An automatic email mining approach using semantic non-parametric K-Means++ clustering

Gunjan Soni
University of Windsor

AN AUTOMATIC EMAIL MINING APPROACH USING SEMANTIC NON-PARAMETRIC K-MEANS++ CLUSTERING

By

Gunjan Soni

A Thesis

Submitted to the Faculty of Graduate Studies through the School of Computer Science in

Partial Fulfillment of the Requirements for the Degree of Master of Science at the

University of Windsor

Windsor, Ontario, Canada

2013

AN AUTOMATIC EMAIL MINING APPROACH USING SEMANTIC NON-PARAMETRIC K-MEANS++ CLUSTERING

By

Gunjan Soni

APPROVED BY:

Dr. Sévérien Nkurunziza
Department of Mathematics and Statistics

Dr. Jianguo Lu
School of Computer Science

Dr. Christie I. Ezeife, Advisor
School of Computer Science

Dr. Dan Wu, Chair of Defense
School of Computer Science

May 3, 2013

AUTHOR’S DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone’s copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Email inboxes are now filled with huge varieties of voluminous messages and thus increasing the problem of "email overload" which places financial burden on companies and individuals. Email mining provides solution to email overload problem by automatically grouping emails into meaningful and similar groups based on email subjects and contents. Existing email mining systems such as Kernel-Selected clustering and BuzzTrack, do not consider the semantic similarity between email contents, also when large number of email messages are clustered to a single folder they retain the problem of email overload.

This thesis proposes a system named AEMS for automatic folder and sub-folder creation, indexing of the created folders with link to each folder and sub-folder, also an Apriori-based folder summarization containing important keywords from the folder. Thesis aims at solving email overload problem through semantic re-structuring of emails. In AEMS model, a novel approach named Semantic Non-parametric K-Means++ clustering is proposed for folder creation, which avoids, (1) random seed selection by selecting the seed according to email weights, and (2) pre-defined number of clusters using the similarity between the email contents. Experiments show the effectiveness and efficiency of the proposed techniques using large volumes of email datasets.

Keywords: Email Mining, Email Overload, Email Management, Data Mining, Clustering, Feature Selection, Folder Summarization.

DEDICATION

To my parents, my mother Mrs. Pushplata Soni and my father Dr. Nawal Kishore Soni. Also, to my sister Dr. Vandana Soni, my bother-in-law Mr. Ajay Soni and my nephew Utkarsh Soni.

ACKNOWLEDGEMENT

My sincere appreciation goes to my parents, siblings, brother-in-law and nephew. Your perseverance and words of encouragement gave me the extra energy to see this work through.

I will be an ingrate without recognising the invaluable tutoring and supervision from Dr. Christie Ezeife. Your constructive criticism and advice at all times gave me the needed drive to successfully complete this work. I like to thank Dr. Ezeife for continuous research assistantship positions from NSERC and FedDev/OBI grants.

Special thanks go to my external reader, Dr. Sévérien Nkurunziza, my internal reader, Dr. Jianguo Lu and my thesis committee chair, Dr. Dan Wu for accepting to be in my thesis committee. Your decision, despite your tight schedules, to help in reading the thesis and providing valuable input is highly appreciated.

And lastly to friends and colleagues at University of Windsor especially, Tamanna Mumu, Sajib Sinha, Yanal Ahmed, Sabbir Ahmed and Mohammad Harun-Or-rashid for their help and support, I say a very big thank you for your advice and support throughout the duration of this work.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Electronic mail, also called as e-mail or email, is an electronic communication system in which messages are sent from an author to one or more recipients electronically (Bogawar and Bhoyar, 2012). Email has become a dominant means of exchanging information across the world for the purposes of business, storing information or for personal matters. The growth of internet has impacted a lot on the growth of email, turning it to an important tool for work. The reason for the email popularity is the speediness of communication, the minimum cost and easiness while using email and most importantly it is asynchronous communication (Asynchronous communications, is in which data can be transferred intermittently rather than in a fixed stream. For example, a telephone discussion is asynchronous because both parties can talk when they like. If the communication were synchronous, each party has to wait a definite interval before speaking).

Email began as a simple and informal communication between colleagues and led to the gateway to the paperless office. People are sending and receiving hundreds of messages daily, communicating with partners and friends, or exchanging files and information around the globe. In recent years, it has contributed to the development of organizations that are distributed world-wide, allowing people to communicate around the globe at no incremental cost, thus, increasing the email overload problem.

1.1 Email Overload

Email overload (Xiang et al., 2009) is the state of completely overwhelming email inboxes by the amount of email one has received. According to an estimate given by (Radicati and Hoang, 2011), the number of email messages sent daily has reached around *3.1 billion* in 2011, resulting in the evolution of the problem of email overload. The problem that email inboxes are overloaded with many more tasks like task management and personal archiving instead of only communication, the study made by (Whittaker and Sidner, 1996) and (Fisher et al., 2006) showed that, dramatic changes have occurred in the way emails were handled when "email overload" term was coined and now in this present century, however, the difference can be seen in terms of archive size which is increased by remarkable size, whereas the message flow and inbox size remained almost unchanged.

In the solution to email overload, managing email messages by summarization and automatically categorizing emails into folders is a challenging application for data mining techniques. Automatic folder creation is, given a set of email messages and we need to semantically assign each message to a suitable cluster according to the email content and similarity between the email messages. Some automatic email folder creating techniques have been proposed in the related work are: Automatic Clustering Email Management System (ACEMS) (Schuff et al., 2006), BuzzTrack (Cselle et al., 2007), Automatic Nonparametric Text Clustering Algorithm (Xiang et al., 2009) and Kernel-selected email clustering (Yang et al., 2010). Another solution to email overload is given by the email summarization. The goal of email summarization is to provide concise, informative summary of email which in turn is helpful if user can read only the most important parts of message and then decide if the message demands immediate attention. Some email summarizations techniques have been proposed in the previous studies are: Task-focused summarization (Oliver et al., 2004), CWS (Clue Word Summarizer) (Carenini et al., 2007) and Regression based email summarization (Ulrich et al., 2009). Email categorization into folders and email summarization both fail to find specific email when there is a vague idea of email sender and vague idea of topic.

1.2 Email Management – Definition and Applications

Email Management is a set of programs that automatically handles email messages and attachments according to user-defined rules. It can involve spam filtering - finding whether the received email is spam or not, contact management - management of names and information of people associated with the names, and email overload which includes task management, email summarization, personal archiving, and categorization of email messages in different folders. The goal of any email management system is that it should be (Entlich, 2006):

- Complete: E-mail records should completely keep the transaction including the recipient(s), sender, time sent, message body text, date sent, subject lines, attachments should be included in full not just the file name, and recipient full email ID and name if the email is being sent to the group.
- Accurate: The transaction should be reflected by the content of the email records.
- Manageable: As part of the daily workflow and records management practices, the email records should be easy to manage by user.
- Secure: The e-mail record should be located in a secure system that controls access, storage, retrieval, modification, and deletion.

Below Figure 1.1 shows the categories of email management tasks.

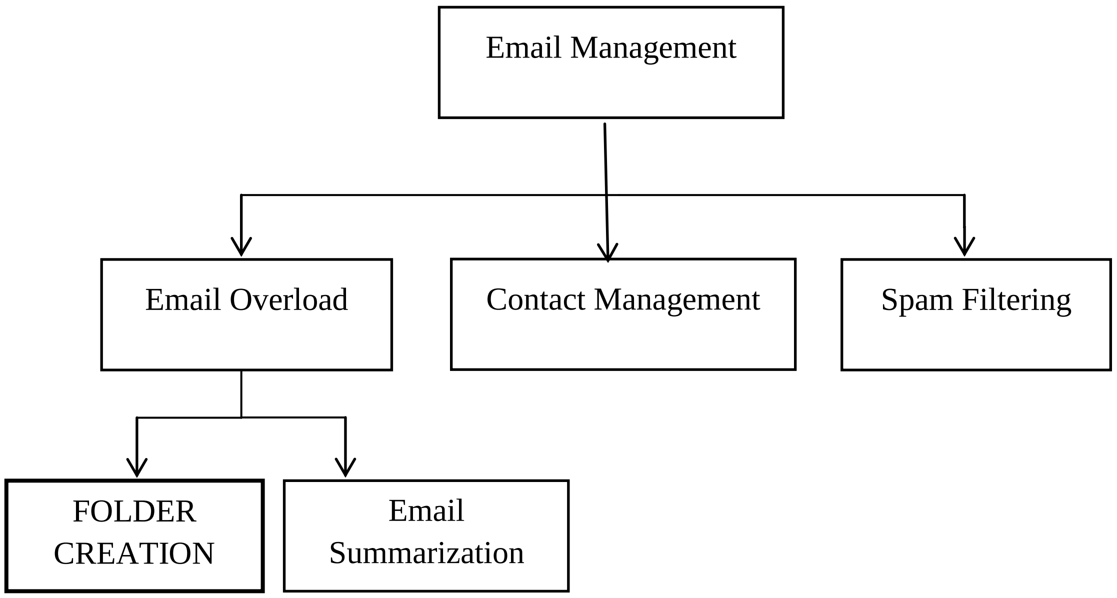


Figure 1.1 Categories of Email Management Tasks

1.2.1 Reducing Email Overload by Automatic Folder Creation

Automatic email organization into folders (Xiang et al., 2009) is a way to manage email messages and to reduce email overload. Many users organize incoming emails in different folders; it can be topic- oriented like “appointments”, “personal” and “entertainment” or group-oriented like “courses” and “project” or people- specific like “John” and “Mary”. To achieve this, many users create some rules to classify their mail by applying data mining methods such as classification (It is a process of classifying data into predefined groups) or clustering methods (clustering is used to place email messages into related groups without prior knowledge of the groups’ definitions). This can be automatic or semi-automatic i.e. involving user involvement. First, subject and body of each email are extracted from the input, which is a set of emails messages, i.e., email dataset and pre-processing is done on the extracted data, i.e., removal of stop words and applying stemming (Porter, 1997), if needed. Next, email is then represented in some format. Then, email grouping is done based on the data mining approached such as classification or clustering methods. Lastly, folders are created for each cluster and topics are detected for each folder. Some automatic folder creating techniques have been proposed in the previous studies including (Semantic email clustering (Li et al., 2006a), BuzzTrack (Cselle et al., 2007), EGM (Ayodele et al., 2010) and (Yang et al., 2010)). However, these techniques are limited because for instance, if a folder is created by the previously proposed techniques and in a single folder if there are 2000 email messages on the same topic then again it is a tedious job to find the desired email. For example, if a folder is created named “Assignment” and it contains 1500 emails all with assignment information, it is hard to find an email of a specific individual whose name is not specifically known by the user to search and it is also time consuming. Specially, these systems fail when user needs to find an email where user does not know or remember the specific topic to search with and the sender of email.

1.2.2 Reducing Email Overload by Email Summarization

Email summarization is a technique developed in order to automatically extract the summary of documents (Ulrich et al., 2009). It is important when people receive hundreds of messages daily of which some are newsletters, business decision-making messages, appointment arrangements etc. So, in this case, it would be very useful if reading of those entire messages can be avoided and instead only the important parts (or summary) are read to decide if the messages demand immediate attention. The building of such a tool includes natural language processing such as tokenizer, POS tagging and data mining techniques such as classification. Some automatic email summarization techniques have been proposed and include: Task-focused summarization (Oliver et al., 2004), CWS (Clue Word Summarizer) (Carenini et al., 2007) and Regression based email summarization (Ulrich et al., 2009). These techniques automatically extract the summary of documents extract the important candidate noun phrases (NPs) using POS tagger tools from the email message and manually classify the selected NPs into those that should be included in the summary and those are not. These NPs are used to build training set which is then used to summarize incoming messages. However, these techniques are limited because it is hard to find an email of a specific individual whose name is not specifically known by the user to search.

1.2.3 Contact Management

Contact Management is another email information management issue which manages names and addresses associated with key contacts (Whittaker et al., 2004). While most email systems can be customized to automatically extract email addresses into the address book, the other information (such as phone numbers) must be extracted manually from messages, which is tedious and error-prone process. But lots of information can be automatically extracted from email; for example, important contacts can be identified automatically through message-header information. Having identified contacts, it should also be possible for data mining techniques such as classification to automatically extract additional information from, such places as, signature files and web pages that could then be used to populate contact address fields. One of the contact management systems is ContactMap (Whittaker et al., 2004).

1.2.4 Spam Filtering

Spam filtering is a process of sorting out by identifying the unsolicited commercial mails (spam/junk/bulk) from a user's mail stream. For example, separating spam email "ONE-POUND-A-DAY DIET" from the email by a friend for technical help. When internet started for common public in mid-1990 the spam in email started to become a problem and is growing exponentially. The large amount of spam not only causes bandwidth problems, but also takes up valuable time from email users who try to separate and delete many unsolicited messages every day. Hence, spam filters are used to automatically handle spam in email. According to (Katakis et al., 2006), the method of building a spam filter can be categorized in two ways: Technical or Non-Statistical Approaches - which include white and black lists, digital signatures and handcrafted rules, and Machine Learning or Statistical Approaches like Naïve Bayes, Support Vector Machines, Latent Semantic Indexing, Case-Based Reasoning and Stacking Classifiers - which includes statistical linguistic analysis and are machine learning algorithms. Some examples of spam filtering systems are SpamBayes (Meyer and Whateley, 2004), Online supervised spam filtering (Cormack et al., 2007) and Semi-supervised spam filtering (Whissell and Clarke, 2011).

1.3 Data Mining Approaches for Email Management

Data mining is the analysis step of knowledge discovery from different views and situations and transforming it into the human useful information that can further be used for many purposes like decision making for day by day running of business. Data mining or knowledge mining (Han and Kamber, 2000) is defined as a nontrivial process of identifying valid, novel, potentially useful and understandable patterns in data. There are different techniques and approaches involved in discovering interesting information from huge set of data. Some of the approaches are: Classification, Clustering, Regression, Association rule mining.

Classification is a process of classifying data into predefined groups. This can involve assigning a given email into "spam" or "non-spam" classes. The algorithm that implements classification is known as a classifier. These classifiers can be built by various data mining techniques such as Naïve Bayes (Sahami et al., 1998), Support Vector Machines (Klimt and Yang, 2004), Rule Learning (Pazzani, 2002), Decision Trees based algorithms (Quinlan, 1986), Neural Networks (Diao et al., 2000) and Inductive Logic Programming (Crawford et al., 2002).

Clustering is used to place email messages into related groups without prior knowledge of the groups' definitions. This can involve automatic folder creation from a set of incoming messages of an email. K-means clustering algorithm (Li et al., 2006a) is one of the popular techniques for clustering.

Regression in data mining is the function that models the data, which can further be used to foresee future behavior of new data. This technique only works well with continuous quantitative data such as weight, speed or age. Genetic programming is one of such techniques used for this purpose.

Association rule mining (Han and Kamber, 2000) is a method for discovering interesting relations between attributes in large databases. For example, the rule {milk, bread} => {tea} found in the sales data of a supermarket would indicate that if a customer buys milk and bread together, customer is likely to also buy tea. Such information can be used as the basis for decisions about marketing activities such as promotional pricing or product placements.

1.4 Automatic Folder Creation by Data Mining Techniques

Existing approaches for automatic folder creation are based on two data mining techniques: Email Classification (SpamBayes (Meyer and Whateley, 2004), Online supervised spam filtering (Cormack et al., 2007) and Semi-supervised spam filtering (Whissell and Clarke, 2011)), and Email Clustering (Semantic email clustering (Li et al., 2006a), BuzzTrack (Cselle et al., 2007), EGM (Ayodele et al., 2010) and Kernel-selected clustering (Yang et al., 2010)).

1.4.1Automatic Folder Creation by Classification

In general, email classification takes into account the assignment of an email message to one of a pre-defined set of categories. Automatic email classification aims at building a model which will undertake this task on behalf of the user, by using data mining technique of classification. Existing email classification systems for folder creation are given by (Online supervised spam filtering (Cormack et al., 2007) and Semi-supervised spam filtering (Whissell and Clarke, 2011)). In classification there are two stages: training stage and classification stage. During training stage, after preprocessing emails, the email messages are grouped into classes i.e. class labels are given to email groups. In classification stage, the rest of the emails are taken as test data and classified using same classification algorithm , using the classifier such as Neural Networks (Diao et al., 2000), Rule Learning (Pazzani, 2002), and Support Vector Machines (Klimt and Yang, 2004) used during the training set, for assigning each email a class. Though classification can provide a better way to arrange email in folders but it is a semi-supervised approach.

1.4.2 Automatic Folder Creation by Clustering

Email Clustering is one of folder creating methods which are used for email mining. Clustering is used for automatic folder creation to reduce email overload. Existing email clustering systems for folder creation are given by (ACEMS (Schuff et al., 2006), BuzzTrack (Cselle et al., 2007), Automatic Non-parametric clustering (Xiang et al., 2009), EGM (Ayodele et al., 2010) and Kernel-selected clustering (Yang et al., 2010)). General steps for folder creation by clustering are shown in Figure 1.2. First, subject and body of each email are extracted from the input, which is a set of email messages, i.e., email dataset and pre-processing is done on the extracted data, i.e., removal of stop words and applying stemming (Porter, 1997), if needed. Next, email is then represented in some format. For example, ((Cselle et al., 2007), (Zeng et al., 2008) and (Yang et al., 2010)) represented email data in vector space model (VSM) (Salton et al., 1975) (this model is discussed later in section 2.2). In VSM the terms of subject and body with their respective weights (can be calculated using term frequency (TF)) are combined to form an email vector such as email E1 can be represented as email vector:

E1 = (copy: 3, apa: 2, citation: 2)

Where, “copy” is the term with 3 as weight, which is the number of times terms appeared in the emails. Then, email grouping or clustering is done ((Cselle et al., 2007) based on the data mining clustering approached such as K-Means clustering method (Li et al., 2006a). Lastly, folders are created for each cluster and topics are detected for each folder.

Email clustering is a better way to provide automatic folder creation for the purpose of reducing email overload because it is unsupervised and is better when one is using unlabeled data.

