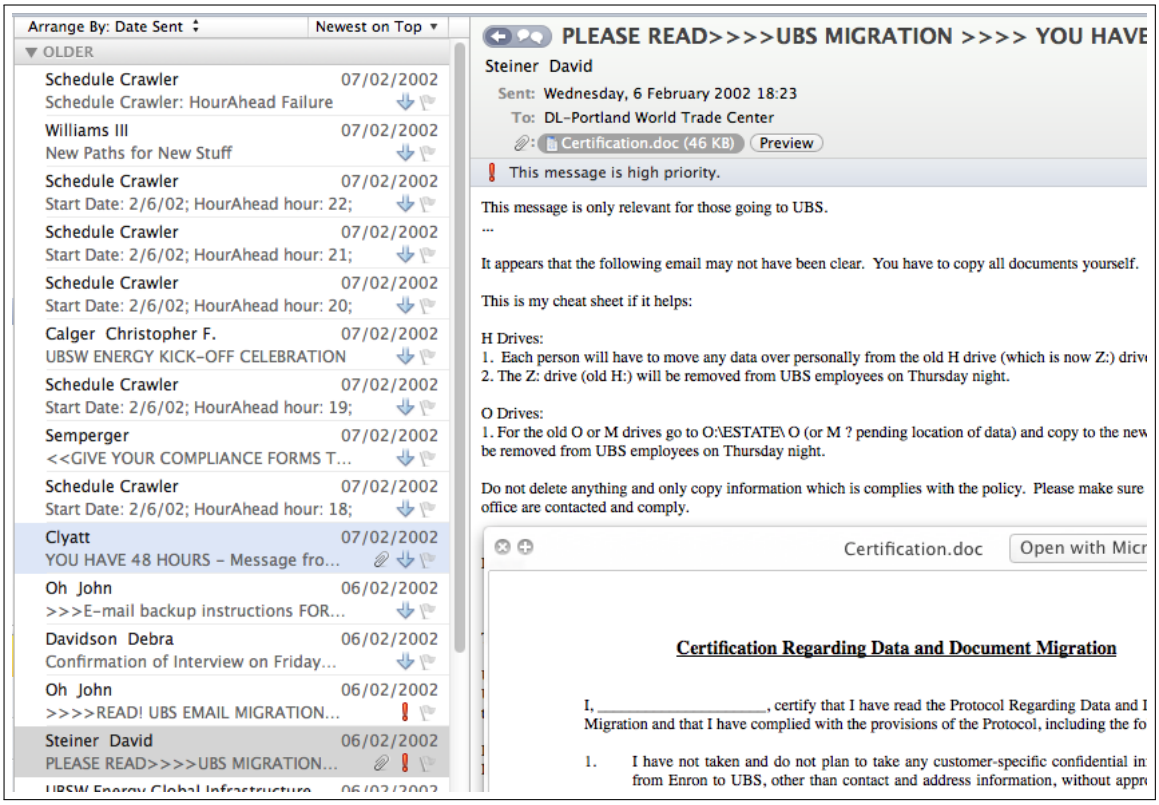

Figure 3.2: Example view of the .PST file in Outlook

The EDRM dataset itself has undergone processing before being released to the public. The

dataset with processed by the NUIX project as documented in (Cassidy and Westwood-Hill, 2010). This processing was centred around cleansing over 10,000 items of Personally Identifiable Information (PII), as below:

- 60 credit card numbers
- 572 social security/national identity numbers
- 292 dates of birth
- 532 messages containing medical or legal information
- 6237 personal contact details
- 3023 resumes containing personal contact information

However while the Nuix project removed this PII, they also added a footer onto every email message, visible in Figure 3.3. This adds another preprocessing challenge onto using the dataset, as this will have to be removed so as not to skew the results of a Bag-of-Words analysis.

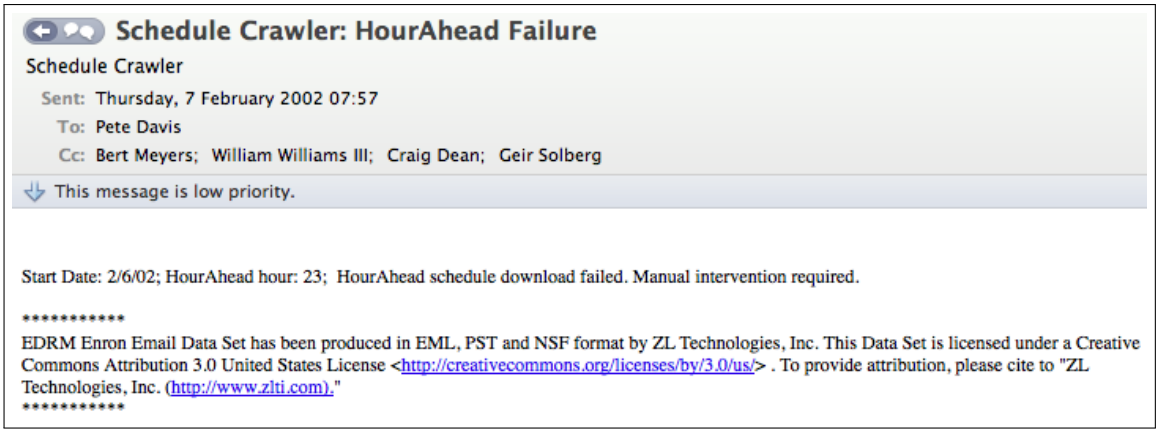

Figure 3.3: Example email with Nuix footer

As established, the EDRM dataset has some advantages over the CALO dataset, however some of these advantages (most notably the presence of attachments) cause some of its key drawbacks.

There has been some work in this area, Dredze has compiled a dataset that applies the EDRM attachment data to the CALO dataset, however it is not openly accessible, which is problematic (Dredze, 2009).

3.1.7 ANLP dataset

The ANLP dataset is a much smaller labelled dataset provided by the UC Berkeley (ANLP Project, 2004). The dataset is a subset of the CALO dataset and consists of around 1700

emails. The emails are provided as they are stored in the CALO dataset (see section 3.1.2), however each email file has been given a uniquely identifying filename.

The category information is provided in the format of an extra .cats file, meaning that for a given email with filename X, there are two files: X.txt and X.cats. X.txt contains the plaintext from the CALO dataset, X.cats contains the labelling data. The labelling data itself was generated by hand, with two separate users classifying each file. Each user was able to apply as many or as few categories to each email as they wanted

There are 4 overarching categories: "*Coarse genre, Included/forwarded information, Primary Topics (if coarse genre 1.1 is selected) and Emotional tone (if not neutral)*". The full set of categories are in Appendix A.1. The ANLP project makes no claims of consistency, comprehensiveness nor generality about these labelings, these are entirely subjective to the categorisers. However, this is useful as it one of the few *openly accessible* labelings of the Enron Dataset.

A mapping file for CALO to the ANLP dataset also exists (although both files will also share the message message ID), supplied by the Enron Data Reconstruction Project (Enron Data Reconstruction Project, 2009).

3.2 Dataset Preprocessing

I have touched on methods of preprocessing the dataset when investigating the work on Spam detection. It is worth mentioning that many methods of dataset preprocessing are specific to different datasets, and are not generalised. However, two generalised methods are that of Stemming and Stopword removal, both of which are used throughout this research area, due to their effectiveness in increasing the efficiency of a whole suite of Natural Language Processing tools.

3.2.1 Stop Word Removal

Stop word Analysis is widely attributed to work by Hans Peter Luhn in 1958 (Williams, 2010), and was considered to have revolutionised work in Natural Language. This process is the process of splitting the text into keywords and non-keywords, and removing the non-keywords.

Practically, this involves tokenising a text (breaking it up into it's component words), and then comparing these tokens with a dictionary of predefined stop words, removing matches. In implementation the choice of this dictionary of stop words can vary (for example, search engines use different stop words to natural language processing). For Natural Language Processing, the widely used toolset is the Natural Language Toolkit, a module for the Python programming language used throughout the research community (Bird, Klein and Loper, 2009).

3.2.2 Stemming

Stemming is the process for reducing all words with the same stem to a common form, first discussed and published by Lovins (Lovins, 1968). The work was publicised by Porter as he provided the first algorithmic implementation in 1980 (Porter, 1980). This algorithm has since been used throughout the research field, translated into many different programming languages and used in many Natural Language Processing projects.

Stemming itself is the process of reducing inflected words to their base form. For example, the words connection, connective connected and connecting are all stemmed from the word connect. There are a number of different implementations, however all should produce the same output. It is notable that the Natural Language Toolkit also supports stemming (Bird et al., 2009).

3.3 Potential Features

Having discussed datasets, I will now go through previous research to attempt to collate a list of potential features that have been discussed in previous papers.

Kilander's studies in 1997 are an excellent starting point for discussing potential features, as they source from a large number of users (Kilander, 1996). The results from Kilander's surveys are visible in Table 3.5. Several of these features are not applicable as the study was focussed on UseNet news, however I will address this when discussing the features in turn.

Senders, Categories of senders and History of senders are all based on users perceptions of the sender, which is difficult to model when looking at as static existing corpus, however it could have relevance in a realtime system. Recipients and Subject is certainly relevant to email, and deserve further investigation. Keyword synonyms are an interesting point as while keywords are not relevant to email, generating synonyms achieves a similar result to stemming. The Language of the text is certainly important, however as there exist no email corpuses with multiple languages, it is difficult to investigate.

Readability and Semantic text analysis are interesting points as these depend on a number of sub-factors, such as the number of long words, the average number of words per sentence and the specific words and phrases. This gives more lean towards performing multiple different analyses on the text itself. The Length of text is discussed with some contention, many users suggested it was important, however the reasons for it being so were varied. One interesting suggestion is the impact of the length of subject on the importance of text, with users suggesting that length of subject would hold more sway on a shorter message. It was also suggested that length should be combined with other metrics.

Date and Time information was defined as being relevant, users suggested that unread email should receive higher priority with time. This specifically is difficult to gather from a static corpus, however it is evident that this could be a useful feature. Relations to other

Feature	Frequency
Keywords	40
Sender	37
Subject	30
Length	21
Date and Time	21
Categories of senders	13
Recipients	11
Relations to other texts	9
Semantic text analysis	6
Keyword synonyms	5
Attachments	5
Language	4
Priority	3
Newsgroup	3
Layout	2
Negative keywords	2
Author's keywords	1
Number of readers	1
History of sender	1

Table 3.5: Kilander's survey of user preferences (Kilander, 1996)

texts such as a reply to a question was regarded as important. While determining replies from a static corpus can be difficult, there has been a large amount of work in this area, so it should certainly be considered.

Attachments, specifically the type and quantity of the attachments were discussed as important, however while the number of attachments is easy to generalise, the type of attachment may be difficult. The priority of a message is certainly relevant, however as it is affixed by the sender this may be slightly misleading. The layout is an odd feature, as this varies based on the users writing style as much as anything else. I suspect this would be subsumed by something like text length.

Kilander summarises by identifying two more general themes seen throughout the responses, identifying keywords which is not immediately applicable to email, could potentially be generated. Another concept is commitment that the email requires of the user, which is a difficult concept to generalise.

Mock performs some additional work classifying using keywords. These keywords are generated from the email, using tf-idf (Mock, 1999). This is excellent as this overcomes one of the main issues with Keywords when applied to email; the fact they need to be generated. This paper proves that they are generating keywords is not only possible, it's relevant.

Whittaker and Sidner identifies that users actually ignore the priority field, given that it is the senders interpretation, and that it is rarely used (Whittaker and Sidner, 1996).

Dabbish et al. performs some relevant in a slightly different field, performing a survey on what users felt was important. This was done with the goal of creating a classifier that predicts a users action on an email, finding that the level of commitment required from the email was a large identifier of its important. There were features concerning the perceived importance for the recipient and sender, along with the urgency of the email contributing to the date/time element (Dabbish et al., 2005). Carvalho and Cohen's work follows in a similar vein, classifying speech acts using a Bag-of-words method (Carvalho and Cohen, 2005).

Guzella and Caminhas, Blanzieri and Bryl and Pantel and Lin all perform work in the spam detection area, where textual analysis is the most important feature. These papers compare many of the different methods of utilising features gathered using the Bag-of-words approach, along with investigating different methods of feature selection(Guzella and Caminhas, 2009; Blanzieri and Bryl, 2008; Pantel and Lin, 1998).

Both Yeh and Harnly along with Lewis and Knowles perform very in-depth studies on email threading, identifying the best methods for determining threads (Yeh and Harnly, 2006; Lewis and Knowles, 1997). It's notable that threading requires a large amount of processing on by itself, I discuss threading further in section 3.4.

In addition to introducing the Enron corpus Klimt and Yang's paper also touches on a number of separate features (Klimt and Yang, 2004). Extending from Manco et al.'s feature classifications they also use the bag-of-words method for generating features. Klimt and Yang also use some Structured features such Senders, Recipients and Subject, and are critical of the numeric features like Text Length, disagreeing with the findings by Manco et al. (Manco et al., 2002). Klimt and Yang also perform some rudimentary threading on the Enron dataset, however this work is eclipsed by the work by Yeh and Harnly.