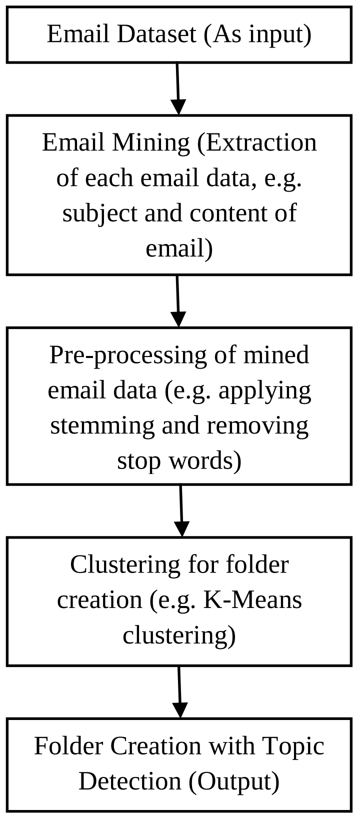

Email Dataset (As input)
Email Mining (Extraction of each email data, e.g. subject and content of email)
Pre-processing of mined email data (e.g. applying stemming and removing stop words)
Clustering for folder creation (e.g. K-Means clustering)
Folder Creation with Topic Detection (Output)

Figure 1.2 Automatic Folder Creation by Email Clustering

1.5 Problem with Current Email Management Systems

BuzzTrack proposed by (Cselle et al., 2007) is the popular tool for folder creation. BuzzTrack's functionality uses clustering as the techniques for automatic folder creation of email messages. Given a set of email messages as {m1, m2, m3 …. mN} and these emails are grouped into clusters {c1, c2 …. cM}. Each email contains Receipts ID (To), Sender ID (From), Message ID, date and time on which message received (date), subject of email (subject) and content of email. After tokenizing, the email body and subject into term of one word each, pre-processing is done by converting foreign character into canonical form, removing words with special character, identifying parts-of-speech for topic labeling and lastly applying stemming (Porter, 1997). After pre-processing, there exists a set of all n stemmed terms that appear in all emails and weight is assigned to all n terms by using TF-IDF method (Salton and Buckley, 1988) (described in detail in section 2.2). Then, the clustering is applied to these terms by calculating four types of similarity: text similarity for email content using cosine similarity coefficient, subject similarity which calculates the overlap of the set of words between two email subject line, people similarity compares the set of people participating in a topic with the set of people to which the email is addressed and thread similarity which measures the percentage of emails in the cluster which are in the same thread. This similarity is then combined and normalized to find the decision score to decide which emails belong to which cluster (Details are explained further in section 2.6.2). The problem identified was that BuzzTrack's algorithm does not include any feature selection method (feature selection is a selection of relevant words from the email that can best represent the content of email. For example, in sentence "Please find assignment #5 attached" word "assignment", "attached" are more important than word "please" for sentence representation. This is usually being done by calculating some quality measure for each word and then selecting to use only the top-N features of the rank. Details are explained in section 2.3), which can significantly reduce the time taken for execution of the program.

Secondly, Kernel-selected clustering algorithm (Yang et al., 2010) proposed for the purpose to detect topics from the email clusters for automatic folder creation. In this approach emails are pre-processed by removing the stop words and stemming of the data, then email vector is created using Vector Space Model (Salton et al.1975) and by combining the body and subject terms of email for clustering by proposed advance K-Means algorithm, a kernel selected algorithm based on the lowest similarity. In this clustering method the initial cluster center is chosen randomly, for example, E2. Then all other cluster centers are chosen by taking similarity between two emails, where similarity (Cosine similarity coefficient) is minimal among all and similarity should be less than the pre-defined threshold β. Lastly, algorithm follows the basic K-Means clustering (Lloyd, 1982). Detailed description is given in section 2.6.5.

The main limitation of both BuzzTrack and Kernel-selected clustering algorithm (Yang et al., 2010) is that they fail to find those emails when user does not have clear idea of the topic of the email.

The problems with the currently automatic folder creation systems for handling email overload can be divided into two parts: Functional and Scientific. Functional deficiency is the problem that the current systems (ACEMS (Schuff et al., 2006), BuzzTrack (Cselle et al., 2007), Automatic Non-parametric clustering (Xiang et al., 2009), EGM (Ayodele et al., 2010) and Kernel-selected clustering (Yang et al., 2010)) lack in some of the tasks needed for email management. They are:

a. Folder categorization and email summarization individually do not give specific contribution to handling email overload when there is vague idea of email sender and vague idea of topic.

b. After the folder creation, if a folder contains 2000 emails on the same topic then again it is a tedious job to find the desired email. For example, if a folder is created named “Assignment” and it contains 1500 emails all with assignment information, it is hard to find an email of a specific individual whose name is not specifically known by the user to search, which in-turn does not give specific contribution to reducing email overload and is time consuming.

The scientific deficiency is the problem with the existing algorithms. They are:

c. There is a lack of intelligent system for data extraction from the email for purpose of feature selection. Feature selection is a selection of relevant words from the email that can best represent the content of email. For example, in sentence “Please find assignment #5 attached” word “assignment”, “attached” are more important than word “please” for sentence representation. This is usually being done by calculating some quality measure for each word and then selecting to use only the top-N features of the rank.

d. There is a lack of semantic relation between the words of two messages, for example, the synonyms are not taken into account when dealing with similarity.

e. Most of the algorithms are using vector space model but it has the disadvantage that the vector dimension is very large compared to the number of words in a short text or sentence which makes the vector terms have many null values, which in turn is inefficient for sentence representation (Li et al., 2006b). Details with example is shown in section 2.2

1.6 Thesis Contributions

This thesis proposes an email mining technique for solution to email overload problem based on email clustering and summarization. The approach applies data mining techniques for automatically creating folders based on the similarity between emails, then creating sub-folders based on the people sending the emails (or on the email IDs) and lastly creating the index of the folders name. Folder summary is also created which extracts frequent patterns from the emails of respective folders.

The following are main contributions of this thesis:

1. Proposed AEMS (Automatic Email Management System) model which manages email by organizing similar email in the folders (model 1 named AEG), then again organizes emails of each folder into subfolder (model 2 named APEG) where subfolder will contain emails sent by only a particular person and lastly creating the index.
2. Defined a term named Associative Term Frequency and introduced a feature selection method based on the associative frequency of the term in the whole email dataset for model AEG (Automatic Email Grouping).
3. Proposed Semantic Non-parametric K-Means++ clustering for folder creation, which overcomes the problem of random seed selection and pre-defined cardinality (k) of clustering in K-Means and K-Means++ clustering algorithms by selecting the seed according to the email weight and deciding the cardinality according to the similarity between the email content respectively, along with adding semantics to the clusters.
4. Proposed algorithm for APEG (Automatic People based Email Grouping) model which creates sub-folders based on the sender of the email.
5. Proposed method for indexing the folders created, which contain name and to the folders, sub-folders and also contains Apriori based folder summarization which

contain the important keywords from the folder which is helpful in identification of content of folder.

1.7 Outline of Thesis

The remainder of the thesis is organized as follows: Chapter 2 reviews related work to this thesis. Chapter 3 details discussion of the problem addressed along with the new algorithms proposed. Chapter 4 gives performance analysis and experiments. Chapter 5 draws the conclusion of this research and discusses future work.

CHAPTER 2

RELATED WORK

As discussed earlier email overload is a big problem faced by the email users. It is being researched ((Xiang et al., 2009) and (Cselle et al., 2007)) that automatic folder creation is a part of email mining which can immensely reduce email overload. Record management is the technique to manage emails as an archive or as information that can be useful for future. It is just a systematic organization for long term storage and access of email archive (e.g., project, department, and thesis), and the establishment of appropriate records series and records maintenance schedules. Mainly when taken the overlook to the trend of research for email mining the work usually starts from extraction of data and the selection of features before applying data mining techniques such as classification or clustering for semi-automatic or automatic folder creation of email messages.

2.1 Email Dataset

Email dataset is a large and structured set of email messages which is used to do statistical analysis and hypothesis testing for research purposes in the field of email mining. There are many types of email dataset available including Enron email dataset (Klimt and Yang, 2004), BC3 (British Columbia Conversation Corpus) (Ulrich et al., 2009), W3C dataset used for TREC Enterprise Track (Balog and de Rijke, 2006), Attachment Prediction Dataset (Klimt and Yang, 2004) and Enron Sent dataset (Klimt and Yang, 2004). Depending on the purpose of research the data set can be chosen for working.

Enron email dataset given by (Klimt and Yang, 2004) is a most popular dataset used for the purpose of email management. This dataset is a public, cleaned and contain large set of email messages organized in folders and contain more than 200 thousand messages belonging to 158 users. For automatic folder creation to reduce email overload this dataset was used by (Kulkarni and Pedersen, 2008), (Li et al., 2006a), (Cselle et al., 2007), (Ayodele et al., 2010), and (Nagwani and Bhansali, 2010). Second popular data set used is 20 Newsgroups given by (Lang, 1995) which is just a collection of 20 newsgroup document contained in almost 20 different newsgroups some of which are closely related and some are highly unrelated. It is designed mainly for the purpose of text classification and text clustering. This data set was used by (Yang et al., 2010) and (Kulkarni and Pedersen, 2008).

2.2 Email Pre-processing and Representation

The email consists of both structured information i.e., email header and unstructured information such as subject and body of the email. Any type of manipulation is done on the unstructured information available in the message. Pre-processing of raw data is the first step for any email management task. In this step the email header, email body and attachments are parsed. From the parsed data subject and email content or other fields (sender-ID, receiver ID etc) are extracted. Once the required text is obtained from the server, we need to remove stop words such as ‘the’, ’for’, ‘of’ etc from the data obtained. Next, stemming algorithm can be used to stem the data. For example, words ‘connection’, ’connecting’, ’connected’ will be converted to ‘connect’. The plug-in can be developed and installed, which can fulfill the purpose of fetching and parsing the data or for the whole system development. Such plug-in can be developed with the help of JavaMail, Visual Studio Tools or by using C++ language components with python. JavaMail is an open source java API (Application Programming Interface, which is an interface used by software components to communicate with each other) used to send or receive email. Later versions also allow fetching data from the emails.

Once the data is being extracted from the email it is then represented in some format. The most prevalent model for representation is the vector space model (Salton et al., 1975). In this model every email message is represented by a single vector and each element as a token or feature. In such type of data the tokens are usually words or phrases. These tokens can be categorized mainly into three categories: Unigram, Bigram and Co-occurrence.

Unigram: unigrams are just significant individual words. For example, in a data such as "hello dear, how are you?" the unigrams are 'hello', 'dear', 'how', 'are' and 'you'.

Bigram: bigrams are pair of two adjacent words. For example, in a data such as "hello dear, how are you?" the bigrams are 'hello dear', 'dear how', 'how are' and 'are you'. Besides, in bigrams word sequence is important. 'hello dear' and 'dear hello' are two different units.

Co-occurrence: Co-occurrences are same as bigrams but the only difference is, here the word sequence is not important. For example, 'hello dear' and 'dear hello' are treated as single unit as their order does not matter. There is another feature called target co-occurrence which is same as co-occurrence with one target word inside each pair.

Once the tokens are found the weights are assigned, these can have Boolean value (0 or 1) in order to denote the presence or absence of the token in the document, or weight (usually between 0 and 1) to denote the importance of the token for the document (Androutsopoulos et al., 2000). Now the email e_i can be represented as a vector, $e_j = (w_{1j}, \ldots, w_{nj})$, where weights w_{1j} to w_{nj} are the weights of tokens for the particular email j, In weight w_{ij}, index i refers to token t_i in collection of tokens. Suppose we use words as tokens, then t_i refers to the i^{th} word of a set of distinct words (Katakis et al.2006). The weights can be assigned by any of the information retrieval methods. Some of these are TF (Term Frequency), TF-IDF (Term Frequency – Inverse Document Frequency) or cosine normalization, which is used when there, is needed to map the values of the weight into the interval. TF is used to determine how often a term occurs in a document. The TF-IDF (Salton and Buckley, 1988) is determined when TF is combined with inverse document frequency (IDF, is a measure of whether the term is common or rare across all documents, obtained by dividing the total number of documents by the number of documents containing the term (DF – Document Frequency), and then taking the logarithm of that quotient). TF-IDF is given by the following formula in Equation 2.1:

$$\mathrm{W_{ij}} = \mathrm{TF - IDF}\,(\mathrm{t_i}, \mathrm{e_j}) = \mathrm{TF_{i,j}} * \log(\mathrm{N}/\mathrm{DF_i}) \tag{2.1}$$

Where, $\mathrm{W_{ij}}$ is the weight of token $\mathrm{t_i}$ in email j.

$\mathrm{TF - IDF}\left(\mathrm{t}_i, \mathrm{e_j}\right)$ is the TF-IDF of term $\mathrm{t_i}$ in email $\mathrm{e_j}$.

$\mathrm{TF_{i,j}}$ is the term frequency i.e., number of time token $\mathrm{t_i}$ occurs in email $\mathrm{e_j}$.

N is the total number of emails in the dataset.

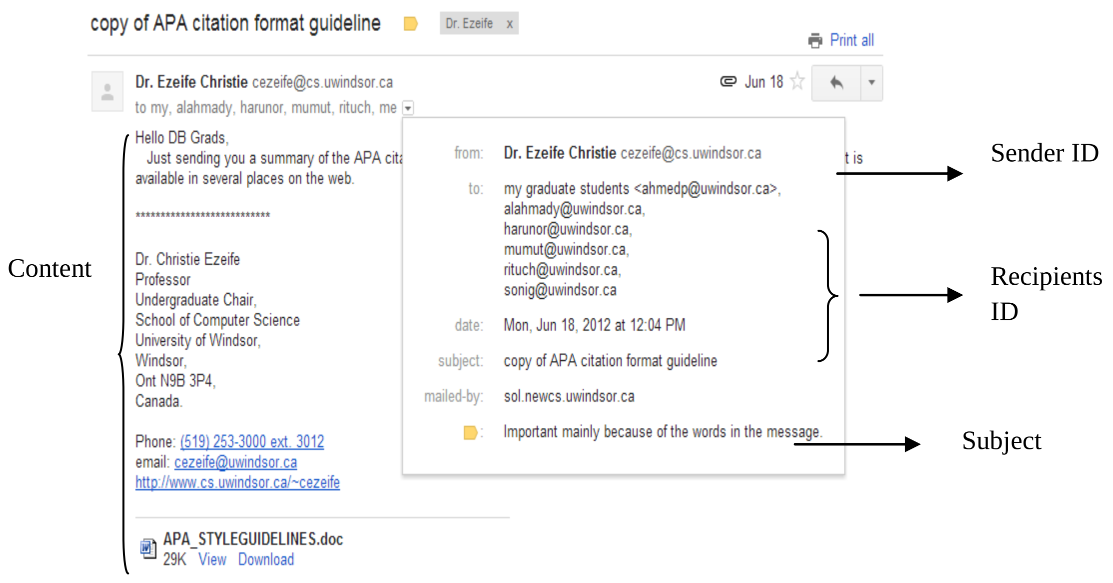
$\mathrm{DF_i}$ is the document frequency i.e., number of emails in which token $\mathrm{t_i}$ occurs.

Weight (W_{ij}) of token t_i in email e_i can also be given by cosine normalization (Sebastiani, 2002). It is used to map the values of the weight into the [0,1] interval and given by the Equation 2.2:

$$W_{ij} = TF - IDF(t_i, e_j) / \sqrt{\sum_{s=1}^{|V|} (TF - IDF(t_i, e_j)^2)} \qquad (2.2)$$

Where, |v| is the size of the vocabulary or the number of words in dataset, $TF - IDF\ (t_j, e_j)$ is a TF-IDF of term t_j in email e_j according to Equation 2.1 and W_{ij}: It is a weight of token i.e., the weight of token t_i in email j.

For example, Figure 2.1 shows an example of email received and the way it appears on the user screen and the contents of it.


copy of APA citation format guideline Dr. Ezeife x

Print all

Dr. Ezeife Christie cezeife@cs.uwindsor.ca Jun 18

to my, alahmady, harunor, mumut, rituch, me

Hello DB Grads,
Just sending you a summary of the APA cit t is
available in several places on the web.

****

Dr. Christie Ezeife
Professor
Undergraduate Chair,
School of Computer Science
University of Windsor,
Windsor,
Ont N9B 3P4,
Canada.

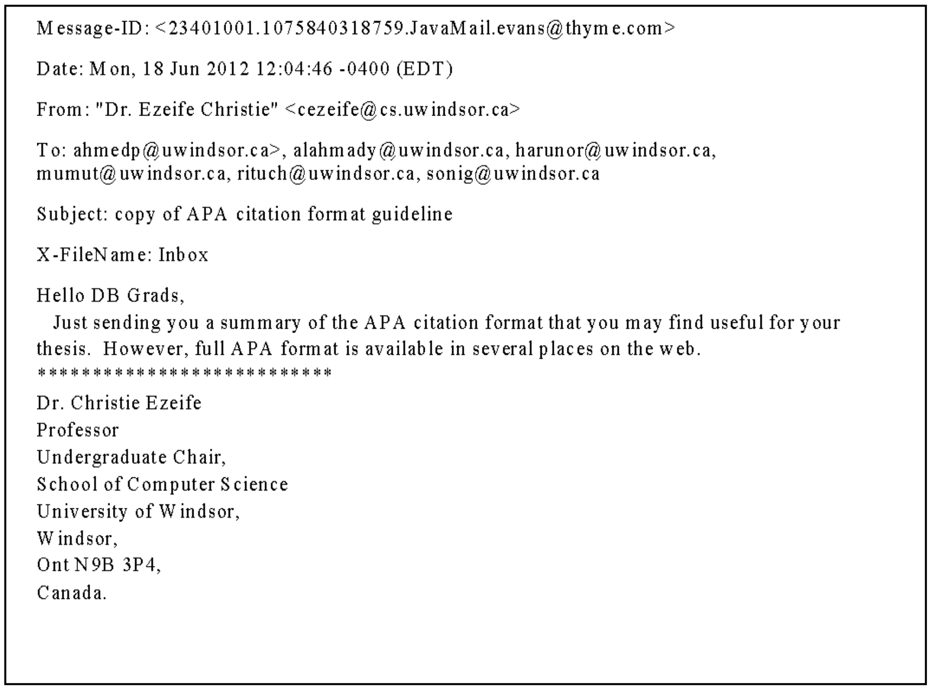
Phone: (519) 253-3000 ext. 3012
email: cezeife@uwindsor.ca
http://www.cs.uwindsor.ca/~cezeife

APA_STYLEGUIDELINES.doc
29K View Download

from: Dr. Ezeife Christie cezeife@cs.uwindsor.ca
to: my graduate students <ahmedp@uwindsor.ca>, alahmady@uwindsor.ca, harunor@uwindsor.ca, mumut@uwindsor.ca, rituch@uwindsor.ca, sonig@uwindsor.ca
date: Mon, Jun 18, 2012 at 12:04 PM
subject: copy of APA citation format guideline
mailed-by: sol.newcs.uwindsor.ca
Important mainly because of the words in the message.

Figure 2.1: Email as shown on User Screen

After pre-processing through any API (Application Programming Interface) the parsed information is as shown in Figure 2.2.


Message-ID: <23401001.1075840318759.JavaMail.evans@thyme.com>

Date: Mon, 18 Jun 2012 12:04:46 -0400 (EDT)

From: "Dr. Ezeife Christie" <cezeife@cs.uwindsor.ca>

To: ahmedp@uwindsor.ca>, alahmady@uwindsor.ca, harunor@uwindsor.ca, mumut@uwindsor.ca, rituch@uwindsor.ca, sonig@uwindsor.ca

Subject: copy of APA citation format guideline

X-FileName: Inbox

Hello DB Grads,
Just sending you a summary of the APA citation format that you may find useful for your thesis. However, full APA format is available in several places on the web.
****
Dr. Christie Ezeife
Professor
Undergraduate Chair,
School of Computer Science
University of Windsor,
Windsor,
Ont N9B 3P4,
Canada.

Figure 2.2: Parsed Email

Once the weights are assigned to the tokens many email management methods ((Schuff et al., 2006), (Manco et al., 2008), (Zeng et al., 2008) and (Yang et al., 2010))used Vector Space Model given by (Salton et al., 1975) to represent each email as a vector which is helpful for further processing.

In vector space model the terms of subject and body with their respective weights are combined to form an email vector, giving the terms of subject more weight because they are more important than terms of email content. For Figure 2.2, after pre-processing and removal of stop words, also by assigning the weights, the email vector will be:

E_j = (copy: w_1, APA: w_2, citation: w_3, format: w_4, guideline: w_5, summary: w_6, thesis: w_7 ….. Web: w_n)

For example, the TF-IDF for terms in email E_j are as shown in Table 2.1

Table 2.1: Example of TF-IDF

Term (t)	$TF_{i,j}$	$IDF_{i,j}$	**Weight (W_{ij}) as TF-IDF**
copy	1	1.65	1.65
apa	6	2.08	12.48
citation	1	1.62	1.62
format	2	1.5	3.0
guideline	1	1.65	1.65
summary	6	2.08	12.48
thesis	2	1.62	3.24
web	2	1.5	3.0

Therefore, email E_j will be represented as follow:

E_j = (copy: 1.65, APA: 12.48, citation: 1.62, format: 3.0, guideline: 1.65, summary: 12.48, thesis: 3.24 ….. web: 3.0)

The limitation of vector space model is that there is a loss of correlation and context of each term which are very important in understanding the document. Secondly, since the vector dimension is very large compared to the number of words in a short text or sentence which make the vector terms having many null values, which in turn is inefficient for sentence representation (Li et al.2006b).

Regardless of using the vector space model, another method for tokenization is Natural Language Processing (NLP) (Li et al.2006a). These are software, which include group of libraries, frameworks, and applications for symbolic, statistical natural language and speech processing. NLP tools typically do sentence detection, tokenization, POS-tagging, text chunking, lemmatization, co reference analysis and resolution, and namedentity detection among others. Some NLP tools are Apache OpenNLP (Apache OpenNLP - Download, 2011) and Text Engineering Software Laboratory (Tesla) used for Eclipse IDE (Integrated development environment for java developers).

2.3 Feature Selection

Feature selection is a selection of relevant words from the email that can best represent the content of email. For example, in sentence “Please find assignment #5 attached” word “assignment”, “attached” are more important than word “please” for sentence representation. This is usually being done by calculating some quality measure for each word and then selecting to use only the top-N features of the rank. There are many feature selection methods such as, TF-IDF. It is a method mostly used for feature selection purpose, for email dataset it should be used for finding the importance of the term in the whole dataset, which can be calculated as explained before in section 2.2. Now, after calculating the TF-IDF either all terms are considered as features or terms having the value greater or equal to certain user defined threshold, is taken as features for further processing. This method was adopted by (Kulkarni and Pedersen, 2008), (Cselle et al., 2007), (Xiang et al., 2009).

2.4 Distance Finding Methods

After the feature selection the email clustering is done for folder creation. So, when applying any clustering method on text data there is need to find the distance between two data points for the purpose of finding which data point is assigned to which cluster. There are many methods including Euclidean distance, Cosine similarity coefficient and semantic similarity.

2.4.1 Euclidean Distance

The Euclidean distance $d(p, q)$ between two data points p = ($p_1, p_2 ... p_n$) and q = ($q_1, q_2 ... q_n$) is given by Equation 2.3:

$$d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.3}$$

For example, consider two emails vectors E1 and E2 with their respective weights for each token (weights can be assigned as shown previously in section 2.2, page 31), as shown in Table 2.2

Table 2.2: Example for weights for finding Euclidean distance

E-mail#	Hello(x_1)	Report(x_2)	School(x_3)	Money(x_4)	Exam(x_5)
E1	0.2	0.4	0.1	0.3	0.7
E2	0.1	0.3	0.5	0.2	0.1

Here, 0.2 represents weight for term ‘Hello’ in email E1. Therefore,$d(E1, E2)$

$$= \sqrt{(0.1 - 0.2)^2 + (0.3 - 0.4)^2 + (0.5 - 0.1)^2 + (0.2 - 0.3)^2 + (0.1 - 0.7)^2}$$

= 0.55

The limitation (Markov and Larose, 2007) of using this method is that this method greatly depends on the length of the vector to be compared and when finding the similarity between two documents, the documents should also be close in the vector space. For example, when finding the similarity between one very large document and one small document there will be a great distance even though they both are very similar to each other.

2.4.2 Cosine Similarity Coefficient

The cosine coefficient for similarity is a degree of similarity between two vectors attained by measuring the cosine of the angle between them, where two vectors are representation of documents. In information retrieval, the cosine similarity of two documents will range from 0 to 1, therefore, the angle between two term frequencies vectors cannot be greater than 90° because the term frequencies (number of times a particular term occurs in particular email) cannot be negative, thus the lowest value for TF-IDF will be zero. The cosine similarity measure can be computed in two ways: Binary cosine similarity measure and weighted cosine similarity measure.

Binary Cosine Similarity Measure (C_{jq}): A word form receives a score of 1 when it appears in a document and 0 when it does not appear. The similarity between an email j and q is given in Equation 2.4.

$$C_{jq} = \frac{|\mathrm{Wj}\ intersection\ \mathrm{Wq}|}{\sqrt{\sum_{i=1}^{n} Wij^2} * \sqrt{\sum_{i=1}^{n} Wiq^2}} \qquad (2.4)$$

Where, |Wj $intersection$ Wq| is the number of tokens common in email j and email q,

W_{ij} is a weight of token i.e., the weight of token t_i in email j; and

W_{iq} is a weight of token i.e., the weight of token t_i in email q.

Weighted Cosine Similarity Measure (C_{jq}): The similarity between a document j and q is given in Equation 2.5.

$$C_{jq} = \frac{\sum_{i=1}^{n} Wij * Wiq}{\sqrt{\sum_{i=1}^{n} Wij^2} * \sqrt{\sum_{i=1}^{n} Wiq^2}} \qquad (2.5)$$

The weighted cosine similarity measure is used in approaches given by (Cselle et al., 2007), (Zeng et al., 2008) and (Yang et al., 2010).

For example, consider Table 2.1

$$C_{jq} = \frac{(0.1 * 0.2 + \ 0.3 * 0.4 + \ 0.5 * 0.2 + \ 0.2 * 0.3 + \ 0.1 * 0.7)}{\sqrt{0.1^2 + 0.3^2 + 0.5^2 + 0.2^2 + 0.3^2} * \sqrt{0.2^2 + 0.4^2 + 0.2^2 + 0.3^2 + 0.7^2}}$$

$= 0.590$

The limitation of using this measure is the lacks of correlation and does not consider the semantics while finding the text similarity.

2.4.3 Semantic Text Similarity (STS) method

The method proposed by (Islam and Inkpen, 2008), determine the similarity of two texts from semantic and syntactic information they contain. There are three main similarity functions to determine the text similarity. Firstly, string similarity and semantic word similarity; secondly, optimal common-word similarity function; and finally above string similarity, semantic similarity and common-word order similarity are combined with normalization.

2.4.3.1 String Similarity between Words

For finding the string similarity between two words r_i and s_j, they calculated normalized longest common subsequence (NLCS)(r_i, s_j) given in Equation 2.6,

$$NLCS(r_i, s_j) = \ v_1 = \frac{length\left(LCS(r_i, s_j)\right)^2}{length(r_i) * length(s_j)} \qquad (2.6)$$

Where, r_i and s_j are the words from the sentence which we need to find the similarity. NLCS is calculated to take into account the length of both the shorter and longer string. **For example**, consider r_i = albastru and s_j = alabaster, then LCS(r_i, s_j) = albastr therefore, NLCS(r_i, s_j) = $7^2/(8 \times 9) = 0.8$

Secondly, Normalized Maximal Consecutive Longest Common Subsequence starting at character 1 (NMCLCS1(r_i, s_j)) is given in Equation 2.7,

$$NMCLCS_1(r_i, s_j) = v_2 = \frac{length\left(MCLCS_1(r_i, s_j)\right)^2}{length(r_i) * length(s_j)} \tag{2.7}$$

In our example, MCLCS1(r_i, s_j) = al therefore, NMCLCS1(r_i, s_j) = $2^2/(8 \times 9) = 0.05$

Thirdly, Normalized Maximal Consecutive Longest Common Subsequence starting at any character n (NMCLCS$_n(r_i, s_j)$) is given in Equation 2.8:

$$NMCLCS_n(r_i, s_j) = v_3 = \frac{length\left(MCLCS_n(r_i, s_j)\right)^2}{length(r_i) * length(s_j)} \tag{2.8}$$

In our example, MCLCSn(r_i, s_j) = bast therefore, NMCLCSn(r_i, s_j) = $4^2/(8 \times 9) = 0.2$

And then these measures are combined to get the string similarity ($\propto$) between strings word r_i and s_j, which is given by Equation 2.9.

$$\propto = w_1 v_1 + w_2 v_2 + w_3 v_3 \tag{2.9}$$

Here, w_1, w_2, w_3 is the weights and $w_1 + w_2 + w_3 = 1$.

String similarity, S = $\alpha 1 v1 + \alpha 2 v2 + \alpha 3 v3 = 0.33 \times 0.8 + 0.33 \times 0.05 + 0.33 \times 0.2 = 0.275$

2.4.3.2 Sematic Similarity between Words

To find the semantic similarity between words (Islam and Inkpen, 2008) used *Semantic PMI (Point-Wise Mutual Information) similarity* function. Semantic PMI similarity between the two words, w_1 and w_2 given in Equation 2.10.

$$Sim(w_1, w_2) = \left(\frac{f(w_1,w_2,\beta_1)}{\beta_1}\right) + (\frac{f(w_2,w_1,\beta_2)}{\beta_2}) \tag{2.10}$$

Where, $f(w_1, w_2, \beta)$ is the β- PMI summation function for word $w_1$with respect to word w_2 and is given in Equation 2.11:

$$f(w_1, w_2, \beta) = \sum_{i=1}^{\beta}(f^{pmi}(X_i^{w_1}, w_2))^{\gamma} \tag{2.11}$$

Where, $X^w$is the set of words sorted in descending order by their PMI values with w and taken the top β words having $f^{pmi}(t_i, w) > 0$ and is given in Equation 2.12:

$$f^{pmi}(t_i, w) = \log_2(\frac{f^b(t_i,w)*m}{f^t(t_{i,})*f^t(w)}) \tag{2.12}$$

Where, $f^t(t_{i,})$ is the type t_i appeared in the entire dataset, $f^b(t_i, w)$ is the times word t_i appeared with word w and m is the total number of tokens in dataset.

For example, (Islam and Inkpen, 2008) consider two sentences:

P = "A cemetery is a place where dead people's bodies or their ashes are buried."

R = "A graveyard is an area of land, sometimes near a church, where dead people are buried."

Step 1: After eliminating all stop words and lemmatizing, we get P = {cemetery, place, where, dead, body, ash, bury} and R = {graveyard, area, land, sometime, near, church, where, dead, bury} where m = 7 and n = 9.

Step 2: Three tokens (where, dead and bury) in P exactly matches with R therefore, δ = 3. Now remove these three from both P and R. So, P = {cemetery, place, body, ash} and R = {graveyard, area, land, sometime, near, church}. As m− δ ≠ 0, so proceed to next step.

Step 3: Construct a 4*6 string matching matrix, M1, as shown in Table 2.3. Consider the words {place, land} pair where length(LCS(place, land)) = 2, η = 5 is the length of the longer token (place), τ = 4 is the length of the shorter token (land) and 2 is the maximal length of the consecutive portions of the shorter token that consecutively match.

Table 2.3: String Similarity Matrix

	Graveyard	area	Land	sometimes	near	church
Cemetery	0.023	0.021	0	0.129	0.052	0.041
Place	0.037	0.083	0.132	0.017	0.033	0.022
Body	0.018	0	0.041	0.021	0	0
ash	0.024	0.083	0.055	0.028	0.055	0.037

Step 4: Construct a 4 * 6 semantic similarity matrix by calculating SOCPMI method as shown in Table 2.4.

Table 2.4: Semantic Similarity Matrix with SOCPMI

	Graveyard	area	Land	sometimes	near	church
Cemetery	0.986	0	0.390	0.195	0.542	0.856
Place	0	0.413	0.276	0.149	0	0
Body	0.465	0	0.363	0.122	0.063	0.088
ash	0.796	0	0.213	0.238	0.395	0.211

Step 5: Construct a 4 × 6 joint matrix as shown in Table 2.5, M and assign equal weight factor by setting both ψ and ϕ to 0.5.

Table 2.5: Joint Matrix

	Graveyard	area	Land	sometimes	near	church
Cemetery	0.505	0.010	0.195	0.162	0.297	0.449
Place	0.018	0.248	0.204	0.083	0.0.17	0.011
Body	0.242	0	0.039	0.071	0.032	0.044
ash	0.416	0.041	0.134	0.133	0.225	0.124

We find the maximum-valued matrix-element, $\gamma_{i\,j}$ = 0.505 and add it to ρ as $\gamma_{i\,j} \geq 0$. So, ρ = {0.505}. The new M after removing ith (i = 1) row and j th (j = 1) column is, as shown in Table 2.6:

Table 2.6: Maximum-valued Matrix

	area	Land	sometimes	near	church
Place	0.248	0.204	0.083	0.0.17	0.011
Body	0	0.039	0.071	0.032	0.044
ash	0.041	0.134	0.133	0.225	0.124

We find the maximum-valued matrix-element, γi j = 0.248 for this new M and add it to ρ as γi j ≥ 0. So, ρ = {0.505, 0.248}. The new M after removing ith (i = 1) row and j th (j = 1) column is, as shown in Table 2.7:

Table 2.7: Maximum-valued Matrix element

	area	Land	sometimes	near	church
Body	0	0.039	0.071	0.032	0.044
ash	0.041	0.134	0.133	0.225	0.124

Here, 0.225 is the maximum-valued matrix-element and γi j ≥ 0. So, ρ = {0.505, 0.248, 0.225}. The new M after removing ith (i = 2) row and j th (j = 3) column is, as shown in Table 2.8:

Table 2.8: Final Maximum-valued Matrix

	Land	sometimes	near
Body	0.039	0.071	0.032

We find 0.071 as the maximum-valued matrix-element and $\gamma i\, j \geq 0$. So, ρ = {0.505, 0.248, 0.225, 0.071}. The new M is empty after removing ith (i = 1) row and j th (j = 2) column. We proceed to next step as $m - \delta - |\rho| = 0$. (Here, m = 7, δ = 3 and $|\rho|$ = 4)

Step 6: $S(P, R) = (3 + 1.049) \times 16/126 = 0.514$

2.4.3.3 Common Word Order Similarity between Sentences

The Common Word Order Similarity is then measured is the denominator, the largest possible dissimilarity value (normalized difference Common Word Order). Suppose, P and R has a and b tokens respectively. There are δ tokens in P exactly matches in R and from P these tokens are moved to X and from R moved to Y. Then, Common Word Order Similarity is given by the Equation 2.13:

$$S_0 = \begin{cases} 1 - \left(\frac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2}\right) \ if\ \delta\ is\ even \\ 1 - \left(\frac{2\sum_{i=1}^{\delta} |x_i - y_i|}{\delta^2 - 1}\right) \ if\ \delta\ is\ odd, \delta\ \neq 1 \\ 1\ \ if\ \delta = 1. \end{cases} \tag{2.13}$$

We sum up all elements in ρ and add $M = \delta(1 - w_f + w_f S_0)$ to get the total score. Now by multiplying M by the reciprocal harmonic mean of m and n, we get the balance score between 0 and 1, inclusively. The score is given by Equation 2.14

$$S(P,R) = \frac{M*(m+n)}{2mn}. \tag{2.14}$$

2.5 Data Mining Techniques for Email Mining

2.5.1 Clustering Methods

Clustering is one of the data mining methods which are used for the purpose of email mining. Clustering is used for automatic folder creation for the purpose to reduce email overload. Most commonly used email clustering methods used are Hierarchical clustering

(Schuff et al., 2006) and K-means clustering by (Surendran et al., 2005) and (Nagwani and Bhansali, 2010).

2.5.1.1 K-Mean Clustering Algorithm

This is the most widely used clustering algorithm proposed by (Lloyd, 2006). K-Means clustering algorithm mainly has four steps. For example, we have 6 emails (E1,.., E6) with the terms having their respective weights as shown in Table 2.9 and we want to group those emails in two clusters.