Manco et al. covers a wide range of features for the purpose of testing various feature selection algorithms in order to build a fully flexible classifier able to deal with a variety of features which may more may not correlate with importance.

Having touched on these papers previously in the literature review, I have assembled Table 3.6 that summarises each of the potential features or feature categories for use in my classifier. This gives me a valuable start point, however many of these features identified here, specifically those concerning unstructured information will need to be generated or processed in some manner. I will first investigate the methods for generating these before providing a summary of the features I will use going forward. Specifically, the features that need further processing are the Reply/Threading (see section and the Text Analysis.

3.4 Email Threading

As discussed in the Literature Review (section 2.8), and again above, email threading is a feature that has high potential as a classification feature. Klimt and Yang were not the first to suggest it's value (Klimt and Yang, 2004), indeed they themselves refer to previous

Feature	Discussed in
Recipients	(Kilander, 1996; Dabbish et al., 2005; Klimt and Yang, 2004; Manco et al., 2002)
Subject	(Kilander, 1996; Klimt and Yang, 2004)
Text analysis	(Kilander, 1996; Dabbish et al., 2005; Guzella and Caminhas, 2009; Blanzieri and Bryl, 2008; Pantel and Lin, 1998; Dumais et al., 1998; Klimt and Yang, 2004)
Text length	(Kilander, 1996; Klimt and Yang, 2004; Manco et al., 2002)
Subject length	(Kilander, 1996)
Date/Time	(Kilander, 1996; Dabbish et al., 2005; Manco et al., 2002)
Reply/Threading	(Kilander, 1996; Klimt and Yang, 2004; Lewis and Knowles, 1997; Yeh and Harnly, 2006)
Number of Attachments	(Kilander, 1996)
Type of Attachments	(Kilander, 1996; Manco et al., 2002)
Priority	(Kilander, 1996; Whittaker and Sidner, 1996)
Keywords	(Kilander, 1996; Mock, 1999; Manco et al., 2002)
Commitment	(Kilander, 1996; Dabbish et al., 2005; Carvalho and Cohen, 2005)
Senders	(Kilander, 1996; Dabbish et al., 2005; Klimt and Yang, 2004; Manco et al., 2002)

Table 3.6: A table collating the notable features in previous research

work by Lewis and Knowles in approaching thread analysis as a language processing task (Lewis and Knowles, 1997). Here I will investigate the specific methods various studies have used in order to assess the difficulty in performing threading, in order to inform my decision on whether to include it as a feature.

3.4.1 Threading Electronic Mail: Lewis and Knowles (Lewis and Knowles, 1997)

Lewis and Knowles performed an exploratory experiment that serves as an excellent jumping off point for this topic. Specifically, they used the Subject of each email message, along with the body text as the key features. Some rudimentary text matching was then performed to further subdivide the body text into quoted and unquoted text along with other simple processing.

Each field in a given email was compared against its respective counterpart in all potential parents, along with body features also being cross-checked. It is notable that all 5 metrics were vastly superior to random parent selection.

Their findings suggested that the Subject field was a good indicator of the thread that a given email belonged to, but fell down when attempting to pick out more complicated thread behaviour such as sub-discussions, and where exactly in the thread the message lies.

Quoted text gives a much better overall indication by providing similar thread predictive power, while also performing much better at identifying specific topic responses. However this feature is dependent on user behaviour when replying (some users may respond in-line, others may respond in one block).

Non-quoted text provides weaker correlation, however this is to be expected as while it is likely that there is common language in a reply, (due to strong lexical cohesion between a message and it's reply (Halliday and Hasan, 1976)) these will not match as exactly as quoted text or subject headers due to the presence of novel words used to construct a meaningful reply.

Quoted text compared against unquoted parent text provided the best indicator of threading, which is unsurprising as one would expect that barring header information in the child, both should be exact copies (assuming the reply has not been edited in any way). As noted above, user behaviour in method of email reply does affect this. The Subject field could be useful here, as it is less likely to be tampered with.

It is also noted that improving the text processing could have had a large positive impact on the results Subject; if the "RE:" tags had been stripped out the text matching would have been much more exact. In addition to advanced text processing, the study also suggests some other features that could be used to further improve accuracy, including; time, authorship info, categorisation and identification of siblings

In terms of relevance to my project, Lewis and Knowles did have a simpler problem than that posed by attempting to thread the Enron dataset. In this study it is already known whether an email is part of a thread, and in the case that an email is part of a thread, it is also the case that this email's parent or children are also present in the dataset. These two assumptions cannot be applied to the Enron dataset, so any classification will have to also make these decisions; introducing two more sources of error.

Finally, it is also worth noting that the study highly recommends using machine learning techniques to use many different features in the name of achieving the highest accuracy possible.

3.4.2 Email Thread Reassembly Using Similarity Matching: Yeh and Harnly (Yeh and Harnly, 2006)

This paper is an almost direct follow on from the paper discussed above (Lewis and Knowles, 1997). Yeh and Harnly perform many of the improvements identified by Lewis and Knowles, in addition to investigating some other methods of identifying threads. However, the most important feature of this paper is that the use of the Enron Dataset, making it very relevant to the study I am performing.

In terms of the specific improvements over the work by Lewis and Knowles, this work performs far more sophisticated text processing. As a result, the Subject matching feature is far stronger (as exact matches are far more likely). The advanced text processing also

extended to the detection of quoted text, and this work is able to identify nested quotations. Like the initial work, neither can deal with in-line replies, however this paper does note that this method of replying is at least minimal in the Enron dataset, reducing its relevance for my project.

This paper also implements some date/time processing, using it to bound some of their searches, making their threading algorithm far more efficient. Some post-processing also revealed the reply timing patterns, noting that almost all replies occur within 14 days (See Figure 3 in (Yeh and Harnly, 2006)).

Beyond building on previous work the paper also introduced some new methods for thread tracking. This method revolves around the use of the "Thread-Index" header defined in the poorly documented Microsoft Exchange Protocol. This field is intended to be read by compatible email clients (such as a Microsoft Outlook) to enable the automatic identification of email threads. The problem is immediately obvious however, as the presence of this header is obviously dependent on the users using Microsoft Outlook, something that is not guaranteed.

To compound this issue, the documentation did not include any mention of how to encode or decode the values of this field, however by performing some investigation Yeh and Harnly were able to deduce enough about the field's behaviour to use it for its original purpose.

The advanced text processing is very well documented, the paper goes into great depth to define both the algorithms used to tie all of a given users email addresses together, along with the advanced reply and quotation extraction. In the second case the paper also provides the dictionary of reply splitters; a key resource when attempting this sort of work.

3.4.3 The Enron Corpus: Klimt and Yang (Klimt and Yang, 2004)

Klimt and Yang's thread analysis on the Enron dataset is not particularly sophisticated, certainly not when compared with the two papers discussed above. To achieve their threading they compared subjects and user addresses, a relatively simple approach that Klimt and Yang do not guarantee the quality of. They do note that the Enron Dataset does not possess any great number of emails with the "in-reply-to" header which aided Lewis and Knowles's work.

thread size:	2	3	4	5	6	7	8	9	10	(10-20]	(20-30]	(30-40]	(40-50]	51+
# of threads:	16736	4782	3049	1282	879	903	378	214	178	1260	209	79	54	88

Table 3.7: Thread Sizes in the Enron Dataset

Nevertheless, this analysis showed that over 60% of the emails in the Enron corpus are in threads, amounting to 123,501 emails in 30,091 non-trivial threads (Table 3.7 shows the distribution). This work does also identify one of the main problems with using the subject method - the fact that one of the strongest threads actually turns out to be a spam message.

3.5 LIWC

When discussing methods of analysing text split using the Bag-of-Words method, LIWC is one of the leading tools in text analysis (Pennebaker, Chung, Ireland, Gonzales and Booth, 2007). At its core LIWC is a set of dictionaries that are used to compare against a set of target words. These dictionaries have been carefully crafted by the authors using their extensive psychology backgrounds over many years, giving leave to a set of over 80 categories (LIWC Project, 2007).

These categories are useful as they give a number of different measures, from Linguistic processes to Psychological processes and personal concerns, data which is otherwise difficult to discern from an email in an automated manner.

3.6 WEKA

WEKA is a massively popular machine learning tool in the natural language processing field (Hall, Frank, Holmes, Pfahringer, Reutemann and Witten, 2009). The tool provides an implementation of a number of the most common machine learning, data exploration and classification tools, along with a number of tools for preprocessing data and visualising result. The project is entirely open source, and accessible to anyone.

3.7 Research Summary

Given this research, I believe I am now in a position to start reasoning more confidently about what features, datasets and tools I want to use going forward in my project, allowing me to inform my requirements and design.

The features are almost entirely correlated with the ones identified by Kilander's study in some way or another, however there has been a large amount of work in all those areas in the time since they've been introduced.

Based on my collated table of features (Table 3.6) I have decided to concentrate on:

1. Number & Type of Recipients (To, Cc, Bcc etc)
2. Text Analysis using LIWC
3. Text Length
4. Date/Time
5. Threading

These features offer what I believe to be a good balance of predictive importance with computational complexity. However, these features are all statistical measures, for which

I lack a training set. With that in mind I intend to also use the ANLP category mapping dataset, to construct my own subjective interpretation of which of those categories I feel are most important. My collated table of ANLP categories is visible in Appendix A.2.

Due to its openness, size and ease of access I will use the CALO Enron Corpus. I will use the LIWC categories to provide analysis on my text, and collate these features together in WEKA to perform the classification.

Chapter 4

Requirements

4.1 Overview

Given the research and technical investigation performed in chapters 2 and 3 respectively I can now begin specifying the components of my system more accurately. At the high level there are four key components in a classifier, as follows:

1. Email Dataset
2. Dataset Parser
3. Processor
4. Machine Learning tool

These elements are all required to build a full classifier, the absence of any of these will cause the system to fail. That is not to say that these components need necessarily be physically independent from each other, but for the purposes of Requirements specification I will define these elements separately (This will be discussed further in the Design and Implementation chapters).

In the Research Summary (Technical Investigations section 3.7, I identified the datasets and tools I am going to use in the project. Specifically the datasets I am going to use are the CALO and ANLP datasets (see sections 3.1.1 and 3.1.7 respectively). The dataset parser and processor are going to mostly built by me, incorporating a LIWC script built on initial work by Simon Jones. The machine learning tool I am going to use is WEKA, discussed in section 3.6.

4.2 Functional Requirements

Requirement priority is indicated by the use of the words must, should and may.

4.2.1 Email Dataset

These requirements are a loose fitting structure that ensure enough is known about the dataset, it's sources and reliability before it is used in the study. They aim to restrict unknown datasets without enforcing too many other restrictions.

D1 The Email Dataset must be investigated to analyse its applicability as follows:

- **D1.1** The source of the Dataset must be known.
- **D1.2** The context the Dataset was gathered under must be known.
- **D1.3** A discussion of the impact of the context on the Dataset should occur.
- **D1.4** A discussion of the Dataset's use in other work should be documented.

D1.5 The encodings of the emails should be identified.

D2 All modifications to the Dataset should be investigated as follows:

D2.1 All modifications made to the dataset by previous users must be documented.

D2.2 A discussion to discern the impact on the Dataset integrity made by these modifications must occur.

D2.3 Any modifications the user makes must be documented.

D2.4 A justification of these changes should occur.

D2.5 A discussion on the impact on Dataset integrity as a result of these modifications should occur.

D3 Efforts must be made to preserve user privacy where sensitive data is used.

4.2.2 Parser

These requirements cover the interface between the Dataset and the system.

Pa1 The Parser must traverse the whole Dataset.

Pa2 The Parser must attempt to import all data.

Pa3 The Parser must attempt to store all values.

Pa4 The Parser must handle different character encodings.

Pa5 The Parser must store files in appropriate datatypes.

Pa5.1 The Parser must attempt to coerce data to the correct filetype. See requirement Pa8 for details.

Pa6 The Parser must store the data in a manner that allows easy data access.

Pa7 The Parser must apply a unique identifier to each email.

Pa8 The Parser must extract the following data from each email. Conditions must be followed:

Pa8.1 Message-ID

Pa8.2 From

Pa8.3 To

Pa8.4 Subject

Pa8.5 Cc

Pa8.6 Bcc

Pa8.7 Date/Time (Must be stored as Date/Time type not string)

Pa8.8 Body

4.2.3 Processor

The processor has two purposes; to filter and refine the data that has already been imported and to generate new features using the imported data.

Pr1 The Processor must generate the following Numeric features:

- **Pr1.1** Number of To's
- **Pr1.2** Number of Cc's
- **Pr1.3** Number of Bcc's
- **Pr1.4** Number of Recipients
- **Pr1.5** Length of Email Body

Pr2 The Processor must perform a LIWC analysis on the message body.

- **Pr2.1** The Processor may perform stemming on the data prior to analysis.
- **Pr2.2** The Processor may perform stop word removal on the data prior to analysis.
- **Pr2.3** The Processor may perform punctuation removal on the data prior to analysis.

Pr3 The Processor must attempt to determine email replies using threading:

- **Pr3.1** The Processor should attempt to strip "RE" and "FW" tags from the Subject prior to analysis.
- **Pr3.2** The Processor must attempt thread detection using subject matching.
- **Pr3.3** The Processor may attempt thread detection using unquoted/quoted text analysis (see section 3.4).

Pr4 The Processor must output data in a format compatible with WEKA.

Pr5 The Processor must normalise the data between 0 and 1.

4.2.4 Feature Selection & Classifier

C1 All features must be initially loaded into WEKA.

C2 The set of features must be reduced using a method of Feature Selection.

- **C2.1** The method of Feature Reduction should be documented.

C3 The Classification method used must be documented.

4.3 Non-Functional Requirements

4.3.1 System

NF1 The system must make efficient use of computer resources.

NF2 The system should provide the user with progress feedback.

NF3 The system must be modular to support future development.

NF4 The system must support development in an Open Source Manner.

NF5 The system must have high performance.

4.4 Technologies and Resources

A variety of technologies and resources are necessary for the completion of this project.

4.4.1 Python

The Python programming language is a flexible and powerful programming language. It offers a number of desirable features, it is lightweight and its dynamic typing and type coercion are helpful when building a flexible data parser. Python has excellent built in support for reading and writing to files and traversing file structures; which is very helpful given that both the CALO and ANLP (see sections 3.1.1 and 3.1.7) datasets are stored in a large number of separate text files.

Python also has built in modules for writing to .csv files, which can be used as WEKA input, and also has an email module for handling standard email headers. In addition to all these built in modules, there also exists the Natural Language Toolkit third party module (see section 3.2) that contains a number of natural language processing methods such as stemming and stop word removal. I also posses a script for performing LIWC analysis using Python, using the dictionaries stored in an external JSON file.

Python will be used to performing all coding tasks in the project, creating the parser and processor as defined above and discussed further in the design (see chapter 5).

4.4.2 WEKA

WEKA is a a widely used machine learning tool that provides a whole suite of tools to support Natural Language processing (Hall et al., 2009). The tool accepts inputs as .csv's among other filetypes, and supports visualisation of the dataset. WEKA also provides a number of different methods of Feature Reduction, allowing the initial set of features to be reduced to a subset of the features most correlated with a given predictor. WEKA also

provides support for a number of different classification methods, along with a result viewer for visualising classifications.

WEKA will be used for creating all the classifiers outlined here, using the feature selection and classification tools.

4.4.3 LIWC Dictionaries

The LIWC dictionaries are a set of dictionaries for mapping bodies of text onto different psychological processes and emotions, discussed further in section 3.5 of my Technical Investigation.

I will use these to generate a large number of text analysis features as inputs for the classifier.

4.4.4 Enron Datasets

The Enron Datasets are datasets that have been collected from the company Enron after their collapse and bankruptcy in 2001. I use two of these datasets in the project, one being a labelled subset of the other. The CALO dataset is a dataset of over 500,000 emails from around 150 users. The ANLP dataset is a subset of 1700 of these that have been labelled with a variety of categories. These are discussed further in sections 3.1.1 and 3.1.7 of my Technical Investigation.

I will use these datasets as the test data for my project, extracting features from them and using them as both the training and tests sets for my classifier.

4.4.5 Sublime Text

Sublime Text is a text editor with support for Python that I will use as my development environment for the duration of my project.

4.4.6 Git & Github

Git is a revision and source control tool for developing software. It will allow me to rollback changes should that be necessary.

I will host my code on Github when completed, achieving my Open Source requirement (requirement S4) and providing the development platform for anyone performing similar work or wanting to continue with some of the future work I define.

Chapter 5

Design

5.1 Overview

I define the high level architecture of the system in section 5.2, decomposing the system into a set of core modules. In sections 5.3 and 5.4 I define the structure of the Parser and Processor modules with greater granularity, defining where specific requirements are covered (Requirements defined in chapter 4) and and describing the interfaces between the modules.

I describe and justify my design choices in section 5.5.

5.2 High Level Architecture

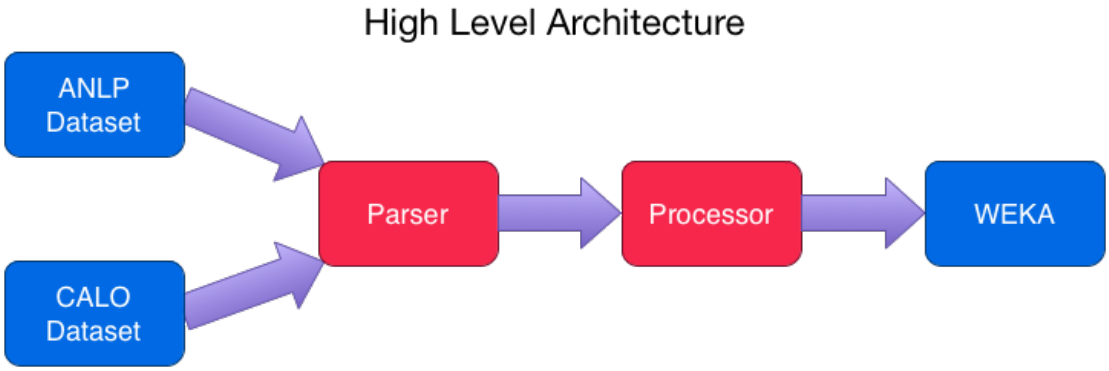

Figure 5.1: High Level Architecture

Figure 5.1 shows the high level architecture of the system, as defined in the Requirements (see chapter 4). The Datasets have a defined file structure discussed in section 3.1, and WEKA takes a .CSV as input.

The design of the Parser and Processor are defined in greater detail in sections 5.3 and 5.4 respectively.

5.3 Parser

The Parser structure is visible in Figure 5.2.

At the high level, the Parser takes a Dataset as input, extracts the relevant fields from the emails, performs some rudimentary processing and dataset storing, before writing that data to a Portable Data Structure as input for the Processor.

The Parser is required to traverse both datasets, which do have defined differences in structure. The ANLP dataset has the same email structure as CALO, however there are

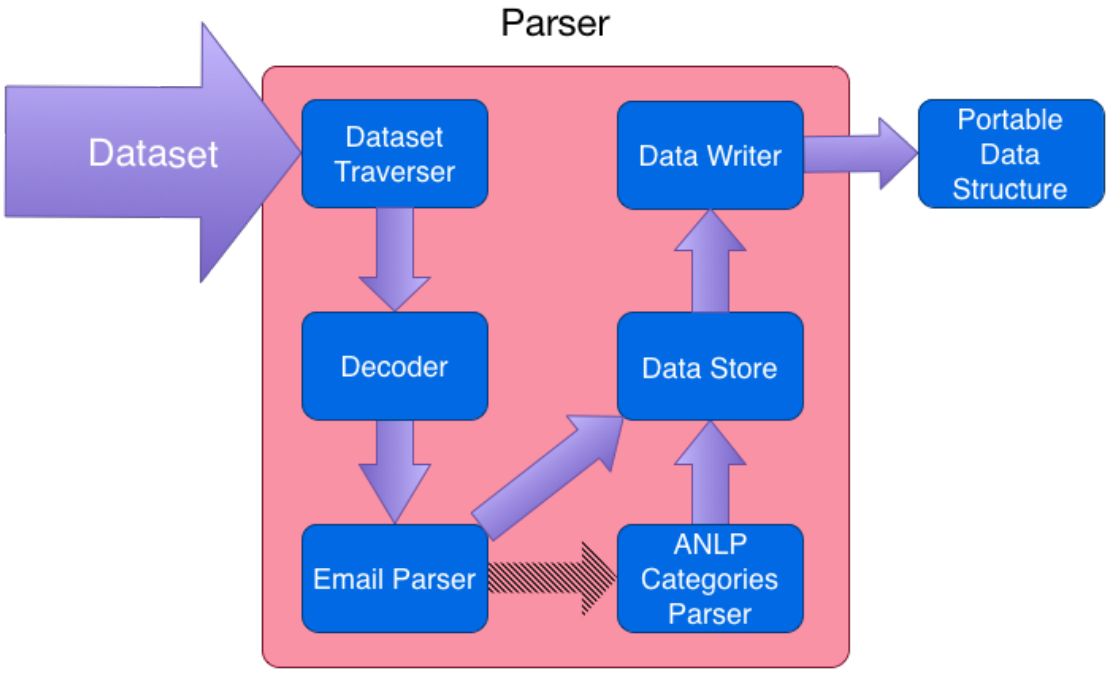

Figure 5.2: Parser Architecture

also the ANLP category files, containing the categories of the ANLP dataset.

The Dataset is input to the Dataset Traverser, which will loop over the whole Dataset attempting to import all the emails as their plaintext, meeting requirements Pa1 and Pa2. The Dataset Traverser is also responsible for applying a unique identifier to each email, meeting requirement Pa7. This Dataset is then passed to the Decoder.

The Decoder is responsible for decoding the dataset. Decoding is the process of processing the email character encodings, turning them into plaintext as per requirement Pa4. This plaintext is then passed to the Email Parser.

The Email Parser will then attempt to parse the plaintext, extracting the data as per requirement Pa8. The specific data to be parsed is stated in requirements Pa8.1-8.8. The Email Parser is then responsible for storing these files in appropriate datatypes, meeting requirement Pa5. These Datatypes are then passed to the Data Store for storage.

If the Dataset is the ANLP dataset the Parser will also parse the contents of the ANLP Categories file. The results are parsed in the same way as the Email Parser. This data is then passed to the Data Store.

The Data Store is responsible for storing all the values, in a manner that allows easy data access, meeting requirements Pa3 and Pa6. Specifically, it is responsible for storing all the data relating to a given email under its unique email identifier. This newly assembled

Dataset is then passed to the Data Writer

The Data Writer is then responsible for writing the contents of the Data Store to a Portable Data Structure compatible with the Processor. The reasoning behind writing to the Portable Data Structure is discussed in 5.5.

5.4 Processor

The Processor data structure is visible in Figure 5.3.

The Processor takes input from the Parser in the format of a Portable data structure, the Portable Data Reader is responsible for parsing the contents of this Portable Data Structure into a more memory efficient local data structure. This local data structure is then passed to the Selection & Processing tool.

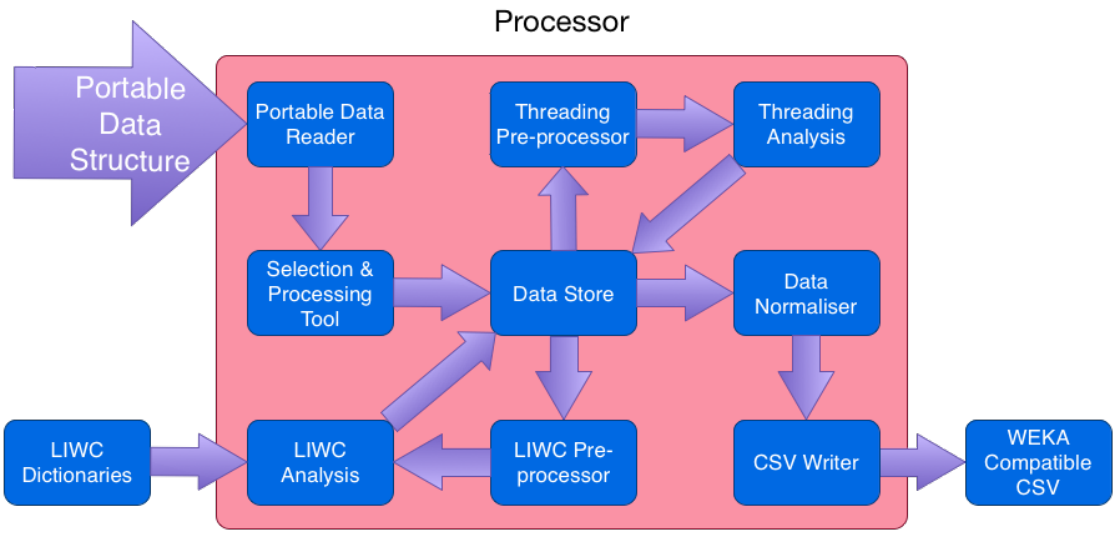

Figure 5.3: Processor Architecture

The Selection & Processing Tool holds the rules for constructing the dataset, pruning the files that do meet the selection criteria, removing them from memory, contributing towards requirement NF1. The pruned files are then processed in order to meet requirement Pr1, specifically producing the features defined in requirements Pr1.1-Pr1.5. The Processing Tool is also responsible for processing the ANLP data previously imported, where relevant. This processed dataset is then passed to become the Data Store.