Table 2.9: Example for input weight to clustering algorithm

E-mail#	Hello(x_1)	Report(x_2)	School(x_3)	Money(x_4)	Exam(x_5)
E1:	.2	.4	.1	.3	.7
E2:	.1	.3	.5	.2	.1
E3:	.5	.4	.1	.3	.9
E4:	.1	.1	.5	.4	.2
E5:	.2	.4	.1	.7	.3

Step1: First randomly two samples or email vectors will be selected as initial cluster center. It could be any two emails from E1 to E5. As we are considering 2 clusters that means K=2. Assume in our case randomly chosen centers are E2 and E5.

Step2: Assign other emails to their closest cluster center. Closeness of any two emails can be determined by any distance algorithm such as Euclidian distance, squared Euclidean distance, Mahalanobis distance (Mahalanobis, 1936) and Cosine similarity coefficient as explained in section 2.4. Most popular distance algorithm used is Euclidian distance.

So, according to our example one email will be picked from E1, E3 and E4. Assume E1 is picked first. Next, the distance between E1 and E2, and E1 and E5 will be calculated and E1 will be assigned to that cluster of which distance is lower. If the distance calculated is lower in E1 and E2, then E1 will be assigned to E2.

Assume E1, E2 and E4 are clustered together and E5 and E3 clustered separately. In this step we need to find the mean (M1) of E1, E2 and E4 for one cluster and mean (M2) of E5 and E3 in another cluster. So, new centers will be M1 and M2.

E1
E2
E4
E5
E3

Figure 2.3: Clustered Email

Step3: In this step we need to find the new centers for clustering which will be the arithmetic mean of existing clusters. In our example, the new cluster center will contain the values, as shown in Table 2.10.

Table 2.10: Example for computation of new cluster center

New Cluster Center	Hello(x_1)	Report(x_2)	School(x_3)	Money(x_4)	Exam(x_5)
C1:	0.133 ((.2 + .1 + .1)/3)	0.267 ((.4 + .3 + .1)/3)	0.367 ((.1 + .5 + .5)/3	0.3 ((.3 + .2 +.4)/3	0.33 ((.7 + .1 + .2)/3
C2:	0.35 (.5+ .2)/2	0.4 (.4+ .4)/2	0.1 (.1+ .1)/2	0.5 (.3+ .7)/2	0.6 (.9+ .3)/2

Hence the cluster centers will be C1 and E5.

Step4: Again each email is compare with these new cluster centers and is assigned to those clusters where the distance is minimum.

Suppose, again E1, E2 and E4 are clustered together and E5 and E3 clustered separately. Since new clusters form are identical to the previous one. Therefore the algorithm stops and finally we have two clusters: Cluster 1 containing E1, E2 and E4 and Cluster 2 containing E3 and E5.

Limitation of using K-Means clustering algorithm is that it is sensitive to initialization and to the presence of outliers.

2.5.1.2 K-Mean++ Clustering Algorithm

The algorithm was first proposed by (Arthur and Vassilvitskii, 2007). The K-means++ algorithm differs from the general K-means algorithm in first step where we need to select the cluster centers. The algorithm works as follows:

Step1: In the first step, it will compute the similarities between every two emails and will take those two emails as two centers, which have minimal similarity among all. Similarity between two emails is computed using any distance formula which is explained before. In our example, we have six emails (E1, E2, E3, E4, E5 and E6). So according to this algorithm we will compute the distances between every two emails as shown in Table 2.11:

Table 2.11: Similarity between two emails

Email#	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
E1	1	.2	.11	.2	.12	.2
E2	.2	1	.12	.082	.21	.4
E3	.11	.12	1	.5	.13	.6
E4	.2	.082	.5	1	.31	.65
E5	.12	.21	.13	.31	1	.6
E6	.2	.4	.6	.65	.6	1

Here we can see that the minimal similarity is .082 and which is between E4 and E2. So our cluster center will be E2 and E4.

Step 2: These three steps are exactly same as explained before in section 2.5.2 in K-Mean from step 2 to step 4.

The advantages of K-Means++ clustering algorithm over K-Means clustering algorithm is that K-Means++ clustering is 10% more efficient then K-Means in term of cluster quality (Arthur & Vassilvitskii, 2007) and K-Means++ clustering algorithm computational time is much less than the K-Means clustering algorithm. The limitation of K-Means++ clustering algorithm is that it is still based on random selection of seed.

2.5.1.3 Hierarchical Clustering

This is a "bottom up" approach, each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. For *n* given data sets, this procedure reduces them to *n - 1* mutually exclusive set by considering the union of all possible *n (n -*

1)/2 pairs and selecting a union having a maximal value for the objective function (ESS). For example, suppose a set of six email messages {E1, E2, E3, E4, E5 and E6}:

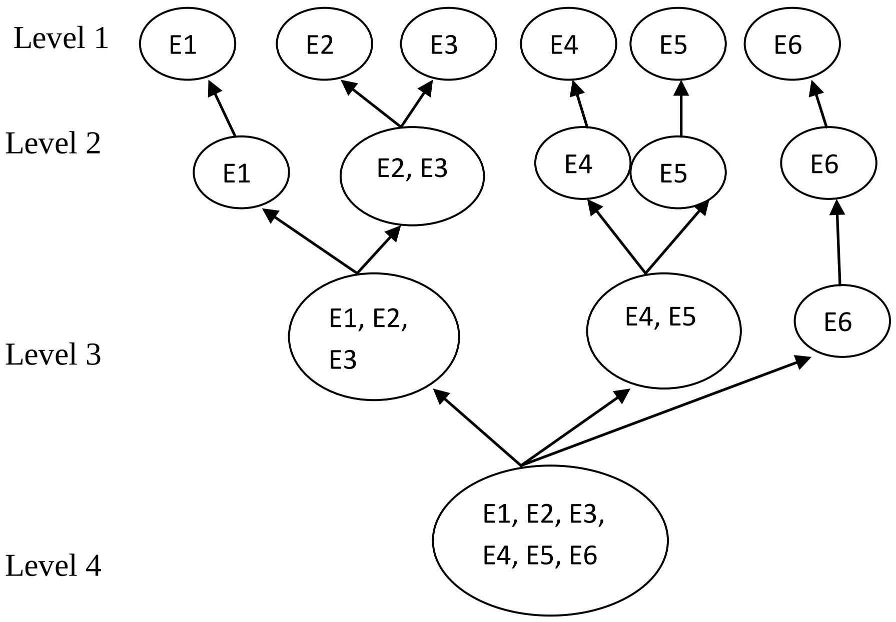


Figure 2.4: Example for Hierarchical Clustering

2.5.2 Frequent Pattern Mining, Association and Correlation

Frequent patterns (Han and Kamber, 2000) are itemsets, subsequences, or substructures that appear in a data set with frequency no less than a user-specified threshold. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent itemset. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern. A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently in a graph database, it is called a (frequent) structural pattern. Finding frequent patterns plays an essential role in mining associations, correlations, and many other interesting relationships among data.

2.5.2.1 Apriori Algorithm

Apriori is a data mining technique for frequent itemset mining which discovers strong associations or correlation relationships among data (Ezeife & Lu, 2005). Let the data of transactions consist of the sets {1,2,3,4}, {1,2}, {2,3,4}, {2,3}, {1,2,4}, {3,4}, and {2,4} and the value 3 is the support threshold (if it appears in at least 3 transactions of the data then itemset is frequent).

Step 1: First count up the number of occurrences (support), of each item separately. We get:

Table 2.12: Example output of 1st Iteration of Apriori

Item	Support
{1}	3
{2}	6
{3}	4
{4}	5

All the itemsets of size 1 have a support of at least 3, so they are all frequent.
Step 2: The next step is to generate a list of all pairs of the frequent items:

Table 2.13: Example output of 2nd iteration of Apriori

Item	Support
{1,2}	3
{1,3}	1
{1,4}	2
{2,3}	3
{2,4}	4
{3,4}	3

Now, {1,3} and {1,4} are not frequent because there threshold is less than the minimum support.

Step 3: In this way, we can *prune* sets: we will now look for frequent triples in the database, but we can already exclude all the triples that contain one of these two pairs:

Table 2.14: Example output of 3rd iteration of Apriori

Item	Support
{2,3,4}	2

So, the frequent itemset are {1,2}, {1,3}, {1,4}, {2,3}, {2,4} and {3,4}. These frequent itemset can be used to find association or correlation among the data.

2.6 Automatic Folder Creation Tools

As discussed earlier email overload is a big problem faced by the email user. It is being researched that record management by automatic folder creation is a part of email management which can immensely reduce the email overload. Record management is the technique to manage emails as an archive or as information that can be useful for future. It is just a systematic organization for long term storage and access of e-mail archive (e.g., project, department, and thesis) and the establishment of appropriate records series and records maintenance schedules. Mainly when taken the overlook to the trend of researches for email management the work usually starts from extraction of data and the selection of features as discussed before and then the techniques such as classification was applied automatic or semi-automatic folder creation. In later research the clustering was taken into account.

2.6.1 Email Clustering by adding Semantics

In this GSP-PCL clustering algorithm proposed by (Li et al.2006a) the email clustering is done on the bases of subject of email in unsupervised manner where Generalized Sentence Patterns (GSP - e.g. {“person”, “seminar”, “date”} which means that some “person” gives seminar on specific “date”) are made which are treated as the pseudo class labels and conduct the semi-supervised email clustering. Now these GSPs are mined using the frequent closed item set mining technique after the generalization of terms by using the NLPWin tool (Heidorn, 2000). Then the GSPs grouping and selection is done on the bases of minimum confidence. Lastly, on the bases of GSP a clustering algorithm named GSP-PCL, used to form the pseudo class label and to cluster the email messages together. The algorithm works as follows:

Step 1: Firstly we need to generalize the terms in the subject of each to produce the generalized sentence. This generalization performed based on the type of noun. For example, if the subject is “welcome Tom Cruise”. Then the generalized term called factoids will be ‘person’ as the noun Tom Cruise is person. So, the generalized sentence will be GSP = {‘welcome’, ‘Tom’, ‘Cruise’, ‘Person’}. Here each subset of GS is also a generalized sentence. This generalization can be done by Microsoft Software named NLPWin Tool (Heidorn, 2000).

Step 2: In next step we need to group the GSPs according to its similar frequent terms. GSPs in the same group will represent the same cluster. Similarity between two GSPs *p* and *q* can be determined by Equation 2.15:

$$Sim(p,q) = \begin{cases} 1, \ \frac{\sup(q)}{\sup(p)} > \min\_conf \\ \quad 0, \ otherewise \end{cases} \tag{2.15}$$

Here, *sup(p)* and *sup(q)* means the support of *p* and *q* respectively and *min_conf* is a pre-defined constant (according to the experiment 0.5 and 0.8 give good results on all dataset tested).

Step 3: Construct *n* number of pseudo classes using GSP groups,

$D_i^0 = \{d | d \in D \ and \ d \ match \ G_i\}$ and *i* =1, 2,…., n

Where, D_i^0 is the set of GSP classes, *d* is each GSP class and G_i is a GSP group.

We need to find those particular data (emails) which are common in every GSP group. For example, if four GSP groups containing of:

GSP -1: {E1, E2, E3, E4}

GSP -2: {E1, E5, E3, E7}

GSP -3: {E1, E6, E3, E8}

GSP -4: {E1, E10, E3, E9}

Now among all these four groups email E1 and E3 is present. Therefore, GSP classes in E1 and E3.

Step 4: Construct all GSP classes, $D^` \Leftarrow D - \cup_{i=1}^{n} D_i^0$

Where, D is the email dataset, D_i^0 is the GSP classes and $D^`$ is a set of classes for classification. So, in our example $D_i^0 = \{E1, \ E3\}$

Step 5: Now for each d in $D^{`}$ classify d into D_i^0, if $P(D_i^{j-1} \mid d)$ is the maximal posterior probability and $P\left(D_i^{j-1} \mid d\right) >$ min_$class_prob$. For example, $P\left(D_i^{j-1} \mid E1\right) = 0.7$ and min_$class_prob$ = 0.3. Then E1 is taken as the final class for classification.

Step6: Use basic k-means to partition D_{other} into (k-n) clusters, where k is pre-defined.

The limitation of the process is that firstly, they are considering only the subject line, which plays an important role in email clustering but body of the email also plays a vital role which is totally ignored in this approach. Secondly, they are using K-means clustering algorithm which is based on pre-defined cardinality.

2.6.2 Multi-Level Clustering using ACEMS clustering algorithm

Email overload reduction approach proposed by (Schuff et al., 2006) is method to represent emails in folders. They first introduced the concept of multi-attribute and multi weight clustering named ACEMS clustering. This approach is almost same as hierarchical clustering. But, the difference is in representation of data. In this approach different email attributes such as subject, from, to, and body of email can be weighted differently according to user choice. For example, if the email subject is "appointment" then the term 'appointment' will be counted 5 instead of 1. Next, for clustering, they used hierarchical clustering (Ward Jr, 1963) explained in section 2.5.3. The ACEMS algorithm works as follows:

Step 1: Retrieve email message from inbox using JavaMail API.

Step 2: Apply weight to each attribute (subject, body, sender, and receiver) of the email. For example, suppose weight assigned as shown in Table 2.15

Table 2.15: Weights assigned to email attributes

Attribute	**Weights**
To	4
Cc	6
From	0
Subject	6
Body	8

Step 3: Cluster weighted message using the Ward clustering algorithm (Ward, 1963) as shown in section 2.5.3. Suppose, there are six emails in the data set then input to the clustering algorithm will be email vector having weight for key phases given by the IBM intelligent Miner (Tkach, 1998) as shown is Table 2.16:

Table 2.16: Weight for key phrase in each email

E-mail#	T1	T2	T3	T4	T5
E1:	3	6	0	3	1
E2:	7	3	5	0	4
E3:	2	0	3	8	5
E4:	4	3	0	4	3
E5:	0	7	3	6	0
E6:	1	3	5	1	5

Step 4: Graphical tool is build using Java applets.

The advantages of using this method is that emails are arranged in multi-level hierarchy, i.e. folders and sub-folders but they does not consider the feature selection for the purpose of clustering which can greatly saves the computational time, which is a major limitation of the approach. Secondly, no provision can be made for a relocation of objects that may have been 'incorrectly' grouped at an early stage. Different distance metrics most likely result in different clusters. Performing multiple experiments and comparing the results is recommended.

2.6.3 Buzz Track

Authors (Cselle et al., 2007) developed a system for reducing the email overload by proposing novel approach for clustering related email in a folder by detecting the topic automatically and named the system as "BuzzTrack". The steps involve the pre-processing with the same method explained before, by using Python and JavaScript language. The representation of the email is in the form of vector where each email is represented as a vector and weight is provided with the modified TF-IDF formula given in Equation 2.16:

$$W_{i,j} = \begin{cases} \left(1 + \log(tf_{i,j})\right) \log\left(\frac{N}{df_i}\right) & if\, tf_{i,j} \geq 1 \\ 0 & if\, tf_{i,j} = 0 \end{cases} \tag{2.16}$$

Here, the $tf_{i,j}$ is taken as $tf_{i,j} \geq 1$ or $tf_{i,j} = 0$ because $tf_{i,j}$ is the total number of term occurred in the document, which is always an integer number.

Then the clusters are made by finding the similarity between two emails based on three measures:

a. Text Similarity: Here, we need to find the similarity based on the term frequency using Cosine similarity algorithm on the bases of TF-IDF.
b. Subject Similarity: A second measure is also text-based and measures subject similarity. It calculates the overlap between the set of words S_i, S_j in the subject lines of two emails. The formula given in Equation 2.17:

$sim_{subject}(m_i, m_j) = 2|S_i intersection\ S_j|/|S_i| + |S_j|$ (2.17)

Here, $|S_i intersection\ S_j|$ is the number of tokens common in email i and email j,

For example, we have two subject sets of two different emails as following:

S_i = {'hello', 'assignment', 'professor', 'exam', 'score', 'grade'}.

S_j = {'hello', 'computer', 'student', 'exam', 'car', 'grade', 'school'}.

So, the subject similarity will be (2*3)/ (6+7).

c. People-based similarity: Here we need to find two people similarity metrics that compare the set of people participating in a topic with the set of people to which the email is addressed. For an email m_i, we derive a set ppl(m_i) with all email addresses in the 'From', 'To', and 'Cc' headers. Similarly, for each topic cluster C_k, there is a set of senders ppl(C_k) which contains all email addresses from all emails in the cluster. The two people-based similarity measures are given in Equation 2.18 and 2.19:

$$sim_{people,subset}(m_i, C_k) = \frac{|ppl(m_i)\ intersection\ \ ppl(C_k)|}{|ppl(m_i)|} \quad (2.18)$$

$$sim_{people,overlap}(m_i, C_k) = \frac{2|ppl(m_i)\ intersection\ \ ppl(C_k)|}{|ppl(m_i)|+|ppl(C_k)|} \quad (2.19)$$

Where, $ppl(m_i)\ intersection\ \ ppl(C_k)$ is the number of people name common in emailm_i and email C_k,

Lastly, topic is detected from the cluster. For topic detection word extracted should be noun and also having the highest TF.IDF in that cluster.

The limitation of approach proposed was that the feature selection is not taken into consideration, which is a core part to an email clustering approach and method is only based on text similarity.

2.6.4 Automatic Nonparametric Clustering Approach for Folder Creation

This approach proposed by (Xiang et al., 2009) is an automatic email clustering system, which uses simple modification of K-means and hierarchical clustering algorithm. In this approach validation of clusters is performed simultaneously with the clustering. The algorithm is as follows:

Step1: Construct a data matrix using vector space model. For example, consider a data matrix as shown in Table 2.17.

Table 2.17: Data matrix

E-mail#	T1	T2	T3	T4	T5
E1:	.2	.4	.1	.3	.7
E2:	.1	.3	.5	.2	.1
E3:	.5	.4	.1	.3	.9
E4:	.1	.1	.5	.4	.2
E5:	.2	.4	.1	.7	.3
E6:	.1	.3	.5	.8	.5

Here, E1 and E2 is the email vector representing two emails calculated using Vector Space Model (Salton et al.1975) and weights are assigned using TF-IDF, formula is given by Equation 2.20:

$$x_i = (n_i / \sum_k n_k) * \log(N/n_i) \tag{2.20}$$

Where, x_i is the recalculated weight for each term in each email, n_i is the number of emails in which the index term t_i appears and denominator is the number of occurrence of all terms and N is the total number of emails.