The Data Store will then pass the relevant parts of dataset onto to the Threading and LIWC Pre-processors, who will perform pre-processing on the data subsets.

The Threading Pre-processor tool will perform pre-processing on the email subject field, while optionally also passing the body. As per requirement Pr3.1, RE and FW tags will be

removed from the subject here. The pre-processor will then pass this data to the Threading Analysis tool.

The Threading Analysis tool is where the thread detection will take place. As per requirement Pr3.2, the Threading Analysis tool must perform thread detection by Subject matching at a minimum. The Threading Analysis tool may optionally attempt to perform more complex thread detection by performing unquoted/quoted text analysis, as per requirement Pr3.3. The output of this Threading Analysis is then passed back to the Data Store, to be appended each emails features.

The LIWC Pre-processor may perform a number of pre-processing methods on the email body. As per requirements Pr2.1-Pr2.3, the LIWC Pre-processor may perform optional stemming, stop word removal and/or punctuation removal. Regardless of whether this processing occurs or not, the data is then passed on to the LIWC Analysis tool.

The LIWC Analysis tool, will compare the processed email body against the LIWC dictionaries, as per requirement Pr2. The LIWC dictionaries are read in from a Portable Data Structure. The output of this is a set of 81 LIWC features, which are passed back to the Data Store to be appended to each emails features.

This newly grown set of features is then passed to the Data Normaliser. The Normaliser will then normalise all features between 0 and 1, as per requirement Pr5. This normalised data is then passed to the CSV writer.

The CSV writer then writes these normalised features to a WEKA compatible CSV, achieving requirement Pr4.

5.5 Design Justification

My Non-Functional Requirements (see section 4.3 have influenced my design at all most every stage, in addition to my aim of developing a platform to allow others to use the tools I develop to perform their own classifications.

In particular my requirements NF1 “The system must make efficient use of computer resources” and NF5 “The system must have high performance” have been the most influential. These two requirements require me to create a design that is as efficient as possible, while balancing that with the ease of performing active development on the tools.

The problem stems from the sheer size of the dataset. Over 500,000 emails are all stored in their own individual text files, and I intend to extract upwards of 80 features from each email.

As a result of this, I have defined two key principles to drive my design:

1. Reduce Hard Disk reads and writes as much as possible.
2. Release Memory when no longer in use.

Given these two principles, my design seems immediately counter intuitive; reading and writing to a portable data structure between the Parser and the Processor introduces a large read/right bottleneck to the system.

This design choice was entirely deliberate; while speed and efficiency are a main goal of the system, the context of system use must also be considered. It is certainly true that for a user with just the dataset and my tools, it would be far faster to run the whole system in series, not writing to the Portable Data Structure.

Take the example of a user using my toolset without the Portable Data Structure read-/write. The user may make some tweaks to my processor, and run the system. Parsing the dataset can take upwards of 4 hours, if the user introduces a small error into the Processor, all of the data from the Parser is lost.

In summary, splitting the system into these modular components makes the system far more flexible to a developer actively developing any components. It also worth considering the nature of the Dataset and the Parser. Assuming the Dataset will not change, the Parser will be reading the same files and performing the exact same computation every time it is run. Accepting a read/write overhead is the lesser of two evils when the Parser is not being actively developed or the Dataset does not change, and the Parser output is required any more than once.

Chapter 6

Implementation

6.1 Design Limitations

The system defined in chapter 5 makes some attempt to address the performance issues, however it does fail to address many other bottlenecks in the system.

The design also decomposes the system into logical groupings of functionality, however this results in data being carried through the system longer than it arguably needs to if processing were performed at an earlier stage. Given this, I will discuss the changes to the design structure in turn below.

The general principle I followed was if something was being generated multiple times, but was performing the same calculations every time then it should be de-coupled into it's own external module.

It is worth noting that while I de-coupled many components, I promoted code reuse through the use of Python's built in module functions.

6.2 LIWC Analysis

The LIWC analysis was the first module which I identified to split off into its own module. When first attempting to implement the system as per the design, the LIWC Analysis would take upward of 4 hours at a time, and I ran into the exact problems that I attempted to avoid in section 5.5 of my Design, albeit in a different part of the system..

I initially attempted to apply the same solution that I applied to the Email Parser in my design 5.3, writing data to the hard drive as a Portable Data Structure. However, attempting to import another large data structure into memory on top of the parsed emails, and then attempting to perform other processing caused system performance to drop dramatically.

In the end I reached a compromise between the two points, the LIWC data was extracted and processed entirely separately from the Email Parser (although there was a small amount of code reuse). Figure 6.3 shows the overall structure of the new Implementation.

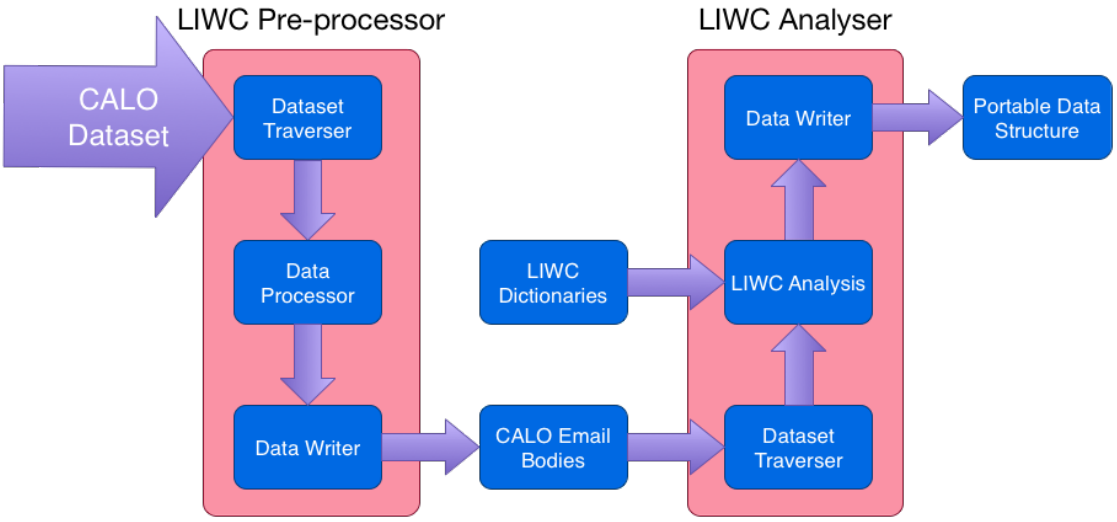

Figure 6.1: LIWC Implementation

The LIWC pre-processor script is used to reduce the memory overhead, rather than opening the whole email and parsing the whole thing to extract the bodies (as would be the case if the emails were fed straight into the LIWC tool), the Pre-processor traverses the whole CALO dataset, parses each email recording the body and the emails unique identifier. The Pre-processor then writes a new to the output folder with the same filename but containing only the email's body in plaintext. This set of files is then used as the input by the LIWC Analyser tool.

Once the LIWC input has been generated, the LIWC analysis can then be run. The only way this has changed from the design is how the data is input, the rest of the functionality matches the design.

This new method is very future proof, as splitting the pre-processing stage allows the pre-processing methods to be changed very easily, supporting requirements Pr2.1-2.3.

6.3 Email Threading

Threading Analysis was easily the most computationally complex and memory intensive process performed. The method of determining Email Threading as per requirement Pr3.2 was to use Subject Matching. This is a relatively simple process, whereby the Threader will iterate through all the emails after a given email, attempting to match the subjects, within a given time period. As the Pre-Processor has removed the Reply and Forward tags from subject lines as per Pr3.1, any replies or forwarded messages should have the same subject names.

In order to do this efficiently (to avoid checking the entire dataset for every single email), the

dataset must first be sorted by date from earliest to latest, allowing the subject comparison tool to iterate down the list until the time period is breached. However, sorting a structure with over 500,000 elements is incredibly resource intensive, necessitating it being moved to a part of the code with lower resource requirements.

As a result the Threading Analysis was moved into the CALO Parser, along with other processing, defined in section 6.5 and shown in Figure 6.2.

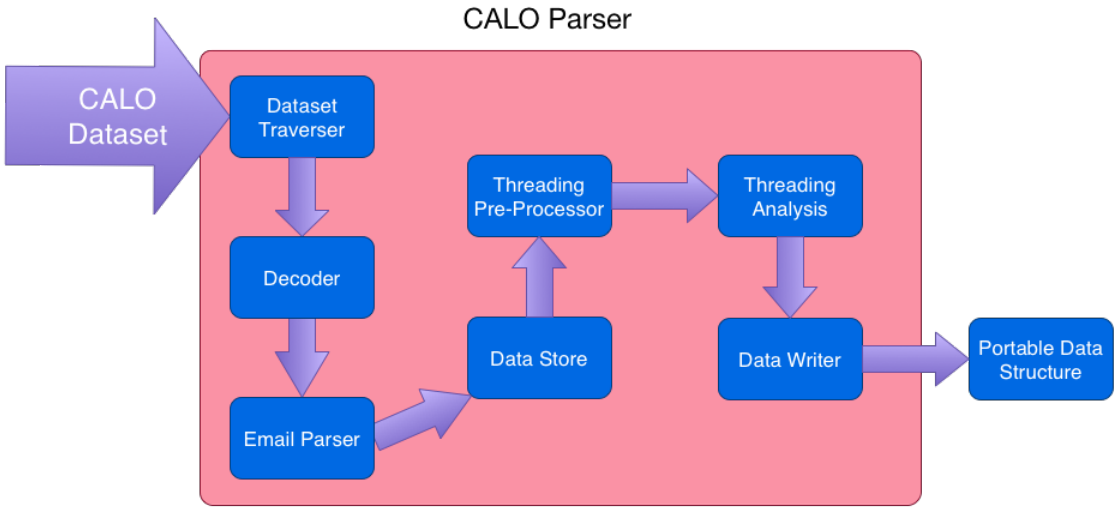

Figure 6.2: CALO Parser Implementation

6.4 The CALO Parser

The Parser itself has been split into two sections, the CALO Parser and the ANLP Parser. The reason for splitting out the ANLP parser has been documented in section 6.6. As a result of this change, the role of the module's individual module parts have changed slightly. The functionality of the Dataset Traverser, Decoder, Email Parser and Data Store have not changed, however the Email Parser no longer has to check for each email's presence in the ANLP dataset, a not insignificant amount of computation given the 500,000+ emails, of which only 1700 are in the ANLP dataset.

The Threading Pre-Processor step has much more functionality in this implementation. It is now responsible for more memory management tasks in order to improve performance and support the Threading Analysis. The Threading Pre-Processor is now responsible for sorting the dataset by date in order to dramatically increase the speed of the Threading Analysis tool (by dramatically limiting the search space from all 500,000 emails down to just 1 more than the date period at worst). The Threading Pre-Processor is also responsible for freeing the memory holding the unsorted copy of the dataset, after the sorted dataset has been created.

6.5 The ANLP Parser

The ANLP Parser has been split out for two reasons, firstly due to the computational overhead of performing checking on a pair of datasets with such a great size disparity, and secondly to allow the regeneration of the ANLP categories or importance ratings without the overhead of regenerating the dataset of all other emails too.

The Dataset Traverser and Category Parser retain the same functionality as the Design (Where the ANLP Category Parser has become the Category Parser). The Category Processor has inherited some functionality from the Processor, now importance ratings are calculated in the ANLP parser. The Data Store and Data Writer share the same functionality as in the design, albeit they now have to handle slightly more data as generated by the Category Processor.

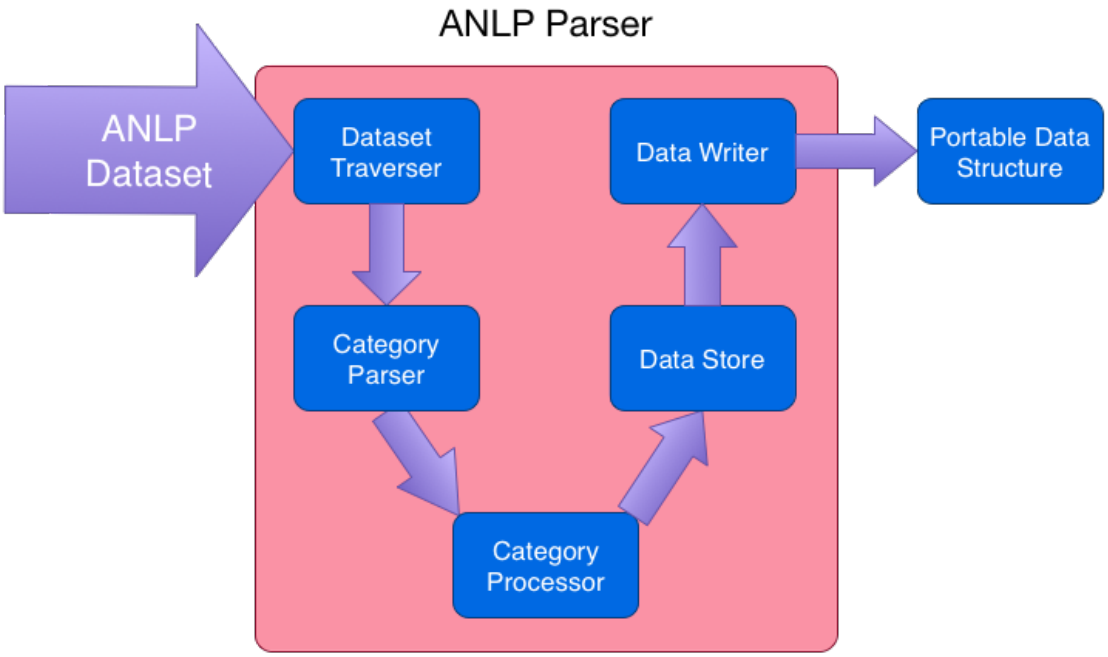

Figure 6.3: ANLP Parser Implementation

6.6 Processor

By necessity, much of the actual functionality and processing has been moved out of the Processor. Instead, the Processor now acts as much a combinator. Where before the actual processing was performed in the Processor it has been moved to other modules, and replaced with an interface to that module.