Step 2: Similarity matrix is created by just calculating the similarity using Euclidean distance measure between each email to other. For example, as shown in Table 2.18

Table 2.18: Similarity matrix

Email#	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
E1	1	.2	.11	.2	.12	.2
E2	.2	1	.12	.082	.21	.4
E3	.11	.12	1	.5	.13	.6
E4	.2	.082	.5	1	.31	.65
E5	.12	.21	.13	.31	1	.6
E6	.2	.4	.6	.65	.6	1

Step 3: Select a seed or cluster center randomly. For example, E1 is chosen to be the first seed.

Step 4: Add or delete point into the currently generated cluster with Hubert's Γ statistic (Xiang et al., 2009) validity test. Hubert's Γ statistic can be calculated by measuring the correlation between X(i,j) and Y(i,j), where value of Γ for each email can be find using the Equation 2.21. The value of Γ is scaled between 0 to 1 for information retrieval because for calculating the weight of term they used TF-IDF, since the term frequencies (TF-IDF weights) cannot be negative therefore Γ cannot be negative. Also, larger the Γ means stronger evidence of their belongingness in a same cluster.

$$\Gamma = \left(\frac{1}{\frac{\mathrm{n(n-1)}}{2}}\right) * \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (\frac{X(i,j)-\bar{X}}{\sigma_x})(\frac{Y(i,j)-\bar{Y}}{\sigma_y}) \qquad (2.21)$$

Where, $\overline{X}$ and $\overline{Y}$ are the means of the values in X and Y, σ_x and σ_y are the standard deviations in X and Y and n is the total number of terms in email X.

Namely, $\sigma_x{}^2 = \left(\frac{1}{\frac{\mathrm{n(n-1)}}{2}}\right) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (X(i,j) - \bar{X})^2$,

$$\sigma_y{}^2 = \left(\frac{1}{\frac{\mathrm{n(n-1)}}{2}}\right) \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (Y(i,j) - \bar{Y})^2$$

Where, $\bar{X} = (1/n) \sum_{i=1}^{n} X_i$ and $\bar{Y} = (1/n) \sum_{i=1}^{n} Y_i$

Also,

$$\mathrm{Y(i,j)} = \begin{cases} 1, & \text{if email i and j are clustered together} \\ 0, & \text{if email i and j are not clustered together} \end{cases}$$

Similarly, we can find X(i, j).

For example, E2 is chosen then distance is calculated to E1 and Γ is calculated, suppose Γ = 0.1, which is very less so E2 will be checked with other center if present else E2 will be new cluster center.

Step5: Repeat step 5 until no point can be allocated.

Step6: Repeat step 4 to 6 until all the clusters are generated.

The advantages of using this approach is that there are no iteration which saves the computational time and secondly, it optimizes the quality of clustering all the time as validation is a part of clustering algorithm.

The limitation of the algorithm is that it is there no consideration of feature selection which can it turn reduce the computational time and also it is based on random selection of seed.

2.6.5 Automatic Topic Detection Algorithm by Clustering

This algorithm is proposed by (Yang et al.2010) for the purpose to detect topics from the email clusters. In this approach the email is pre-processed by removing the stop words and stemming of the data then after preprocessing the email vector is created to label the email combining the body and subject of email. Lastly the advance k-means algorithm is used to cluster the emails and design a kernel selected algorithm based on the lowest similarity.

Suppose there are preprocessed email messages (stop words are removed and stemming is already done on the each email). Now body and subject features are selected based on the formula that if the term appears in the subject then weight is assigned as 1/n where n is the number of terms in the subject, else if the term is in body then weight is given in Equation 2.22:

$$b_{wi} = f_{wi} / \sum_{j=1}^{n} f_{wj} \qquad (2.22)$$

Where, f_{wi} means the frequency of w_i and bw_i is the weight of the word in the body. Once initial weight is given then we combine email subject and body and provide the email term weight. The email weight E_w of the each term is based on three factors:

i. If term available only in subject, $E_w = \alpha * S_w$ (2.23)

ii. If term available only in body, $E_w = (1-\alpha) * b_w$ (2.24)

iii. If term available in both subject and body, $E_w = \alpha * S_w + (1-\alpha) * b_w$ (2.25)

Variable α is calculated by the finding the similarity by cosine similarity measure between standard vector space and compare with each of subject and body respectively.

Now, email vector is constructed by giving the global measurement to each feature selected and based on that clustering is done by finding the cosine similarity between two email vectors. It is calculated using Equation 2.26:

$$EwIDw(e,t) = \begin{cases} Ew(e,t) * \frac{logN}{Dw(t)} & Ew(e,t) > 0 \\ 0 & Ew(e,t) = 0 \end{cases} \tag{2.26}$$

Where, *Ew(e,t)* is the term weight of *t* in the email *e*, *Dw(t)* is the number of emails that contain *t.*

Lastly, the clustering is done according to the advance k-mean algorithm, this clustering algorithm is similar to the K-Mean++ clustering algorithm as explained before just the difference is in place of finding the distance they used cosine similarity measure. The threshold *β* is used to select the kernel. The topic is then detected by choosing the top words whose weights of corresponding dimension are ranked high to label the topics. The evaluation of the approach presented is done by finding the recall, precision and f-measure.

For example, suppose there is a set five emails {E1, E2, E3, E4, E5}.

Step 1: For each email subject and body vectors are created. For example, in email E1 the vector representation of subject (E_s) and body (E_b) is given by:

Terms in email subject, E_s: {t1, t2} and Terms in email body, E_b: {t1, t3, t4} and frequency of terms t1 -> 4, t3 -> 2, t4 -> 2. Hence the standard vector *S* will be {t1, t2, t3, t4}.

Step 2: Combining subject and body terms. For E1, initial weight of each term will be

For Terms in Email Subject, E_{sw}: {1/2t1, 1/2 t2, 0t3, 0t4} (By the Equation 2.23)

For Terms in Email Body, E_{bw}: {1/2t1, 0t2, 1/4t3, 1/4t4} (By the Equation 2.24)

Now, when we combine email vector from email subject and body terms there is a need to find α, which represents the cosine similarity coefficient. Here, α = 0.6.

Now, by the method described above the weighted feature terms of an email will be:

E1: {1/2t1, 1/2t2, 3/4t3, 3/4t4} (By the Equation 2.25). Similarly we can find for other email also. Suppose, *E2:* 1/2t1, 3/4t2, 1/4t5, 1/8t6; *E3:* 1/8t4, 1/5t5, 2/7t7, 3/7t8; *E4:* 1/3t3, 1/2t5, 1t7, 1/5t8; *E5:* 1/4t2, 1/6t4, 2/3t6.

Step 3: The final email data representation shown in Table 2.19 will be given by Equation 2.26:

Table 2.19: Example of Email Vectors

Email#	T1	T2	T3	T4	T5	T6	T7	T8
E1	0.4	0.5	0.6	0.71	0	0	0	0
E2	0.8	0.9	0	0	0.1	0.2	0	0
E3	0	0	0	0.3	0.4	0	0.25	0.4
E4	0	0	1.2	0	0.45	0	0.4	0.15
E5	0	0	0	0	0	0	0	0.75

Step 4: The clustering of emails are done using advanced K-Means clustering algorithm.

a. Choosing initial cluster center randomly, for example, E2.
b. Will take those two emails as another centers, which have minimal similarity (Cosine similarity coefficient) among all and similarity should be less than the pre-defined threshold β. Suppose, E4.
c. Follows the basic K-Means clustering, as explained in section 2.5.1.

Step 5: Folders are created, where each folder represents each cluster and topic is detected to identify the folder contain. For topic detection top word is chosen whose weights of corresponding dimension are ranked high to label the topics. For example, here if E1, E2 and E3 belong to same cluster the topic will be “T2”.

The limitation of using this approach is that they used the vector space model which has mainly two disadvantages: Firstly, loss of correlation and context of each term which are important in understanding and grouping the document and secondly, it is inefficient for sentence representation because the vector representing the sentence contains many null value.

The main limitation of all of the above approaches for handling email overload problem mentioned in section 2.6 is that they fail to find specific email when there is a vague idea of email sender and vague idea of topic.

CHAPTER 3

EMAIL OVERLOAD MANAGEMENT SYSTEM VIA AUTOMATIC FOLDER CREATION AND INDEXING.

As discussed in section 1.4, our goal is to reduce email overload by semantically arranging similar emails in folders and sub-folders and indexing them for the purpose of easy searching and retrieval of a particular email. In this thesis, a model is defined which automatically and semantically indexes and arranges email in similar group.

3.1 Problem Addressed in Managing Personal Email

1. Existing work, such as BuzzTrack (Cselle et al., 2007), Folder Creation Approach based on Concept Vector Space Model (Zeng et al., 2008), Automatic topic detection algorithm by clustering (Yang et al., 2010) which automatically or semi-automatically reduce email overload problem by folder creation and topic detection. However, these algorithms do not consider the semantic similarity between two emails and do not consider the sub-folder and indexing of folder. This problem can be sub-divided into following issues:

a. Similarity finding algorithm, such as Cosine similarity, which finds similarity between text documents when represented in the form of vector. The main problem is the loss of correlation and measure for semantic similarity between the documents and also they are inefficient for sentence representation.

b. Clustering algorithm, such as K-Means++ (Arthur and Vassilvitskii, 2007), by which it is able to predict the group of similar documents/emails. In this algorithm, there is a random generation of the seed i.e. first cluster center is chosen randomly, which cannot always give optimal solution because it is yet based on approximation for first seed generation. Secondly, in K-Means++ clustering algorithm the cardinality of clustering (the number of clusters) is predefined but when using an unlabeled data it is hard to say the exact number of clusters needed.

c. Feature selection algorithm which selects terms for clustering by calculating the associativity of that term in the whole bunch of email messages.

2. Sub-folder creation in the main folder created according to the similarity of the content, on the bases of email sender ID and keeping the email from that specific person in that respective folder. For example, a main folder is created named "Assignment" and having 15 emails of which 10 are from "Dr. Chen" and 5 from "Dr. David". Then 2 sub-folders should be created named "Chen" and "David" and these folders should contain respective emails.
3. Separate index creation which contain name of folder and its sub-folder and contain the links, for searching. It is like an index of a book containing link to the particular email.

3.2 Automatic Email Management System (AEMS)

This section discusses the system AEMS which mines data from the email and cluster email in the specific group and sub-group of similar email and person respectively and produce index by folder summarization. The method for building AEMS model is divided into three modules: Automatic Email Grouping (AEG); Automatic People based Email grouping (APEG) and Indexing. The process flow of AEMS module is provided in Figure 3.1.

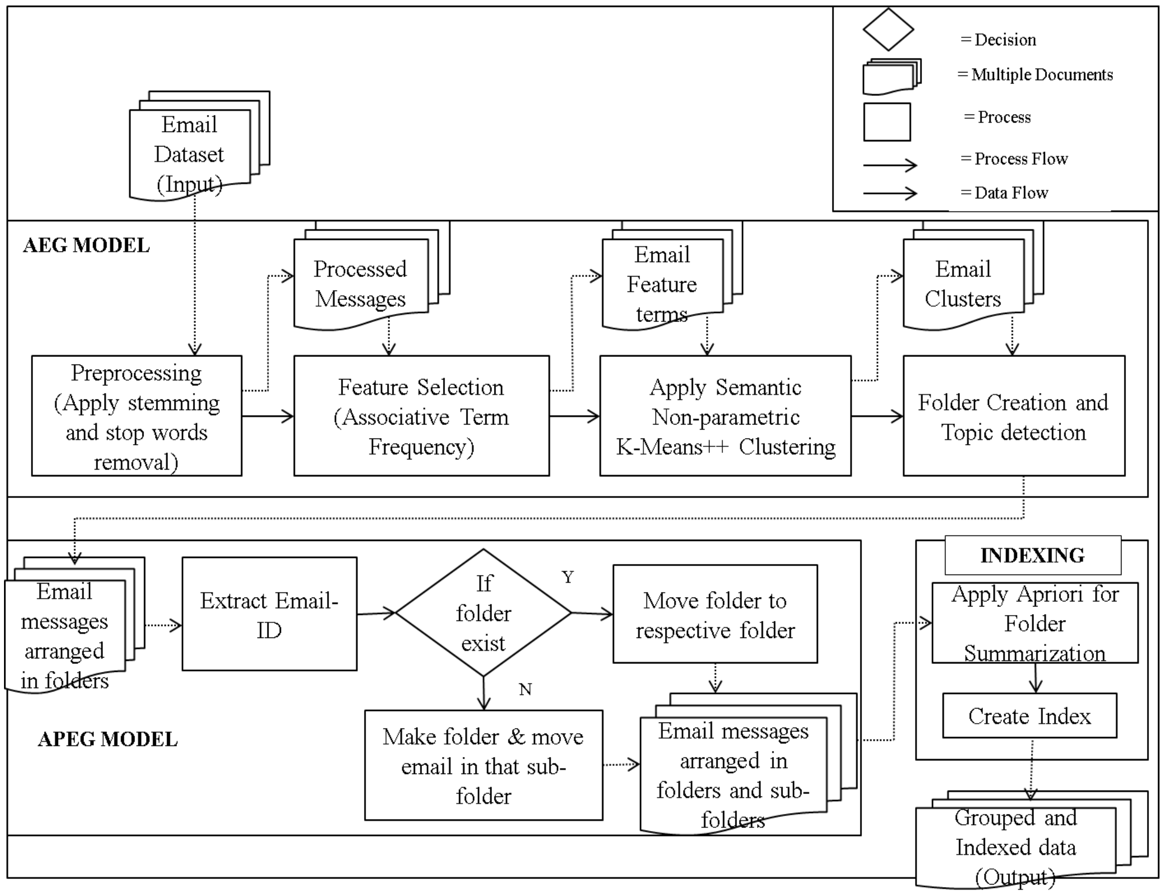

= Decision
= Multiple Documents
= Process
= Process Flow
= Data Flow
Email Dataset (Input)
AEG MODEL
Processed Messages
Email Feature terms
Email Clusters
Preprocessing (Apply stemming and stop words removal)
Feature Selection (Associative Term Frequency)
Apply Semantic Non-parametric K-Means++ Clustering
Folder Creation and Topic detection
Email messages arranged in folders
Extract Email-ID
If folder exist
Y
N
Move folder to respective folder
Make folder & move email in that sub-folder
Email messages arranged in folders and sub-folders
APEG MODEL
INDEXING
Apply Apriori for Folder Summarization
Create Index
Grouped and Indexed data (Output)

Figure 3.1: Process flow of AEMS Model

In AEMS module: First, the AEG_algorithm (given in Figure 3.5) is called, as shown in step 1 of AEMS_system in Figure 3.2. The group of email data (arranged in text files where each file represents a single email and all information related to that email) is given as input to the AEG_algorithm, which semantically organizes similar email in a meaningful group by creating a folder for each group. Secondly, the APEG_algorithm (given in Figure 3.7) is called, as shown in step 2 of AEMS_system in Figure 3.2. The group of email given by AEG serves as input to APEG, which again group emails of each folder based on the sender email ID. Lastly, the Indexing method (given in Figure 3.9) is called, as shown in step 3 of AEMS_system in Figure 3.2, where index or view will be created in a separate web page, which contain link to the respective folder and sub-folder and contain the summary of each folder. The main algorithm of AEMS_system is given in Figure 3.2.

```
Algorithm: AEMS_system() – {Automatic Email Management System: Manages emails by folder creation and summarization}
Input: A set of email messages in text files (EM), where each text file represents a single email message.
Other Variables:
P: String containing sender of email
Output: Folders as directory (F) and sub-folders as directory (SF) containing similar and people based email messages and an index file (IF) containing link to particular email.
Begin

    1. F = Call AEG_algorithm (EM) // Semantically organizes emails into meaningful groups – Figure 3.5 on page 69
    2. SF = Call APEG_algorithm (P) // Sub-group creation based on email sender ID – Figure 3.7 on page 71
    3. IF = Call Indexing (F) // Create index as HTML web page, which contains summary of folder content and link to respective group, subgroup and email – Figure 3.9 on page 73
End
```

Figure 3.2: Algorithm for AEMS

3.2.1 AEG Model – An Automatic Folder Creation and Topic Detection Module

AEG_algorithm given in Figure 3.5 is a method called by AEMS_system in step 1 of Figure 3.2. AEG is a process of creating main folder based similar email messages and semantic similarity measures which includes the details of pre-processing and representation, feature selection, improved version of K-Means++ clustering algorithm and topic detection. By the systematic arrangement of email messages in folders gives the user with organized representation which in turn gives ease to user with retrieval of emails. The process flow for AEG model is given in Figure 3.3.