Specifically, both the computationally heavy Threading Analysis and LIWC Analysis have dropped out into their own modules (see sections 6.3 and 6.4). The LIWC analysis is performed in it's own module and imported in through a Portable Data Structure. The Email Threading Data has been merged with the CALO Parser, so is now read in in the same manner as the email data.

The ANLP data has been de-coupled from the email data, so now rather than the Data Processor checking for presence of ANLP data on the whole dataset, the Processor can apply the ANLP directly to the 1700 relevant emails, a small speed gain. The processing for the ANLP dataset processing has also been moved into the ANLP parser, again giving a small speed increase and decrease in complexity.

Otherwise the Data Store, Data Normaliser and CSV Writer mimic follow the design. While there are now several Data Readers, these all reuse the same code.

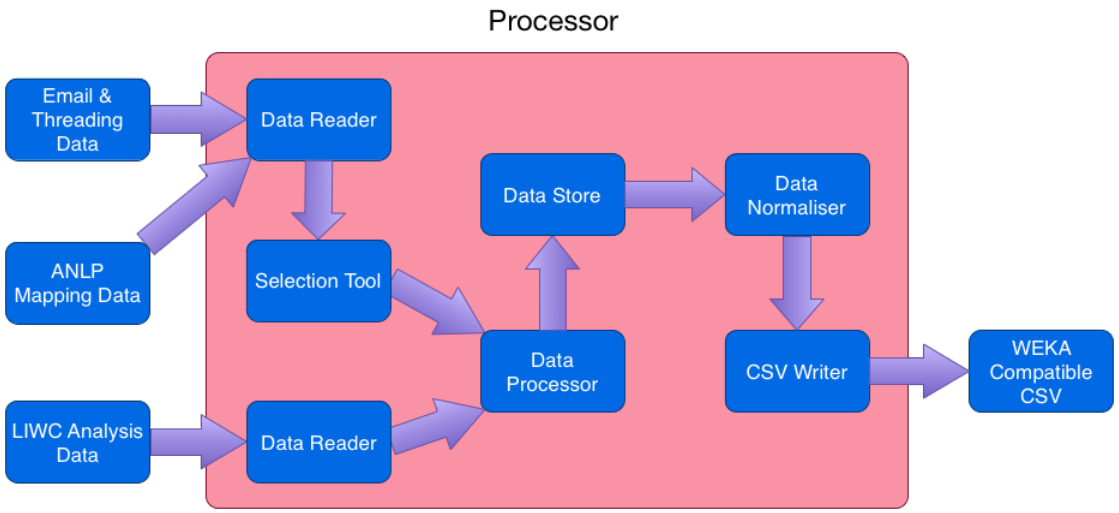

Figure 6.4: Processor Implementation

6.7 High Level Architecture

As a result of implementing these changes, the High Level Architecture of the system has changed from the original. The new High Level Architecture is visible in 6.5. While the implemented system shares all the same functionality as the original, the modular structure makes it far easier to implement changes and develop small parts of the system at a time, and also makes the system far quicker on anything except a straight run from dataset to classifier.

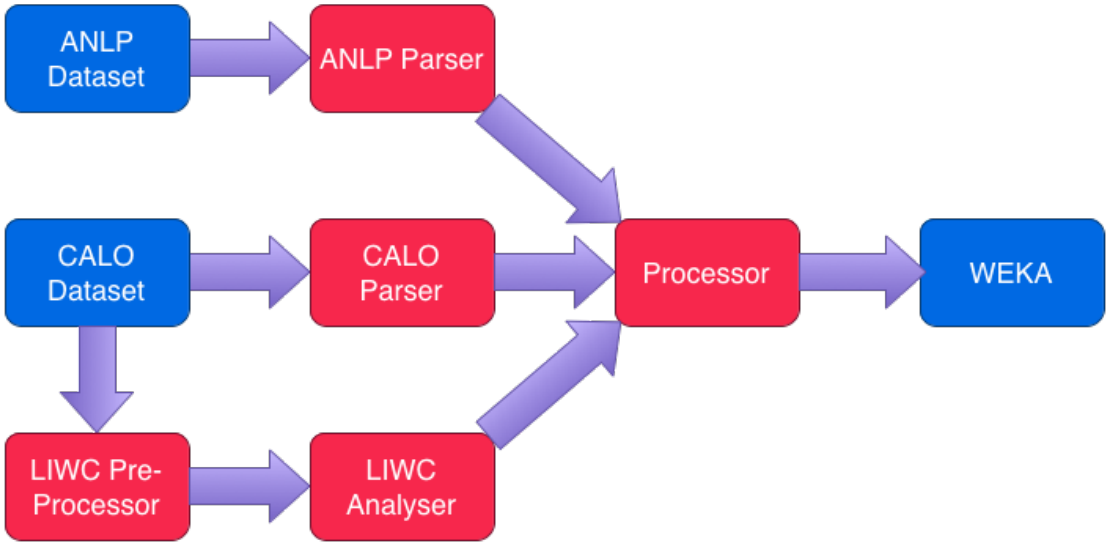

Figure 6.5: High Level Implementation

Chapter 7

Analysis

7.1 Analysis Methods

All my analyses follow the same structure using the tools identified below. Building the classifier has two stages in WEKA. The first is to identify the "best" features using an evaluation tool, search method and training set traversal method (the definition of best varies depending on the evaluation tool used).

The second stage is to reduce the initial set of features down to those defined as significant in stage 1. Then using a classification algorithm and another traversal method, the classifier is trained on the dataset. The resultant classifier will then output a statistical analysis on the data, including a confusion matrix and some other key figures. The methods used are discussed at each stage below.

7.1.1 Feature Selection

Attribute Evaluator & Search Method

InfoGain was selected at the Feature Selection tool, combined with Ranker as search method. This is based on previous success in specific email classification work by a number of authors using this pairing of (Janecek, Gansterer, Demel and Ecker, 2008; Gilad-Bachrach and Navot, 2006).

The InfoGain algorithm integrates with the Ranker algorithm very well, giving me a clear, ordered overview of the most influential features, allowing me to quickly prune features that do not contribute to the overall classification.

InfoGain itself measure of how much the information in a given attribute contributes to the classification as a whole, an attribute with an InfoGain score of 1 contributes entirely to the classification whereas a very low InfoGain score means that the attribute contributes little or nothing to the overall classification.

The Ranker algorithm simply ranks the features based on their merit, the output from the InfoGain algorithm. The Ranker's behaviour is more complex when the Traversal Method performs several passes over the data, like when performing a number of cross-validation passes. Here the Ranker algorithm will also perform some analysis of standard error, tracking fluctuation over the various data folds.

Traversal Method

Cross-validation has been identified as the Traversal Method of choice. This performs computation on 10 different folds of the data, in order to ensure my results are as smooth and consistent as possible. Rather than performing one pass of the data like when used with a full training set, cross validation with 10 folds will first split the dataset into 10 subsets. Then 10 passes over the data will be performed, each time a different subset will

be used as the test set, and the other 90% of the data will be used as the training set. The results from each of these folds will then be averaged to give the overall result.

7.1.2 Classification

Classification Method

NaiveBayes was used as the method of classification, on a 10 fold cross-validation test set. The 10 fold cross-validation is used for the same reasons as above.

Naive Bayes is a very popular classification method used throughout a large number of research fields. The method itself is an implementation of Bayes' theorem, operating under the assumption that all features are statistically independent (naive). There have been a large number of direct comparison works, and Naive Bayes will frequently prove itself the most flexible algorithm, achieving good results in almost all cases.

7.2 Predicting ANLP Importance

The first experiment is to build a classifier of importance based on the categories in the ANLP Dataset. The email categories have been grouped into 5 by the author, ranging from strong positive importance, to neural importance to strong negative importance on a 5 point scale. The categories are visible in Appendix A.1 and the mapping of categories to groups is visible in Appendix A.2.

This is certainly a subjective grouping, in fact it is doubly subjective. The author performed a subjective judgement on which importance group a category belongs to. However, the given email had already been subjectively judged by both the users who applied the initial categories to the dataset.

Given these judgements, this classifier is almost classifying which categories each given email falls into, which have then been assigned to an importance grouping. This is not entirely true due to the way that I have decided on the importance statement, and the one-to-many relationship it has, however it is close enough.

7.2.1 Predictor

The predictor for this classification is an email's perceived Importance Statement. This is generated using the importance groups discussed above. Each of these groups are assigned an importance rating, and the emails overall importance rating is the calculated mean of these values.

The individual importance ratings are between 1 and 0 with increments of 0.25, so strong positive importance has a rating of 1, and neutral importance has a rating of 0.5.

This measure of calculating the importance rating means that the classifier is not quite predicting categories, as there are many different ways of achieving a given importance rating (although finite, given the finite number of categories and maximum 2 people classifying each email).

For this experiment emails have been defined as important if its importance rating is greater than 0.75, meaning that regardless of the other categories assigned, at least one of them must be from the Strong Positive Importance group.

7.2.2 Data

The input data consists of the 1702 emails in the ANLP dataset, with 87 attributes each. Of these, 242 have been identified as important (as per the definition above), leaving 1460 emails as Non-important.

Given these raw values, we can see that a random classifier would have an expected success rate of 16.6%, so any improvement over this can be considered an improvement.

A feature of note for the ANLP dataset are the results from the threading analysis. Of the 1702, threading identified 1438 emails as being in a thread, and 264 not in a thread, so 81.6% of the dataset is in a thread, a figure corroborated by the overall dataset thread average of 83.6% (see section 7.3.1). This is significantly higher than the benchmark 60% of the emails in the overall Enron dataset that Klimt and Yang identified as being in threads, albeit using their self admittedly untested thread detection mechanism (Klimt and Yang, 2004).

This could be a side effect of the unsophisticated thread detection mechanism described above, or perhaps the "semi-motivated fashion" that the emails were selected in. For this, the creators focussed on "business-related emails and the California Energy Crises and on emails that occurred later in [that] collection, trying to avoid very personal messages, jokes, and so on".

This does also suggest a worrying inverse relationship between my definition of importance from the ANLP dataset and whether a message is replied to, however this could be a result of the email selection process described above.

7.2.3 Feature Selection

Prior to building the classifier and attempting the classifier in WEKA, Feature Selection was performed in order to identify the most influential features, cutting the initial set of 87 features down to 20.

By using InfoGain+Ranker the Feature Selection outputted a ranked list of results, allowing the easy identification of the most influential features. Of the 86 prediction features, 16 gave non-zero results. Figure 7.1 shows a graph of the distribution of the remaining results. Based on these findings the feature set was reduced down to the most significant 20 features.