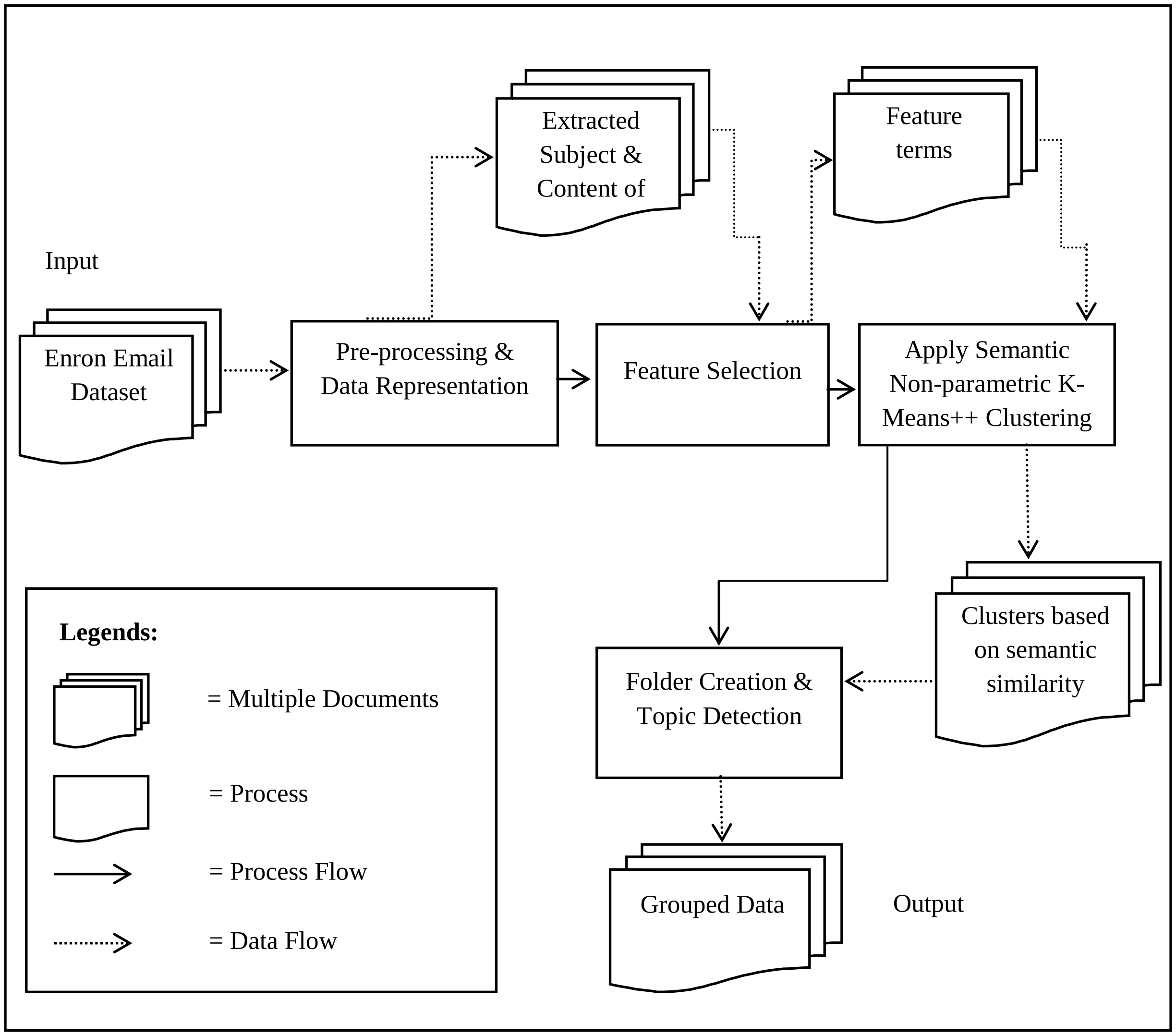

Extracted Subject & Content of
Feature terms
Input
Enron Email Dataset
Pre-processing & Data Representation
Feature Selection
Apply Semantic Non-parametric K-Means++ Clustering
Clusters based on semantic similarity
Folder Creation & Topic Detection
Grouped Data
Output
Legends:
= Multiple Documents
= Process
= Process Flow
= Data Flow

Figure 3.3: Process Flow for AEG model

The whole process for the AEG model consists of 4 stages, which are explained below:

Preprocessing

In the step 1 of the AEG_algorithm of Figure 3.5 for the AEG model, which extract subject line and content from each email and remove the stop words, this can reduce the size of the documents to be processed and the features selected.

Feature Selection and Email Representation

In the step 2 of the AEG_algorithm of Figure 3.5, review the features and represent data from the email. In this step take each term from the processed data and calculate the associative term frequency (R_{tf}) of a particular term x as shown in AEG_algorithm of Figure 3.5 step 2.1. R_{tf} is the number of emails that contains the term, x. Select features as shown in Figure 3.5 step 2.2, where R_{tf} should be greater or equal to the pre-defined threshold, T_s and T_b depending on whether the term appears in subject or content of the email respectively. R_{tf} is given in Equation 3.1.

Associative Term frequency, $R_{tf}(x)$ = (100 * df_x)/N (3.1)

Where, N is the total number of email messages in the dataset; df_x is the total number of emails in which the term x appeared. Advantage of using the associative term frequency over simple TF is that, for example, if there is a term “Assignment” in “Email 1” for 5 times and in “Email 2” it occurs only 3 times then its TF will be 8 but if there is a term “appointment” which appears in 6 email messages and only ones in each email then its TF will be 6. Therefore in this case the term “assignment” will get more weight over term “appointment”. The term “assignment” is important for a particular email whereas the term “appointment” is important in the whole document. Create the email vector by combining the feature terms and removing the duplicate terms from the vector, as shown in AEG_algorithm of Figure 3.5 step 2.3.

Email Clustering using Semantic Non-parametric K-Means++ Clustering Algorithm

Step 3 of the AEG_algorithm of Figure 3.5 call Semantic_nonparametric_kmeanspp algorithm of Figure 3.3, where the emails are clustered together according to the proposed the Semantic Non-parametric K-Means++ clustering algorithm. First, select the initial cluster center by calculating the email weight as shown in Semantic_nonparametric_kmeanspp algorithm of Figure 3.5 step 1. The initial cluster center is the email with the maximum weight, where email weight is just the total number of feature terms in the email. After this, chose all other clusters center by calculating the similarity between all emails with the initial cluster center $\mathrm{D}(\mathrm{C}_{\mathrm{init}}, x_i)$ as shown in Semantic_nonparametric_kmeanspp algorithm of Figure 3.4 step 2. Chose other clusters center using a weighted probability distribution where an email x is chosen with probability proportional to $D(x_i)^2$ and $\mathrm{D}(\mathrm{C}_{\mathrm{init}}, x_i)$ is should be less or equal to β, as shown in Semantic_nonparametric_kmeanspp algorithm of Figure 3.4, step 3. The similarity $\mathrm{D}(\mathrm{C}_{\mathrm{init}}, x_i)$ (calculated using the semantic text similarity (STS) algorithm (Islam and Inkpen, 2008), explained in section 2.4.3). Once all centers are created, then form the clusters by assigning email (x_i) to the cluster center C_k where, similarity ($\mathrm{D}(\mathrm{C}_{\mathrm{k}}, x_i)$) is minimum in comparison to other center, as shown in Semantic_nonparametric_kmeanspp algorithm of Figure 3.4 step 4.

Here, proposed clustering algorithm overcomes the problem of other clustering algorithm by semantically clustering the emails in the meaningful group. Firstly, in K-Means++ clustering algorithm the first center is chosen randomly which always cannot give optimal solution because it is based on approximation whereas our proposed clustering algorithm overcomes this problem by selecting the first center based on the weight of the particular email vector in the whole dataset. Secondly, in K-Means++ algorithm the cardinality of clustering is predefined, which again make it user dependent. Therefore, Semantic Non-parametric K-Means++ algorithm gives the approach, which will automatically find the number of clusters formed. Lastly, calculated the similarity $D(x_{i,j})$ between two emails for the purpose to form the more related and meaningful group.

```
Algorithm: Semantic_nonparametric_kmeanspp(X) – {Clustering algorithm}
Input: Email vector set $X$
Other Variables:
$D(C_{init,} x_i)$: Decimal value indicating similarity between initial cluster center and the email $x_i$
$C_k$: Email vector of text represented as cluster centers
X: Email vector of text containing terms of each email
$\beta$: Minimum threshold value, where $0.0 \leq \beta \leq 1.0$
|T|: Total number of features terms
$E_w$: Integer value as email weight (Total number of features)
x: Particular email in email vector of text set X
$C_{init}$: Email vector of text represented as initial cluster center
Output: Set of clustered email represented as grouped text
Begin

    1. FOR each email $(x_i)$ in email vector set (X) DO
        1.1. Email weight of email $x_i$, $E_{wi} = |T|$ // Calculate each email weight by calculating the total number of features terms of the email.
        1.2 Initial cluster center, $C_{init} = \max(E_w)$ // Email having maximum weight is assigned as initial cluster center ($C_{init}$).
    2. FOR each email $(x_i)$ in email vector set (X) DO
        2.1 Calculate similarity $D(C_{init,} x_i)$ // Calculate the similarity between initial cluster center and the each email.
    3. Choose all cluster centers $C_k$, select $C_k = x'$ with probability
                $\left(\frac{D(C_{init,} x')^2}{\sum_{x \in X} D(C_{init,} x)^2}\right)$ and $D(C_{init,} x') \leq \beta$
    4. FOR each email $(x_i)$ in email vector set (X) DO
        4.1 Assign email $(x_i)$ to the cluster $C_k$ where, similarity $(D(C_k, x_i))$ is minimum.
            // Cluster formation - each email is assigned to cluster where the similarity is minimum of all

End
```

Figure 3.4: Algorithm for Semantic Non-parametric K-Means++ Clustering

The advantage of using this approach is that at first instance it will always consider the email vector as initial center whose number of feature term and their respective weight is maximum, which in turn can attract maximum number of associative emails. By this, we eliminate those cluster centers that have less or no cluster members because if such case arises then compare each email with that center and which in turn will increase the number of fruitless comparisons so that an inefficient cluster will be formed.

Topic Detection and Folder Creation

Create folders and assign topic to each folder, as shown in step 4 of the AEG_algorithm of Figure 3.5. We considered the term with the highest weight, M_t in the whole cluster, C_k as a folder name. M_t can be calculated using 3.2:

$$M_t = \max(|E_{i,j}|)\forall i \forall j \tag{3.2}$$

Where, $|E_{i,j}|$ is the decimal value of the weight of a term j in the email i and $E_{i,j} \in C_k$.

```
Algorithm: AEG_algorithm (X) – {Semantically organizes emails into meaningful groups}
Input: Set of all email in text format, X = {$x_1$, $x_2$,…, $x_n$}
Other variables:
$R_{tf}$: Decimal value representing associative term frequency
N: Integer value representing total number of emails in whole dataset
$df_x$: Integer value representing the total number of email where term x appeared
T: Pre-defined threshold where $0 < T < 100$
$M_t$: Integer value representing term with the highest $R_{tf}$ in the cluster $C_{i,j}$
C: Clusters formed by clustering algorithm
EV: Email vector as text containing feature terms of each email
EF: Terms in subject and body of the email
FT: Text representing selected feature terms
Output: Folders with similar emails (F) represented as directory
Begin

    1. EF = Data_preparation (X) //Extract subject and content from each email and remove stop
       words
    2. FT = Select_features (EF)
        2.1 $R_{tf}$ = Calculate $R_{tf}$ // For term $x$ in each email, $R_{tf}$ is calculated, where $R_{tf}(x)$ = (100
            * $df_x$)/N
        2.2 Select FT //Select term as feature if the term selected is from subject and $R_{tf} \geq T_s$ or
            If term is from content and $R_{tf} \geq T_b$
        2.3 Return EV // representation of selected feature terms in the vector form
    3. C = Call Semantic_nonparametric_kmeanspp (EV) // Clustering algorithm – Figure 3.4 on
       page 68
    4. F = Create_folder (C) // Creating folders by topic detection
        4.1 Select term, t where, $M_t = \max(|C_{i,j}|)\forall i \forall j$
        4.2 Create folder with topic, t
        4.3 Repeat 4.1 and 4.2 until all folders are created.
End
```

Figure 3.5: Algorithm for AEG model

3.2.2 APEG Model – An Automatic Sub-Folder Creation

APEG_algorithm given in Figure 3.5 is a method called by AEMS_system in step 2 of Figure 3.2. APEG is a process for creating the sub-groups. Sub-groups are creation based on email sender ID and contain the emails from that specific person in the respective group. Figure 3.6 gives the process flow for APEG model.

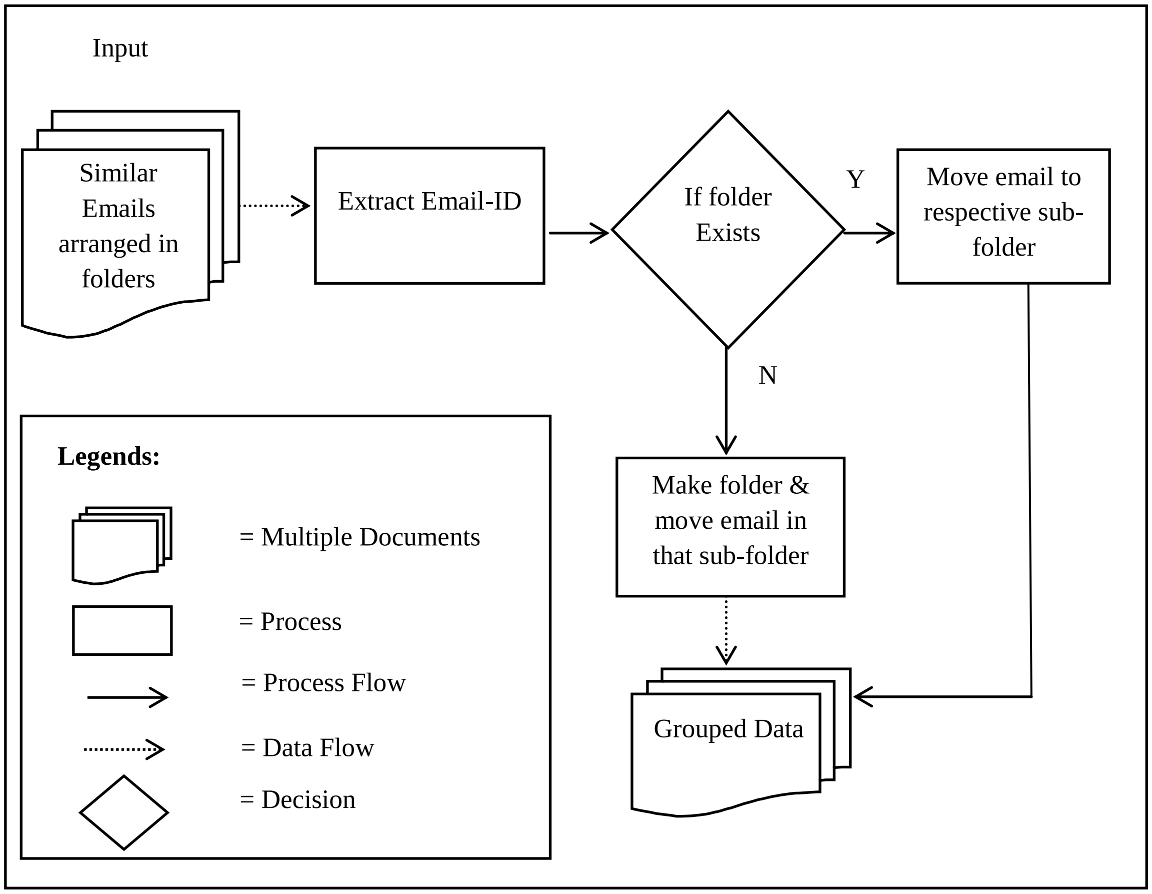


Figure 3.6: Process Flow for APEG model

The whole process of *APEG* model is divided in two steps:

Step 1: Once the folders (F) are created from the AEG model they serve as input for APEG model. So, first extract the email ID of the sender (P) from email message from each group, as shown in step 1 of algorithm APEG_algorithm in Figure 3.7.

Step 2: Then do some comparison, as shown in step 2 of algorithm APEG_algorithm in Figure 3.7.

a. If a sub-folder exists with the name, same as the sender email ID exists then move that respective email message to that sub-folder.

b. Else, create a new sub-folder with the sender email ID and move that respective email message to that sub-folder.

For example, create a main folder named "Assignment" and having 15 emails of which 10 are from "Dr. Chen" and 5 from "Dr. David". Then 2 sub-folders should be created named "Chen" and "David" and these folders should contain respective emails.

```
Algorithm: APEG_algorithm (F) – {Sub-group creation based on email sender ID}
Input: Similar email arranged in folders (F) represented in the form of directory
Other Variables:
SF: Text representing sub-folder name
N: Text representing sender's email ID
$E_n$: Email as text file
Output: Sub-folders (SF)
Begin
    1.  N = Extract (F) // Extract sender's email ID from each email
    2.  FOR each SF and each email $E_n$ DO
            4.1  F_name = get_name (SF) // Extract the sub-folder (SF) name
            4.2  IF F_name = N // Check if folder according to sender's ID exists or not
                         4.2.1    Move email $E_n$ to sob-folder (SF) named F_name //Move email
                                  to that folder (SF)
                 ELSE
                         4.2.2    Create folder named N
                         4.2.3    Move email $E_n$ to sob-folder (SF) named N //Create another
                                  folder named N and move that email to the folder (SF)
End
```

Figure 3.7: Algorithm for APEG algorithm

The advantages of using *APEG* model is that for instance, if a folder is created by the previously proposed techniques and in a single folder there is 2000 emails on the same topic then again it is a tedious job to find the desired email. For example, if a folder is created named "Assignment" and it contains 1500 emails all with assignment information. It is hard to find an email of a specific individual whose name is not specifically known by the user to search, which in-turn does not give specific contribution to reduce email overload and it is also time consuming.

3.2.3 Indexing

Indexing given in Figure 3.9 is a method called by AEMS_system in step 3 of Figure 3.2. Create index or view in a separate web page. The output of APEG which is a systematic arrangement of email messages in folder, serves as input for indexing. The Apriori algorithm is applied to data feed in each folder which will extract important keywords which in turn will provide approximate identification of content of the folder. Later, by these content users view is created which contain the names, link to the respective folders and folder identifier. The whole process of indexing can be divided in three steps (repeat following steps for all folders created):

Step 1: Feature terms of subject and content of each email from the folder are extracted using associated term frequency, as shown in step 1.1 and 1.2 of Indexing algorithm of Figure 3.9 (explained before in section 3.1.2).

Step 2: Apply Apriori algorithm (explained in section 3.3.1), to the feature terms to extract the terms that are important, as shown in step 1.4 of Indexing algorithm of Figure 3.9.

Step 3: Create index/view as HTML web page as shown in step 1.4 of Indexing algorithm of Figure 3.9, which is a hierarchical representation of groups from AEG model, sub-groups from APEG model, and contain link to each individual email. Additionally contain summary of folder contain.

Indexing provides easiness to user in searching for a particular email with vague idea of topic and sender. For example, two main folders are created named “Assignment” and

"Job". "Assignment" folder has 15 emails of which 10 are from "Dr. Chen" and 5 from "Dr. David" and "Job" folder has 2 emails from "Lisa Durante". Then create index file that contains link to the particular email with respect to position in the folder as well the folder summary created, as shows in Figure 3.8.