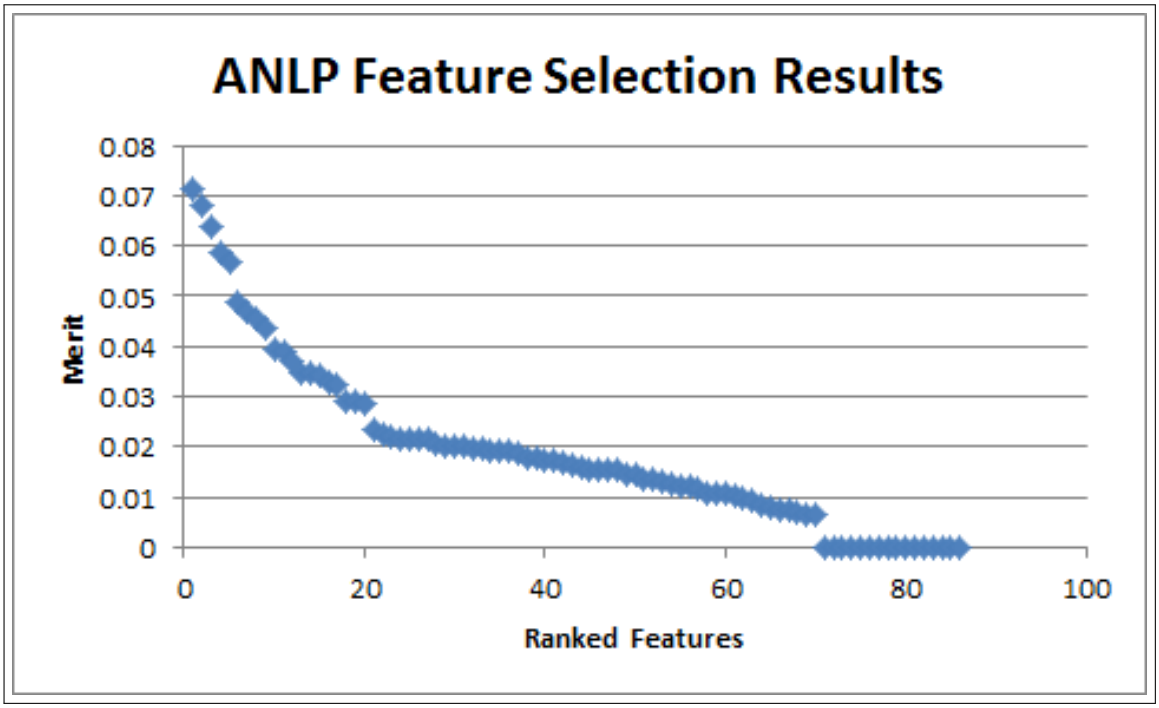

Figure 7.1: ANLP Initial Feature selection

7.2.4 Feature Discussion

This reduced set of features is visible in Table 7.1. Their datatype has been added, as this is interesting to review with previous findings by Klimt and Yang and Manco et al., discussed in section 2.6.1 of my Literature Review. These authors suggested that Numeric features were not at all useful (Klimt and Yang, 2004), and could be useful (Manco et al., 2002) respectively.

These results support Manco et al.'s view, a view also originally supported by the author. However, 4 of 20 features in one feature reduction having the Numeric type is hardly conclusive.

Other items of note here is the predictive power of punctuation. AllPct, Colon and OtherP all number in the top 5, and a full quarter of the top 20 are features associated with punctuation. While there is some overlap between the groups; AllPct subsumes Colon, OtherP and Dash, those elements are all clearly all strong enough predictors of their own accord.

This could be for a number of reasons, it could perhaps be linked with the Length of the Email, as both the Word Count and Length of email also possess a level of predictive power. Another category that features heavily are pronouns, which is certainly relevant. Heavy use of pronouns could suggest more questions or statements, both of which are likely to be more important.

Rank	Feature Name	LIWC Parent Category	Data Type
1	Number_of_Recipients	N/A	Categorical, Numeric
2	AllPct	Punctuation	Unstructured
3	Number_of_Tos	N/A	Categorical, Numeric
4	Colon	Punctuation	Unstructured
5	OtherP	Punctuation	Unstructured
6	article	Linguistic Processes	Unstructured
8	Dic	Linguistic Processes	Unstructured
7	funct	Linguistic Processes	Unstructured
9	ipron	Pronouns	Unstructured
10	Dash	Punctuation	Unstructured
11	ppron	Pronouns	Unstructured
12	cogmech	Psychological Processes	Unstructured
13	i	Pronouns	Unstructured
14	preps	Linguistic Processes	Unstructured
15	you	Pronouns	Unstructured
16	hear	Psychological Processes	Unstructured
17	past	Linguistic Processes	Unstructured
18	WC	Linguistic Processes	Numeric
19	money	Personal Concerns	Unstructured
20	Length_of_Email_Body	N/A	Numeric

Table 7.1: Ranked ANLP Features

7.2.5 Classification

Using the Naive Bayes classifier and the features described above, a 10 fold cross validation was performed, giving an overall correct classification percentage of 78.2%. The results are visible in Figure 7.2.

This is is a significant improvement over random classification, however the confusion matrix results are not entirely convincing. There are a large number of false negatives and a reasonable number of false positives here.

Considering this classifier as a solution to my original problem statement, we are more interested in what is considered Important rather than Non-Important. This is important as the user may lose trust in the classifier if it does not appear to be supplying a good selection of Important emails.

This measure of a classifier is known as the classifier's precision, that is how many relevant to irrelevant results are returned. In this respect, the precision of the Importance classification is especially bad, with a precision of 34.8%. That is to say, of all the emails that this classifier returns as Important, only 34.8% of them are true positives.

It is unfortunate, as the classifier performs relatively well in other metrics achieving 92.6% precision on Non-Importance.

```
Correctly Classified Instances        1331               78.2021 %
Incorrectly Classified Instances       371               21.7979 %
Kappa statistic                          0.3189
Mean absolute error                      0.2276
Root mean squared error                  0.4187
Relative absolute error                 93.1764 %
Root relative squared error            119.882  %
Coverage of cases (0.95 level)          91.3631 %
Mean rel. region size (0.95 level)      65.5699 %
Total Number of Instances             1702

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.607    0.189    0.348      0.607    0.442      0.338    0.772     0.389     Important
                 0.811    0.393    0.926      0.811    0.865      0.338    0.771     0.943     Non Important
Weighted Avg.    0.782    0.364    0.844      0.782    0.804      0.338    0.771     0.864

=== Confusion Matrix ===

    a    b   <-- classified as
  147   95 |    a = Important
  276 1184 |    b = Non Important
```

Figure 7.2: ANLP Initial Feature Selection Results

7.2.6 Discussion

While these results show the classifier as performing relatively well overall, the low precision for Important emails means that it is not effective in the roll originally envisaged. However, there are still some feature correlations here worth discussing.

Number of Recipients and Word Count (WC) as features correlated positively with Importance (Appendix B.1 B.2). Equally interesting is the inverse relationships between punctuation and Importance, that is emails with more punctuation appear to be less important. The results supporting this claim are visible in (Appendix B.3). The significance of these features are discussed further when comparing features in section **??**.

There are a number of potential changes that could potentially be made in the interests of improving this classifier. The importance groupings could be revisited, perhaps re-evaluating the groups or splitting up the groups further. The method of weighting could also be investigated, perhaps giving a category that both users apply to a category extra weighting.

Further processing on the dataset itself could also be performed, perhaps performing stemming and stop word removal, however this could have an effect on the existing punctuation features.

The simplistic threading algorithm could be revisited, and replace it with one matching unquoted to quoted text, following the work by Lewis and Knowles (Lewis and Knowles, 1997).

7.3 Predicting Message Threading

The second analysis performed is the prediction whether an email is in a thread or not, in effect classifying whether an email is replied to. As opposed to the ANLP classification method discussed above, the result is not subjective, although the test for thread membership can still be put into question.

7.3.1 Predictor

As stated above, the predictor for this classification is whether an email is in a thread or not. This feature was generated using a subject matching algorithm on the email subjects. Prior to performing this threading analysis, each subject was pre-processed, and "RE" and "FW" tags removed, in order to improve the efficiency of the matching algorithm.

The set of emails was sorted by date and for each given email the threading algorithm would check every email in a given time period for a matching subject. If a matching subject was found, then the email was considered to be in that thread. For this analysis a time period of one day was selected, mainly for performance reasons, as threading proved to be incredibly computationally taxing.

7.3.2 Data

Threading Analysis was performed on the whole CALO Dataset, consisting of 517424 emails. Of these, the Threading Analysis calculated that 432550 were in a Thread, and

84874 were not, that is 83.6% of emails are in a thread. This is a very high percentage of emails, far higher then the 60% that Klimt and Yang identified using their simple method of Threading Analysis (Klimt and Yang, 2004).

It is immediately apparent that achieving a classifier better than random selection with this data will be very difficult; a random classifier will guess correctly 83.6% of the time. This immediately brings both the quality of the dataset and the quality of the threading algorithm into question. Nevertheless the classification was performed.

7.3.3 Feature Selection & Discussion

As per experiment 1, the Feature Selection was performed using InfoGain+Ranker, which identified the 23 most influential and allowed the feature to be cut down from 87. The top 23 were chosen as there were 7 features with the same merit in the 16-23 range. Figure 7.3 shows the distribution of merit amongst features, and Table 7.2 shows the specific features.

There is some overlap between both sets of Features, that I discuss further in section **??**

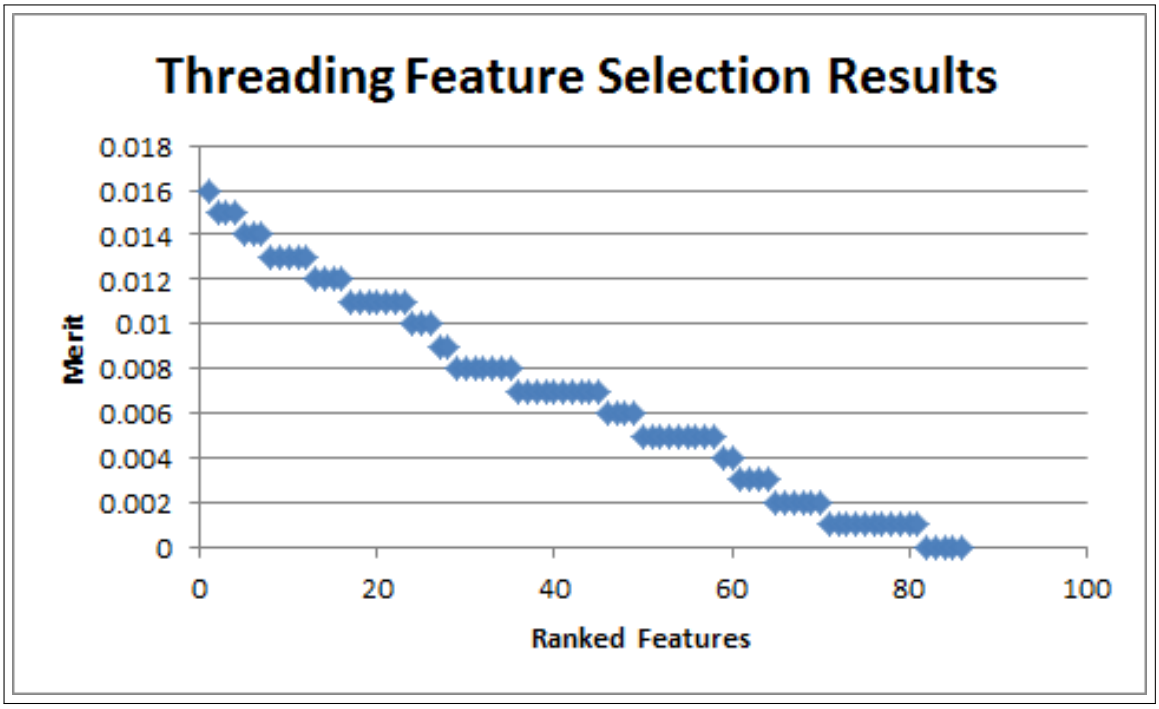

Figure 7.3: Threading Initial Feature Selection Results

7.3.4 Classification

A 10 fold classification on the dataset in the same manner as for experiment 1. As predicted, the classifier did not perform better than random classification, achieving 78% overall

Rank	Feature Name	LIWC Parent Category	Data Type
1	Length_of_Email_Body	N/A	Numeric
2	Period	Punctuation	Unstructured
3	WC	Linguistic Processes	Categorical, Numeric
4	WPS	Linguistic Processes	Categorical, Numeric
5	affect	Psychological Processes	Unstructured
6	Colon	Punctuation	Unstructured
7	Apostro	Punctuation	Unstructured
8	present	Linguistic Processes	Unstructured
9	time	Relativity	Unstructured
10	verb	Linguistic Processes	Unstructured
11	Dic	Linguistic Processes	Unstructured
12	posemo	Affective Processes	Unstructured
13	Sixltr	Linguistic Processes	Categorical, Numeric
14	funct	Linguistic Processes	Unstructured
15	OtherP	Punctuation	Unstructured
16	ppron	Pronouns	Unstructured
17	Number_of_Recipients	N/A	Categorical, Numeric
18	pronoun	Pronouns	Unstructured
19	motion	Relativity	Unstructured
20	space	Relativity	Unstructured
21	adverb	Linguistic Processes	Unstructured
22	auxverb	Linguistic Processes	Unstructured
23	relativ	Biological Processes	Unstructured

Table 7.2: Ranked Threading Features

accuracy. The results of classification are visible in Figure 7.4.

Given the Threading Analysis' poor performance it is clear that the method of Thread Analysis is to blame here, being the common denominator between this analysis and the ANLP dataset.

```
Correctly Classified Instances      403483               77.9792 %
Incorrectly Classified Instances    113941               22.0208 %
Kappa statistic                          0.1171
Mean absolute error                      0.2311
Root mean squared error                  0.4429
Relative absolute error                 84.2606 %
Root relative squared error            119.6051 %
Coverage of cases (0.95 level)          85.2952 %
Mean rel. region size (0.95 level)      59.7667 %
Total Number of Instances           517424

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.216    0.110    0.279      0.216    0.244      0.118    0.631     0.233     Not In Thread
                 0.890    0.784    0.853      0.890    0.871      0.118    0.631     0.892     In Thread
Weighted Avg.    0.780    0.673    0.759      0.780    0.768      0.118    0.631     0.784

=== Confusion Matrix ===

      a      b   <-- classified as
  18343  66531 |      a = Not In Thread
  47410 385140 |      b = In Thread
```

Figure 7.4: Thread Prediction Results

Chapter 8

Future Work

8.1 Future Work

Given that the project is an exploratory study, it is not surprising that the research touched on a great many topics spanning a wide breadth of fields. Many of these topics are large enough that they could easily be a project unto themselves. A number of these topics are explored below, identifying how they could be used to extend further extend the work.

8.2 Kilander's Studies

Despite their age and the topic they cover (Usenet News rather than email), there is a large amount of work to be performed investigating several of the Features that Kilander defines in his survey. Of the 13 Features he suggested (see Table 3.6), there has been work that mainly focusses on one or two of the features, yet very few that have attempted to combine several together in order to develop a classifier (Kilander, 1996).

Keywords and Commitment seem to hold the most promise as potential features, based on existing work. Work by Dabbish et al. and Carvalho and Cohen gave some very interesting results on speech acts. Speech acts concern predicting what action the email is asking of a user, very close to the concept of commitment - "What does this email expect of me?" (Dabbish et al., 2005; Carvalho and Cohen, 2005).

Keywords were initially dismissed when potential features were discussed, it being argued that Keyword analysis is not being relevant to email, given their obvious lack of keywords. However, with an analysis method like tf-idf (text frequency, inverse document frequency), it suddenly becomes possible to generate a set of features from an email that look suspiciously similar to Keywords. Perhaps performing a LIWC analysis on these keywords could further refine this feature.

8.3 ANLP Dataset

There may be far more predictive power in the ANLP dataset, however any true classification was hindered by the very subjective method with which the groups were selected. A more rigorous, statistical analysis of the dataset could allow a far more informed judgement when defining which class is assigned to each group. Performing a LIWC analysis of the emails in each category could also aid this.

However, it is entirely possible that either the categories are not correlated do not have a statistically significant correlation with importance, or perhaps the subjective categories were not applied in a rigorous enough manner in order for the labels to be a truly good method of classification.

Assuming that a good correlation could be found, an interesting experiment would be to use the ANLP dataset as a training set for the whole CALO Dataset.

The current dataset does not apply any special reasoning to the frequency with which a category is applied to an email. This could be explored further, as the same category applied independently by two users is certainly more reliable than the alternative, although it could be argued if it is more significant. Work could focus only on these independently agreed upon values (of which there might only be a few, given there are only 1700 emails in total), or could apply an increased weighting to these emails.

8.4 Advanced Thread Analysis

During research into Thread Analysis, a number of papers were encountered focusing entirely on Email Threading. These papers covered the subject at a far greater depth than that in the Technical Investigation (section 3.4). Given the suspected poor performance of the Threading Analysis method, implementing the Quoted/Unquoted text analysis described by Lewis and Knowles, and expanded up by Yeh and Harnly (see section 3.4) would likely improve reliability dramatically (Lewis and Knowles, 1997; Yeh and Harnly, 2006).

Yeh and Harnly also describes a method for recovering emails from quoted email messages, which could be worth investigating if the Enron Dataset does have as many as 60% of it's emails in threads (as stated by (Klimt and Yang, 2004)), there is a high likelihood of being able to recover a number of emails not otherwise in the dataset.

Using the threading data to manipulate the dataset is another potential use, organising a dataset by groups of emails. Labelling a small subset of emails in a given thread could allow machine learning techniques to apply labels to all emails in a thread, in a semi-structured manner. This labelled set of emails could be used as another subjective measure of importance, as a user labels threads as important/non-important based on the thread name and a few other key features.

8.5 Performance Increases

While attention was paid to performance throughout both the Design and Implementation sections (5 and 6 respectively), there are still further improvements that can be made, in particular to the methods for storing the files between modules. Different storage methods could be considered, including databases. An SQLite implementation was attempted but abandoned due to inflexibility, however this was attempted before the re-design documented in the Implementation, which may have helped.

Another potential route to improve Performance could be through parallelisation. The modular structure of the system inherently supports this, however the problem of data bottlenecks could potentially become even more prevalent as networks are introduced to the system.

8.6 Further Dataset Processing

Finally, improved dataset processing could improve the quality of many results. As already identified in 3.2 stop word removal, stemming and punctuation removal could all improve the results generated by the LIWC tool, (however punctuation has also proved itself to be a potentially useful feature).

The removal of duplicate emails could also have a significant positive impact on the project. This refers to the case where an email exists in several of a user's folders (having been copied there automatically by their email software), or the case that the sender and receiver are both in the CALO dataset, so existing in one users sent folder, and another users inbox.

While a large amount of folder analysis was performed as a part of my Technical Investigation into the CALO Dataset (see section 3.1.1), this data was not used when performing the Analysis. This could have been useful for partitioning the Dataset up and utilising some of the basic folder categorisation (even deleting emails is some form or categorisation).

The presence of attachments (and the filetype of the attachment is another of Kilander's features. However, the CALO dataset does not contain the attachment information. Investigation into the EDRM dataset (see section 3.1.6 revealed that the EDRM dataset does have these attachments. There had been some work to apply these attachments back across to the CALO dataset, that could be implemented to add another potentially valuable feature (Dredze, 2009).

One trend of emails in the dataset is the tendency towards extreme values. This became exceedingly obvious when visualising and applying filters to the data in the WEKA suite. It was often the case that data would be skewed by just 1 or 2 large values, especially the Numeric fields like email body length. Given the method of normalising the data, just one large value could significantly skew the data. As a result, filtering the data down to 2 or 3 standard deviations from the mean as per the 3 sigma rule, could have a positive impact on many of the features in the dataset.

However, calculating this data is still potentially complex, the data would have generated entirely, then the outliers identified and removed, then the remaining data re-normalised. This sort of action is not supported by the current system implementation, and would not be without computational complexity (calculating the mean and standard deviation on 87 features over 500,000 emails).

Chapter 9

Conclusion

9.1 Overall Summary

The overall aim of the project was to present an exploratory study into the field of email classification. This overarching goal was made up of a number of specific objectives:

The initial work towards this goal was the Literature Review (chapter 2), where a number of high level the overarching areas of research were identified, in addition to some more specific tools and methods of analysis. This also identified several areas that required further investigation.

Following the Literature Review, the Technical Investigation (chapter 3) investigated the tools, potential features and datasets identified in the Literature Review at much greater depth. The Technical Investigation concluded with a summary of all research to date, identifying the features and tools that would be used in the project going forward.

Based on this summary, the high level Requirements were formed (chapter 4), specifying the features and tools to be used, along with the technologies.

Using the Requirements, a high level system Design (chapter 5) and justification was performed, mapping the requirements to their specified design parts.

The Implementation chapter documented the actual Implementation (chapter 6) of the Design, documenting and justifying deviation from the design, and producing new documentation to reflect these changes.

The Analysis chapter (chapter 7) documents the initial analysis performed, discussing results correlation and implications.

The Future work chapter (chapter 8) begins reflecting on my work, identifying ways to improve my work, and other directions to take it in.

9.2 Objectives

The Introduction set out these objectives as the measure by which project success would be considered:

1. Carry out a thorough survey of existing work, specifically focussing on the various datasets, tools and methods used in other work.
2. Identify an appropriate email dataset.
3. Investigate a variety of potential definitions to inform my definition of email importance.
4. Identify a number of potential features based on these initial definitions.
5. Design and Implement a tool for parsing and processing the given dataset.

6. Produce a classifier based on the previously determined features to produce a statistical model of a given emails importance.

7. Host completed work online for use in future studies.

Chapters 2 and 3 performed a comprehensive survey of existing work, investigating several different Enron Datasets in close detail and identifying both the LIWC and WEKA tools, along with identifying many methods of email threading addressing objectives 1-4.

Chapters 4-6 addressed the specific Requirements specification, Design and Implementation of the suite of tools used to parse the dataset from its basic file structure, perform the feature analysis methods described in chapters 2 and 3, and output them to a format compatible with the WEKA tool.

Chapter 7 discussed the specific analyses performed and classifiers created, along with their success in addressing the original project goal.

The suite of tools developed by the project are accessible at `https://github.com/jamesdowdall/Enron-Suite`, meeting objective 7.

9.3 Reflection

While the Objectives are one way of judging project success, these are binary measures based on goals identified over 6 months prior to the project completion.

The overall goal of the project was as to perform an exploratory study, to establish a stepping stone into the field of email importance. In this aspect the project has been successful, a significant amount of time was put into the literature review and technical investigation, giving a novice user a comprehensive introduction into the field, the tools it uses and the datasets that are commonly worked on.

The publishing of source code online in an open source manner also significantly reduces the cost of entry into the field for any prospective researcher, contributing to the concept of this paper as a stepping stone for someone wanting to perform in this area.

The Design and Implementation sections have outlined some of the key practical problems with performing work in this area, namely the very real performance and storage issues that result from working with such large datasets.

The Future Work category also builds on the initial research by providing some very directions in which to build and improve on this initial work, referring back to the in-depth investigation performed in the Literature Review and Technical Investigation.

The Analysis itself has not been as positive as the work carried out in the rest of the project, while some initial classifications were produced, this work was inconclusive at best. While LIWC was identified as a good tool for generating features, and work on the ANLP dataset showed some initial promise as a dataset, the sophistication of the email

threading algorithm caused issues with the prediction. As the threading algorithm did not produce results in line with previous work, it was difficult to trust the prediction resulting in the inconclusive analysis that makes little progress towards achieving an overall classifier of email importance.

To conclude, the project presents a strong introduction into the field of email importance and provides an ideal starting point for someone looking to perform work in this area. While the project provides little in the way of contribution towards achieving an overall classifier of email importance, it provides contribution in the form of a suite of tools that make further work in the field easier to perform, removing a significant barrier to entry.

Bibliography

Ahmed, S. and Mithun, F. (2004), Word stemming to enhance spam filtering., *in* 'CEAS', Citeseer.

ANLP Project (2004), 'Anlp dataset', `http://bailando.sims.berkeley.edu/enron_email.html`. Accessed: 10-04-2014.

Balter, O. (1997), Strategies for organising email, *in* H. Thimbleby, B. OConaill and P. Thomas, eds, 'People and Computers XII', Springer London, pp. 21–38.

Bekkerman, R., McCallum, A. and Huang, G. (2005), 'Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora'.

Bird, S., Klein, E. and Loper, E. (2009), *Natural language processing with Python*, O'Reilly Media, Inc.

Blanzieri, E. and Bryl, A. (2008), 'A survey of learning-based techniques of email spam filtering', *Artificial Intelligence Review* **29**(1), 63–92.

Carvalho, V. R. and Cohen, W. W. (2005), On the collective classification of email "speech acts", *in* 'Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '05, ACM, New York, NY, USA, pp. 345–352.
URL: *http://doi.acm.org/10.1145/1076034.1076094*

Cassidy, A. and Westwood-Hill, M. (2010), 'Removing pii from the edrm enron data set'.

Chan, J., Koprinska, I. and Poon, J. (2004), Co-training with a single natural feature set applied to email classification, *in* 'Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence', WI '04, IEEE Computer Society, Washington, DC, USA, pp. 586–589.
URL: *http://dx.doi.org/10.1109/WI.2004.43*

Dabbish, L. A. and Kraut, R. E. (2006), Email overload at work: An analysis of factors associated with email strain, *in* 'Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work', CSCW '06, ACM, New York, NY, USA, pp. 431–440.

Dabbish, L. A., Kraut, R. E., Fussell, S. and Kiesler, S. (2005), Understanding email use: Predicting action on a message, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '05, ACM, New York, NY, USA, pp. 691–700.
URL: *http://doi.acm.org/10.1145/1054972.1055068*

Dabbish, L., Kraut, R., Fussell, S. and Kiesler, S. (2004), To reply or not to reply: Predicting action on an email message, *in* 'ACM 2004 Conference', Citeseer.

Dredze, M. (2009), 'Calo emails with edrm attachments', `http://www.cs.jhu.edu/~mdredze/code.php`. Accessed: 10-04-2014.

Dumais, S., Platt, J., Heckerman, D. and Sahami, M. (1998), Inductive learning algorithms and representations for text categorization, *in* 'Proceedings of the Seventh International Conference on Information and Knowledge Management', CIKM '98, ACM, New York, NY, USA, pp. 148–155.
URL: *http://doi.acm.org/10.1145/288627.288651*

EDRM Project (2010), 'Edrm dataset'.
URL: *http://www.edrm.net/resources/data-sets/edrm-enron-email-data-set*

Enron Data Reconstruction Project (2009), 'Anlp to calo email mappings', `http://enrondata.org/assets/edrp_calo-ucb-anlp-map.txt`. Accessed: 10-04-2014.