Index

Assignment (Summary)

- ► Dr. Chen <chen@gmail.com> (10)
- ► Dr. David <david@gmail.com> (5)

Job (Summary)

Lisa Durante <lisa.trigon@hiredesk.net> (2)

- ► Lisa Durante - Confirming Receipt of complete Questionaire – Hi Suzy, To …
- ► Lisa Durante - Trigon Computer Solution Received resume – Hi Suzy, …

Figure 3.8: Example for Indexing

The algorithm for indexing is as shown in Figure 3.9.

```
Algorithm: Indexing (F) - {Create index as HTML web page, which contains summary of folder
content and link to respective group, subgroup and email}
Input: Similar email (X) arranged in folders (F) represented in the form of directory
Other Variables:
R_tf: Decimal value representing associative term frequency
N: Integer value representing total number of emails in whole dataset
$df_x$: Integer value representing the total number of email where term x appeared
T: Pre-defined threshold where $0 < T < 100$
EF: Terms in subject and body of the email
FT: Text representing selected feature terms
S: Text representing folder summary
Output: Index and folder contain summarization (IF)
Begin
    1.  FOR each folder (F) DO
            1.1 EF = Data_preparation (X) //Extract subject and content from each email and remove
                stop words
            1.2 FT = Select_features (EF)
                    1.2.1   $R_{tf}$ = Calculate $R_{tf}$ // For term $x$ in each email, $R_{tf}$ is calculated, where
                            $R_{tf}(x) = (100 * df_x)/N$
                    1.2.2   Select FT //Select term as feature if the term selected is from subject
                            and $R_{tf} \geq T_s$ or if term is from content and $R_{tf} \geq T_b$
                    1.2.3   Return EV // representation of selected feature terms in the vector form
            1.3 S = Mine frequent terms using Apriori method.
            1.4 Create link to emails in F and the respective summary S
End
```

Figure 3.9: Algorithm for Indexing

3.3 A Walk through Example

Example 1: Given a user, u email inbox with say 6 emails from 2 senders, s1 and s2. Create topic folders F, sub-folders of sender (SF) and index i containing links to those F and SF (small size of file chosen only for illustration purposes). Consider thresholds T_s = 5%, T_b = 20% and $\beta = 0.2$, email content as follows:

Email E1 – From (Sender ID): “david@gmail.com”

Subject: “Assignment”

Content: “Hi all 60-510 students, Please find assignment #5 attached. Note that I shall try and give you the assignment a little sooner in the future.”

Email E2 – From (Sender ID): “david@gmail.com”

Subject: “Assignment”

Content: “Dear 60-510 students, There is no class today (Friday 24 February) as it is study week and the university is closed today. All but 2 students have now sent me assignment#6. I shall review the assignments and return to Mandy on Monday for you to collect on Tuesday. Please see the attached assignment#7. ”

Email E3 – From (Sender ID): “richi@gmail.com”

Subject: “Assignment”

Content: “Hi all 60-510 winter 2012 Students, Please find assignment #3 attached. Please complete the assignment and email to Dr. Frost no later than next Monday 23rd January 2012. ”

Email E4 – From (Sender ID): “john@gmail.com”

Subject: “Need an appointment”

Content: “You can audit it no problem but the room is pretty full and very hot! There no need to meet just turn up at class. You do know that you are not the GA for 315 Dr. John, Associate Professor, university of Windsor.”

Email E5 – From (Sender ID): “david@gmail.com”

Subject: “Thesis Defense”

Content: “David will be presenting his Master’s thesis defense entitled: An Approach for Intention-Driven, Dialogue-Based Web Search" Please see the following attachment for time, location and abstract. Everyone is welcome to attend. ”

Email E6 – From (Sender ID): “sonig@gmail.com”

Subject: “Appointment”

Content: “Dear Madam, I have already arrived to Windsor from India on August 27, 2011 for joining Graduate program in Computer Science and staying at Clark Residence. Now, I am seeking for your appointment to meet you. So I will be very kind if you can give few minutes from your precious time. Regards Gunjan.”

Solution 1: These emails will go as the input to the AEMS_system of Figure 3.2, which calls the AEG_algorithm of Figure 3.5, and the algorithm works as follows:

Step 1: The subject and content of the email are extracted and after eliminating all special characters and punctuation and then removing all stop words, we need to calculate R_{tf} for each term and select only those terms whose R_{tf} is greater or equal to threshold defined. So in our example,

Email E1 – Subject: “Assignment”

Content: “student find assignment attached note try assignment little sooner future.”

Step 2: Calculate R_{tf} (Associative Term Frequency) for each term and select those terms as feature terms whose R_{tf} is greater than the given threshold. Table 3.1 shows R_{tf} for each term in E1.

Table 3.1: Rtf for terms in email E1

Terms	**R_{tf}**	**R_{tf} %**
Assignment	3	50
Student	3	50
Find	1	16
Assignment	3	50
Attached	4	67
Note	1	16
Try	2	33
Assignment	3	50
Little	1	16
Sooner	1	16
Future	1	16

Step 3: Therefore, the email vector by the selected features. For email E1 is:

E1 – {Assignment, Student, Attached, Try}

Similarly, for other emails email vector will be:

E2 – {Assignment, Student, Attached, University}
E3 – {Assignment, Attached}

E4 – {Appointment, Meet, University, Windsor}

E5 – {Thesis, Defense, Attached}

E6 – {Appointment, Windsor, Meet}

Step 4: Now we will cluster the email with Semantic Non-parametric K-Means++ clustering algorithm of Figure 3.4. For this, we need to follow steps of algorithm semantic_nonparametric_kmeanspp of Figure 3.4 as given below:

a. Find email containing the maximum number of features ($\max(E_w)$) from the matrix created containing the total number of features (E_w) for each particular email.

Table 3.2: Number of Features for each email

Email	**No. of features (E_w)**
E1	5
E2	5
E3	3
E4	4
E5	4
E6	3

Here, $\max(E_w)$ = 5 which is the email weight for E1 and E2, obtained by calculating the total number of feature term in email vector. In this case, the first email will be selected as the initial cluster center. Therefore, the first initial cluster center will be E1.

b. The similarity between all other emails with email E1 is calculated, to choose the other cluster centers (Detail are also given in section 2.4.3, page 35)

For example, similarity between E1 and E2 is calculated as given below:

E1 – {assignment, student, attached, try}

E2 – {assignment, student, attached, university}

i. Here, there are four tokens in E1 and E2 both, therefore, m = 4 and n=4.

ii. Three tokens (assignment, student and attached) in E1 exactly matches with E2 therefore, δ = 3. Now remove these three from both E1 and E2. So, E1 = {try} and E2 = {university}. As m− δ ≠ 0, so proceed to next step iii.

iii. Calculate the string similarity between "try" and "university", as shown in section 2.4.3.1, page 35. Since,

NLCS (try, university) = $\frac{length(LCS(\text{try,university}))^2}{length(try) * length(university)}$ = 0.1333 according to equation 2.6 (page 35), $\text{NMCLCS1(try, university)}$ = $\frac{length(MCLCS_1(\text{try,university}))^2}{length(try) * length(univeristy)}$ = 0 according to equation 2.7 (page 36) and $\text{NMCLCSN(try, university)}$ = $\frac{length(MCLCS_n(\text{try,university}))^2}{length(try) * length(university)}$ = 0.1333 according to equation 2.8 (page 36). Therefore, according to equation 2.9 (page 36),

String similarity, M1 = $\alpha = 0.33 * (0.1333 + 0 + 0.1333) = 0.0879$

iv. Calculate the semantic similarity between "try" and "university", as shown in section 2.4.3.2, page 36. According to the six email vectors created the semantic similarity,

$$Sim(w_1, w_2) = \ (f(w_1, w_2, \beta_1)/\beta_1) + \left(\frac{f(w_2, w_1, \beta_2)}{\beta_2}\right) = 8.75524$$

(According to equation 2.10 (page 37))

Therefore, after normalizing $M2 = Sim(w_1, w_2) = 1.0$

Here, $\frac{f(w_1,w_2,\beta_1)}{\beta_1} = \frac{\sum_{i=1}^{\beta}\left(f^{pmi}(X_i^{w_1},w_2)\right)^{\gamma}}{\beta_1}$ = 0 (according to equation 2.11 (page 37)) and,

$$\frac{f(w_2,w_1,\beta_2)}{\beta_2} = \frac{\sum_{i=1}^{\beta}(f^{pmi}(X_i^{w_1},w_1))^{\gamma}}{\beta_2} = \frac{log_2\left(\frac{(1*20)}{2*2}\right)}{1.43} = \frac{2.321928}{1.43} = 8.75524$$

where, $\beta_1 = (\log(f^t(w_1))^2 * \left(\frac{\log_2 n}{\delta}\right) = 0 * \frac{3.322}{0.7} = 0$ and,

$$\beta_2 = (\log(f^t(w_2))^2 * \left(\frac{\log_2 n}{\delta}\right) = 0.30103 * \frac{3.322}{0.7} = 1.43$$

v. Calculate the joint similarity, as sown in section 2.4.3, page 35.

Joint similarity, M = ψ*M1 + ϕ*M2 = 0.5 * M1 + 0.5 * M2

= 0.5 * 0.0879 + 0.5 *1.0 = 0.54395

vi. Overall email similarity between E1 and E2,

$$S(P, R) = \frac{(M + \ \delta) * (m + n)}{2mn} = \frac{(0.54395 + \ 3) * (4 + 4)}{2 * 4 * 4} = \ 0.886$$

Table 3.3: Similarity between emails

Email	**Similarity Score ($D(x)$)**
E1, E2	0.89
E1, E3	0.75
E1, E4	0.10
E1, E5	0.34
E1, E6	0.15

Chose other clusters center using a weighted probability distribution where an email x is chosen with probability proportional to $D(x_i)^2$ and $\mathrm{D}(\mathrm{C}_{\mathrm{init}}, x_i)$ is should be less or equal to β. Here, $D\left(x_{1,4}\right) \leq 0.2$, therefore, the next cluster center will be E4 and similarly as other cluster center will be E5. That is, there will be three clusters with centers E1, E4, E5. Here, β is the threshold value and is pre-defined in the system (test is done with different value of β, to find the optimal value of β according to all dataset on which the experiment will be performed).

c. The cluster will be formed by finding the similarity between emails and the cluster centers. With the maximum similarity, that email will be assigned to the respective cluster. Hence, the clusters formed will be:

C1 – {E1, E2, E3}

C2 – {E4, E6}

C3 – {E5}

Step 5: Using these clusters, create folders with a folder name. Choose term as folder name, which have maximum R_{tf} and term should appear in the subject of any emails.

Here, for C1 – {E1, E2, E3} the terms having maximum R_{tf} and appear in subject is {Assignment = 3}. Therefore, term "Assignment" is the folder name for C1. So,

Folder 1 – Name: Assignment

Content: E1, E2, E3

Folder 2 – Name: Appointment

Content: E4, E6

Folder 3 – Name: Thesis

Content: E5

Step 6: Now these folders will serve as input for the APEG system. In APEG system, firstly, extract sender email ID. Here, for folder 1, E1 is taken and sender email ID (david@gmail.com) is extracted and a sub-folder named "david@gmail.com" is created and email E1 is moved to that particular email.

Secondly, E2 is taken and sender email ID (david@gmail.com) is extracted. Now, a check is given if folder named "david@gmail.com" already exists, if it exists then that email is moved to that folder. Therefore folder "david@gmail.com" will contain 2 email messages E1 and E2. Repeat this step for all folders created and every email message.

Therefore the output for *APEG* system is:

Folder 1 – Name: Assignment

Sub-Folders: david@gmail.com – {E1, E2}

Sub-Folders: richi@gmail.com – {E3}

Folder 2 – Name: Appointment

Sub-Folders: john@gmail.com – {E4}

Sub-Folders: sonig@uwindsor.ca – {E6}

Folder 3 – Name: Thesis

Sub-Folders: david@gmail.com – {E5}

Here, we can see that for "Thesis", the title does not indicate the specific content in the folder. Therefore, to overcome this problem we summarize the content of folder by applying "Indexing" method explained in step 7.

Step 7: Lastly, these folders and sub-folders serves as input to indexing function. In indexing a separate html file is creates named "Email Index", which contain the each folder name and respective link to that folder, and folder summarization to identify keywords, is done applying Apriori algorithm (if, minimum support = 2).

a. From folder "Assignment" again feature terms are extracted according to step 2. For email,

   E1 – {Assignment, Student, Attached, Try}

   E2 – {Assignment, Student, Attached, University}

   E3 – {Assignment, Attached}

b. Apply Apriori algorithm on these emails:

   i. The first counts the number of occurrences (support), of each item separately. We get:

Table 3.4: Example output of 1st iteration of Apriori

Item	Support
Assignment	3
Student	2
Attached	3
Try	1
University	1

ii. The next step is to generate a list of all pairs of the frequent items:

Table 3.5: Example output of 2nd iteration of Apriori

Item	Support
{Assignment, Student}	2
{Assignment, Attached}	3
{ Student, Attached}	2

iii. Similarly, we perform the 3rd iteration:

Table 3.6: Example output of 3rd iteration of Apriori

Item	Support
{Assignment, Student, Attached}	2

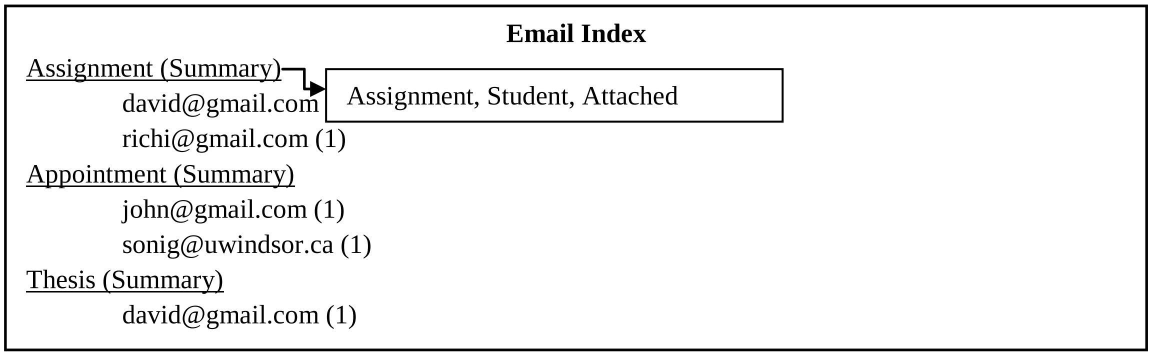
iv. In this way, we can *prune* sets and finally we get:

Table 3.7: Example output of 4th iteration of Apriori

Item	Support
{Assignment, Student, Attached}	2

c. These frequent patterns are used as keywords for folders identification as summarization.

In our considered example output for whole AEMS model as shown in Figure 3.9.



Email Index

Assignment (Summary)
david@gmail.com
richi@gmail.com (1)
Appointment (Summary)
john@gmail.com (1)
sonig@uwindsor.ca (1)
Thesis (Summary)
david@gmail.com (1)

Assignment, Student, Attached

Figure 3.10: Email Index File

CHAPTER 4

EXPERIMENTS AND PERFORMANCE STUDY

This section discusses the implementation and experiments performed to evaluate our proposed system in terms of effectiveness in finding email with vague idea of topic and sender, correctness for cluster and also did CPU execution time analysis.

4.1 Dataset

To test the correctness and effectiveness of our proposed method of email mining, the publically available 20-Newsgroup (20NG) (Lang, 1995) datasets is used. The 20-Newsgroup datasets available online on the website http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html/. It is a collection of 19,997 email messages taken from Usernet newsgroup collections, which then partitioned equally across 20 different newsgroups as shown in Table 4.1. Each message is assigned into one or more 20 semantic categories. The dataset contains the following 6 main classes (computer, science, recreation, political talks, atheism and social) with their subclasses. Some data in dataset are closely related and some are highly unrelated. It is mainly used for the purpose of text classification and text clustering evaluation.

Table 4.1: Usenet newsgroups used in newsgroup data

File Name	**Number of Documents**
alt.atheism	1000
comp.graphics	1000
comp.os.ms-windows.misc	1000
comp.sys.ibm.pc.hardware	1000
comp.sys.mac.hardware	1000
comp.windows.x	1000
misc.forsale	1000
rec.autos	1000
rec.motorcycles	1000
rec.sport.baseball	1000
rec.sport.hockey	1000
sci.crypt	1000
sci.electronics	1000
sci.med	1000
sci.space	1000
soc.religion.christian	997
talk.politics.guns	1000
talk.politics.mideast	1000
talk.politics.misc	1000
talk.religion.misc	1000

In 20 Newsgroups dataset, data comes in a format of one file per email message where, directory name shows its respective class. Each file contains email logs information as in Table 4.2.

Table 4.2: Each file content 20 Newsgroup dataset

Field Name	Description	Example
From	It contains the email address of the person who sent the message	bed@intacc.uucp
Date	It is the date when message originally posts to the network	Thu, 18 Mar 1993 00:21:49 GMT
Newsgroup	It specifies the newsgroup or newsgroups in which the message belongs	sci.med
Subject	It briefly tells what the message is about	INFO NEEDED: Gaucher's Disease
Message-ID	It gives the message a unique identifier	1993Mar18.002149.1111@intacc.uucp
Path	It shows the path the message took to reach the current system	cantaloupe.srv.cs.cmu.edu!crabapple.srv
Lines	It contains a count of the number of lines in the body of the message	33
Content	It is the body of the message	I have a 42 yr old male friend.....

4.2 Algorithm Implementation

The whole AEMS module is implemented in Java, and Eclipse is used as a development IDE (Integrated Development Environment), where the programs are compiled with AEMS_system.java as filename. For command prompt the programs are compiled with "javac AEMS_system.java" and executed with "java AEMS_system". Using method Data_preparation of class AEG_model, the subject and content of each email that is present in binary file format in folder 20_newsgroup is extracted. This data is then used in the whole process of the AEMS module.