Fisher, D., Brush, A. J., Gleave, E. and Smith, M. A. (2006), Revisiting whittaker & sidner's "email overload" ten years later, *in* 'Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work', CSCW '06, ACM, New York, NY, USA, pp. 309–312.
URL: *http://doi.acm.org/10.1145/1180875.1180922*

Gervasio, M. and Kaelbling, L. (2009), 'The enron dataset (calo)'.
URL: *http://www.cs.cmu.edu/ enron/*

Gilad-Bachrach, R. and Navot, A. (2006), Margin based feature selection and infogain with standard classifiers, *in* 'Feature Extraction', Springer, pp. 395–401.

Graham, P. (2002), 'A plan for spam'.

Guzella, T. S. and Caminhas, W. M. (2009), 'A review of machine learning approaches to spam filtering', *Expert Systems with Applications* **36**(7), 10206–10222.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I. H. (2009), 'The weka data mining software: an update', *ACM SIGKDD explorations newsletter* **11**(1), 10–18.

Halliday, M. and Hasan, R. (1976), *Cohesion in English*, Vol. 9.

Hedley, S. (2006), 'A brief history of spam', *Information and Communications Technology Law* **15**(3), 223–238.
URL: *http://www.tandfonline.com/doi/abs/10.1080/13600830600960758*

Introduction to Outlook Data Files (2011), `http://office.microsoft.com/en-gb/outlook-help/introduction-to-outlook-data-files-pst-and-ost-HA102749465.aspx`. Accessed: 10-04-2014.

Janecek, A., Gansterer, W. N., Demel, M. and Ecker, G. (2008), 'On the relationship between feature selection and classification accuracy.', *Journal of Machine Learning Research-Proceedings Track* **4**, 90–105.

Kilander, F. (1996), Properties of electronic texts for classification purposes as suggested by users, Technical report, Dept. of Computer and Systems Science, Stockholm University.

Kilander, F., Fåhræus, E. and Palme, J. (1997), 'Intelligent information filtering', *The IntFilter Project, Dept. of Computer and Systems Science, Stockholm University* .

Klimt, B. and Yang, Y. (2004), The enron corpus: A new dataset for email classification research, *in* J.-F. Boulicaut, F. Esposito, F. Giannotti and D. Pedreschi, eds, 'Machine Learning: ECML 2004', Vol. 3201 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 217–226.

Lantz, A. (1995), Useful criteria for intelligent filtering?, Technical Report 95-042, Dept. of Computer and Systems Science, Stockholm University.

Lewis, D. D. and Knowles, K. A. (1997), 'Threading electronic mail: A preliminary study', *Information processing & management* **33**(2), 209–217.

LIWC Project (2007), 'Liwc categories', `http://www.liwc.net/descriptiontable1.php`. Accessed: 10-04-2014.

Lovins, J. B. (1968), *Development of a stemming algorithm*, MIT Information Processing Group, Electronic Systems Laboratory.

Manco, G., Masciari, E., Ruffolo, M. and Tagarelli, A. (2002), Towards an adaptive mail classifier, *in* 'Proc. of Italian Association for Artificial Intelligence Workshop', Citeseer.

Mintzberg, H. (1973), *Nature of managerial work*, Harper and Row.

Mock, K. (1999), Dynamic email organization via relevance categories, *in* 'Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on', pp. 399–405.

Mock, K. (2001), An experimental framework for email categorization and management, *in* 'Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval', SIGIR '01, ACM, New York, NY, USA, pp. 392–393.
URL: *http://doi.acm.org/10.1145/383952.384033*

Pantel, P. and Lin, D. (1998), Spamcop: A spam classification & organization program, *in* 'Proceedings of AAAI-98 Workshop on Learning for Text Categorization', pp. 95–98.

Pennebaker, J. W., Chung, C. K., Ireland, M., Gonzales, A. and Booth, R. J. (2007), 'The development and psychometric properties of liwc2007', *Austin, TX, LIWC. Net* .

Porter, M. F. (1980), 'An algorithm for suffix stripping', *Program: electronic library and information systems* **14**(3), 130–137.

Postel, J. (1975), On the junk mail problem, Technical report, Network Working Group Request for Comments: 706.

Spamhaus (1998), 'The spamhaus project - the definition of spam'.
URL: *www.spamhaus.org/consumer/definition/*

Whittaker, S. and Sidner, C. (1996), Email overload: Exploring personal information management of email, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', CHI '96, ACM, New York, NY, USA, pp. 276–283.
URL: *http://doi.acm.org/10.1145/238386.238530*

Williams, R. V. (2010), 'Hans peter luhn and herbert m. ohlman: Their roles in the origins of keyword-in-context-permutation automatic indexing', *J. Am. Soc. Inf. Sci. Technol.* **61**(4), 835–849.
URL: *http://dx.doi.org/10.1002/asi.v61:4*

Yeh, J.-Y. and Harnly, A. (2006), Email thread reassembly using similarity matching, Conference on Email and Anti-Spam.

Appendix A

ANLP Category Data

A.1 ANLP Categories

Format of each line in .cats file: n1,n2,n3

n1 Top-level category
n2 Second-level category
n3 Frequency with which this category was assigned to this message

A.1.1 Categories

1 Coarse genre

1.1 Company Business, Strategy, etc. (elaborate in Section 3 [Topics])
1.2 Purely Personal
1.3 Personal but in professional context (e.g., it was good working with you)
1.4 Logistic Arrangements (meeting scheduling, technical support, etc)
1.5 Employment arrangements (job seeking, hiring, recommendations, etc)
1.6 Document editing/checking (collaboration)
1.7 Empty message (due to missing attachment)
1.8 Empty message

2 Included/forwarded information

2.1 Includes new text in addition to forwarded material
2.3 Forwarded email(s) including replies
2.2 Business letter(s) / document(s)
2.4 News article(s)
2.5 Government / academic report(s)
2.6 Government action(s) (such as results of a hearing, etc)
2.7 Press release(s)
2.8 Legal documents (complaints, lawsuits, advice)
2.9 Pointers to url(s)
2.11 Newsletters
2.12 Jokes, humor (related to business)
2.10 Jokes, humor (unrelated to business)
2.13 Attachment(s) (assumed missing)

3 Primary topics (if coarse genre 1.1 is selected)

3.1 Regulations and regulators (includes price caps)
3.2 Internal projects – progress and strategy
3.3 Company image – current
3.4 Company image – changing / influencing
3.5 Political influence / contributions / contacts
3.6 California energy crisis / california politics
3.7 Internal company policy
3.8 Internal company operations
3.9 Alliances / partnerships
3.10 Legal advice
3.11 Talking points
3.12 Meeting minutes
3.13 Trip reports

4 Emotional tone (if not neutral)

4.1 Jubilation
4.2 Hope / anticipation
4.3 Humor
4.4 Camaraderie
4.5 Admiration
4.6 Gratitude
4.7 Friendship / affection
4.9 Sympathy / support
4.8 Sarcasm
4.10 Secrecy / confidentiality
4.11 Worry / anxiety
4.12 Concern
4.13 Competitiveness / aggressiveness
4.14 Triumph / gloating
4.15 Pride
4.16 Anger / agitation
4.17 Sadness / despair
4.18 Shame
4.19 Dislike/ scorn

A.2 Subjective Categories

Strong Positive Importance

1.5 Employment arrangements (job seeking, hiring, recommendations, etc)
1.6 Document editing/checking (collaboration)
2.6 Government action(s) (such as results of a hearing, etc)
2.8 Legal documents (complaints, lawsuits, advice)
3.1 Regulations and regulators (includes price caps)
3.2 Internal projects – progress and strategy
3.5 Political influence / contributions / contacts
3.6 California energy crisis / california politics
3.10 Legal advice

Positive Importance

1.1 Company Business, Strategy, etc. (elaborate in Section 3 [Topics])
1.4 Logistic Arrangements (meeting scheduling, technical support, etc)
2.3 Forwarded email(s) including replies
2.4 News article(s)
2.5 Government / academic report(s)
3.4 Company image – changing / influencing
3.7 Internal company policy
3.8 Internal company operations
3.9 Alliances / partnerships
3.11 Talking points
3.12 Meeting minutes
3.13 Trip reports

Neutral Importance

1.2 Purely Personal
1.3 Personal but in professional context (e.g., it was good working with you)
2.1 Includes new text in addition to forwarded material
2.2 Business letter(s) / document(s)
2.9 Pointers to url(s)
2.13 Attachment(s) (assumed missing)
3.3 Company image – current

Negative Importance

2.7 Press release(s)
2.10 Jokes, humor (unrelated to business)
2.11 Newsletters
2.12 Jokes, humor (related to business)

Strong Negative Importance

1.7 Empty message (due to missing attachment)
1.8 Empty message

Appendix B

Results Visualisation

B.1 ANLP Classification Results

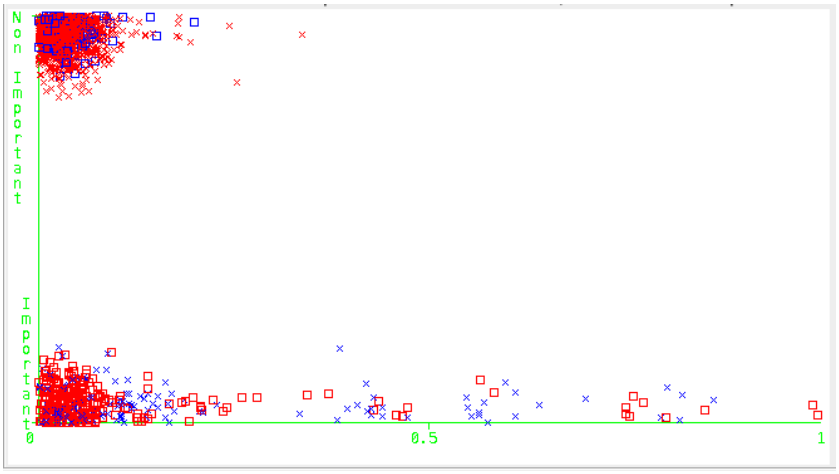

Figure B.1: Number of To's Classification compared with Importance

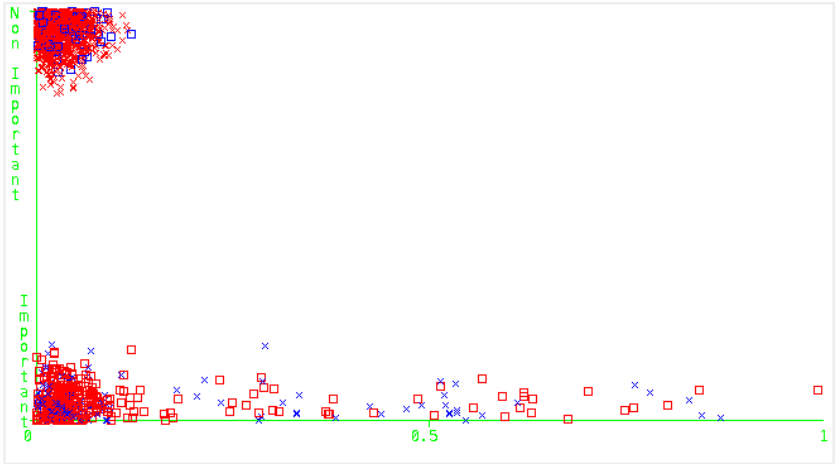

Figure B.2: Word Count Classification compared with Importance

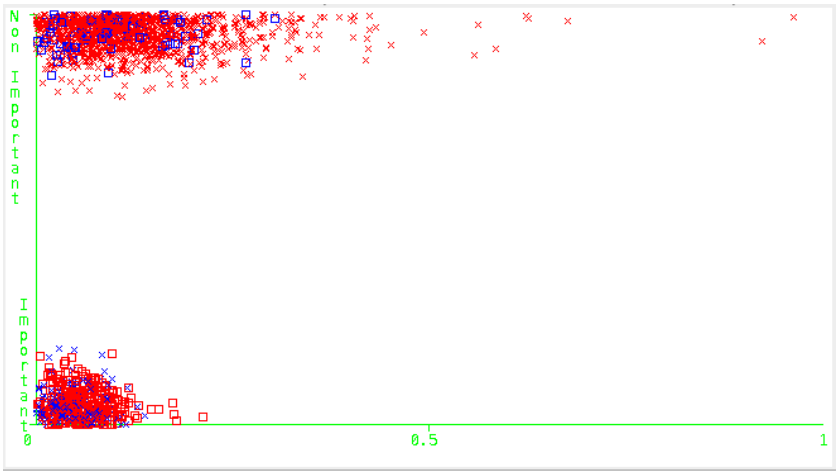

Figure B.3: All Punctuation Classification compared with Importance