4.3 Evaluation Criteria

The effectiveness and correctness of the proposed technique is calculated by using the two criteria which are precision rate and recall rate. Precision is measured as average in percentage for the number of correct data retrieved divided by the total number of data retrieved by the system. Recall is measured as average in percentage for the total number of correct data retrieved divided by the total number of existing data in the web document. The mathematical formulas for precision rate and recall rate are given in Equation 4.1 and 4.2 respectively:

$$Precision, P(i,j) = n_{ij}/n_j \tag{4.1}$$

$$Recall, R(i,j) = n_{ij}/n_i \tag{4.2}$$

n_i: The number of the cluster *i* which human has labeled;

n_j: The number of the emails *j* with clustering algorithms;

n_{ij}: The number of the emails clustered correctly.

Also the F-value is calculated to evaluate the clustering quality. F-value is a function that combines precision and recall according to the formula is defined in Equation 4.3:

$$F - Value, F(i,j) = 2 * P(i,j) * R(i,j)/P(i,j) + R(i,j) \tag{4.3}$$

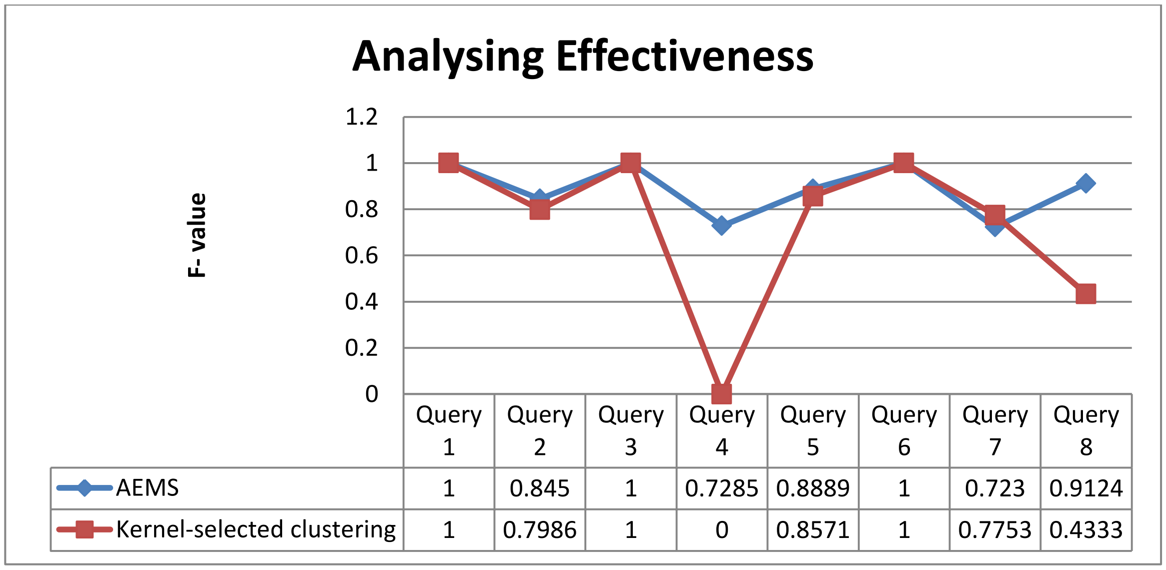
4.4 Studies on Effectiveness

The purpose of this experiment is to measure the effectiveness of AEMS module for finding email with vague idea of topic and sender. Therefore, we have run a simulation with a set of 8 queries, where these queries indicate the user purposes for searching a particular email. Figure 4.1 gives the list of sample queries. Based on these queries we have compared the effectiveness in terms of F-value and run time. Here, F-value indicates the correctness of results, which shows the effectiveness of folder summarization, by extracting emails using both AEMS module and Kernel-selected clustering. We have also observed the run time for finding the desired email using both AEMS module and Kernel-selected clustering.

***Query 1**: Find all email from sender with email ID as "mathew@mantis.co.uk", "cavalier@blkbox.com" or "amjad@eng.umd.edu" and topic as "politics", "religion" or "atheism".*
***Query 2**: Find all email from sender with email ID as "mathew@mantis.co.uk" or "cavalier@blkbox.com" and topic as "politics" and "religion" and "video".*
***Query 3**: Find all email on topic "graphics".*
***Query 4**: Find all emails from sender with email ID as "zyeh@caspian.usc.edu" and topic as "video".*
***Query 5**: Find all email from sender with email ID as "mathew@mantis.co.uk" and "cavalier@blkbox.com" and topic as "graphics".*
***Query 6**: Find all email from sender with email ID as "mathew@mantis.co.uk", "cavalier@blkbox.com" and "amjad@eng.umd.edu".*
***Query 7**: Find all email from sender with email ID as "mathew@mantis.co.uk" and topic as "code" or "graphics".*
***Query 8**: Find all email from sender with email ID as "mathew@mantis.co.uk" and topic as "god", "doctor" and "atheism".*

Figure 4.1: List of sample queries

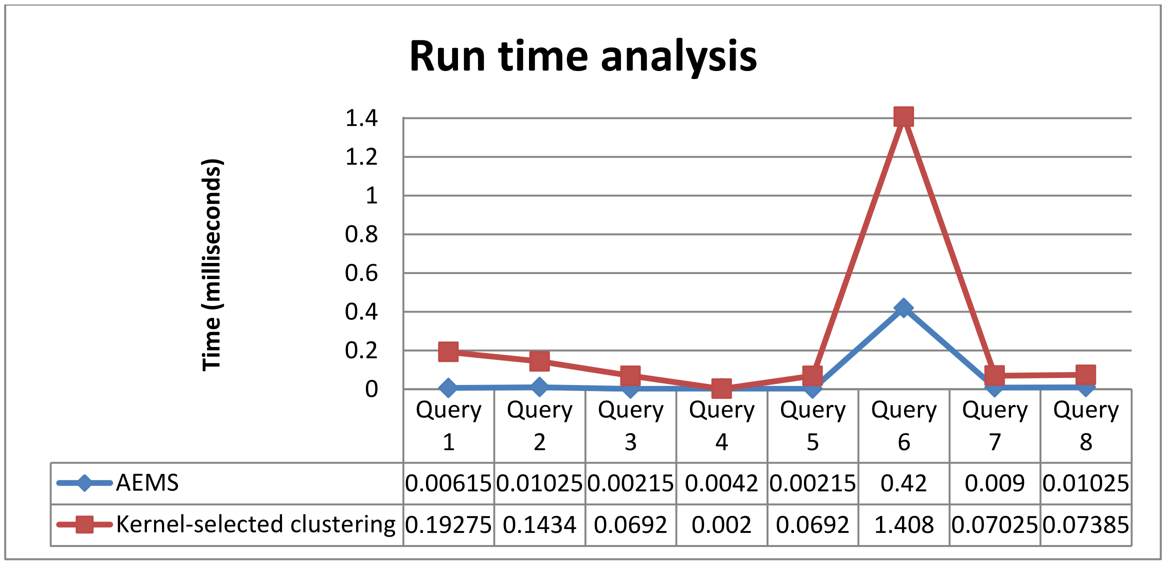
The results shown in Figure 4.2, are showing the correctness of results when applying the queries and Figure 4.3 shows the runtime analysis to execute each query by comparing both AEMS module and kernel-selected clustering.



	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8
AEMS	1	0.845	1	0.7285	0.8889	1	0.723	0.9124
Kernel-selected clustering	1	0.7986	1	0	0.8571	1	0.7753	0.4333

Figure 4.2: Analyzing effectiveness

Here in Figure 4.2, we can observe that the efficiency of kernel-selected sometimes decreases such as in *query 4* the F-value for kernel-selected is coming zero, this is because the topic given in the query was not available in the topic list of kernel-selected method. Therefore, it did not extract any email, whereas for our AEMS module as keyword was present in the folder summary, so, we got the set of emails on that particular topic. Similarly, in *query 8* the F-value for kernel-selected is coming much less than the AEMS module because topic *"atheism"* belongs to the main folder name but topic *"god"* and *"doctor"* are present in the summary only. Hence, kernel-selected cannot retrieve the topics *"god"* and *"doctor"* but AEMS can do and provide efficient results.



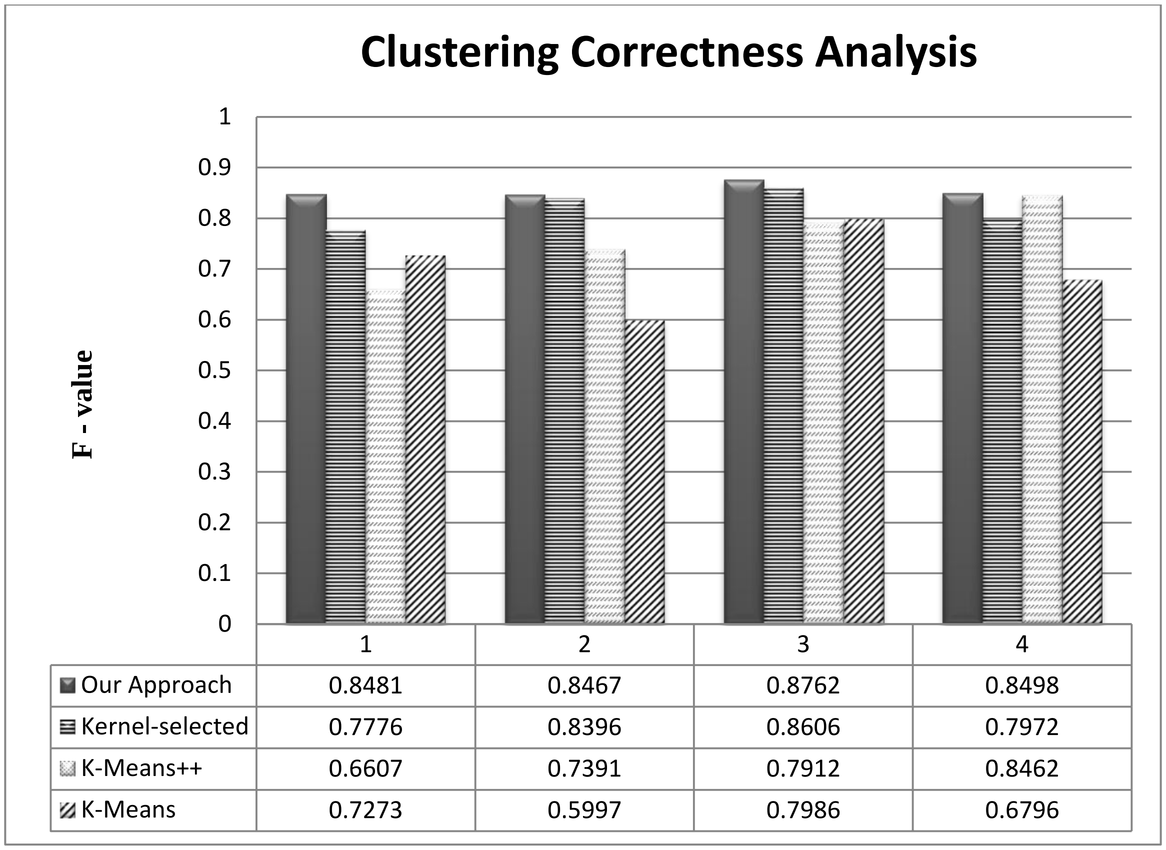
	Query 1	Query 2	Query 3	Query 4	Query 5	Query 6	Query 7	Query 8
AEMS	0.00615	0.01025	0.00215	0.0042	0.00215	0.42	0.009	0.01025
Kernel-selected clustering	0.19275	0.1434	0.0692	0.002	0.0692	1.408	0.07025	0.07385

Figure 4.3: Runtime analysis

Here in Figure 4.3, we can observe that the run time for AEMS module is less than kernel-selected clustering without making change in the F-value because in kernel-selected clustering for finding email, we need to go to each email and then extract the sender email ID whereas for our AEMS module we need to match the subfolder name and we get the desired email.

4.5 Studies on Cluster Correctness

The evaluation process we compared of our Semantic Non-parametric K-Means++ clustering with the standard K-Means algorithm (Surendran et al., 2005), standard K-Means++ algorithm (Arthur and Vassilvitskii, 2007) and Kernel-selected algorithm (Yang et al., 2010) with respect to cluster correctness. We choose four folders of the 20 NewsGroup dataset. These four folders consist of 1000 documents each and results are evaluated in terms of F-Value. Now, when experimenting data with Kernel-selected clustering method and our proposed Semantic Non-parametric K-Means++ clustering the $\beta = 0.5$ is taken, because when $\beta = 0.1$, get the one cluster centers or K; when $\beta = 0.2$, get the two cluster centers or K, when $\beta = 0.5$, the four email is selected as cluster center. Additionally, for cluster performance threshold T_s and T_b is set to zero (i.e., taking $T_s = T_b = 0$), therefore, algorithm will consider all terms present in each email. When clustering emails with the standard K-Means and K-Means++ clustering algorithm to form the email vector, the standard TF-IDF is used to form the email vector represented using standard vector space model, by combining the subject and content of email. In K-Means clustering all kernels selected randomly, and in Kernel-selected algorithm and in K-Means++ clustering initial kernels selected randomly therefore, the results are different with the test times, therefore, we choose a best result which is shown in Figure 4.3.



	1	2	3	4
■ Our Approach	0.8481	0.8467	0.8762	0.8498
▤ Kernel-selected	0.7776	0.8396	0.8606	0.7972
▒ K-Means++	0.6607	0.7391	0.7912	0.8462
▨ K-Means	0.7273	0.5997	0.7986	0.6796

Figure 4.3: F-value comparison of clustering algorithms

Graphical representation shown in Figure 4.4, is the average F-Value of four clusters for the standard K-Means, standard K-Mean++, Kernel-selected algorithm and semantic K-Means++ clustering. Here, graph shows that our approach performs better than the standard K-Means, standard K-Means++ and Kernel-selected algorithm. Since the average of the F-Value when threshold is taken as zero comes out to be 0.8552 for Semantic Non-parametric K-Means++ clustering algorithm whereas for Kernel-selected clustering method comes to be 0.8187

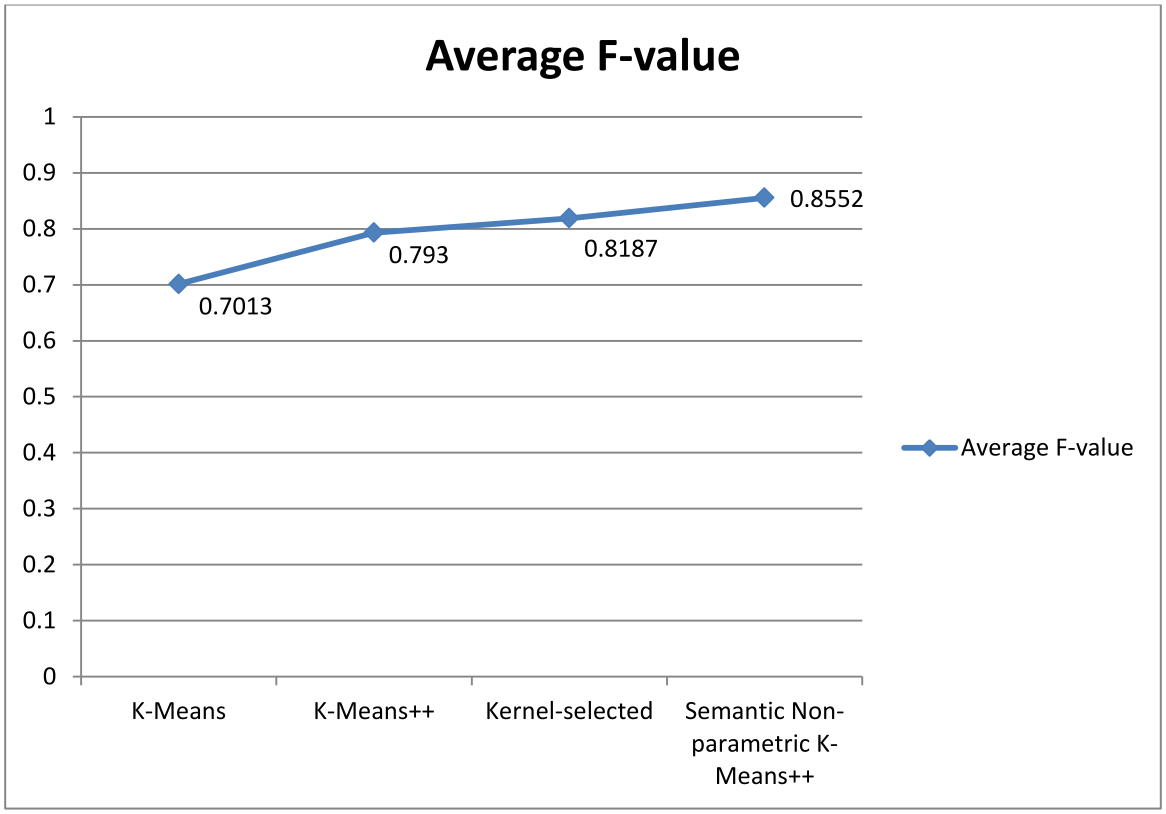


Figure 4.4: Comparison of average of F-Values of algorithms

4.4 Runtime Analysis

The runtime analysis is done to check the advantage of using feature selection for email clustering explained in section 3.3.2. The features are selected based on the different values of threshold T_s for subject and T_b for body of the email and calculated the clustering performance in terms of average F-value for four folders created, where the total number of emails is 100. Table 4.3 shows the results for feature selection and effect of it on clustering performance.

Table 4.3: Runtime evaluation using feature selection

Threshold	**Number of feature terms**	**Execution time (Seconds)**	**Average F-Value**
T_s = 0% T_b = 0%	5094	164768	0.8116
T_s = 0% T_b= 5%	406	12771	0.7981
T_s = 0% T_b= 10%	280	9460	0.8061
T_s = 5% T_b= 5%	206	6240	0.7868
T_s = 5% T_b= 10%	50	1680	0.6994

From Table 4.3 it can be observed that careful selection of subject and body threshold can significantly reduce the number of feature terms which apparently reduces the execution time of the program without compromising the accuracy in terms of F-value significantly.

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

Email mining for managing incoming email messages is a challenge for text mining community. The main goal of this research is to handle email overload by systematically arranging email messages for further reference so that the user is also able to find a particular email when there is vague idea of sender and topic of email. To handle this an algorithm AEMS (Automatic Email Management System) which manages emails by organizing similar emails in the folders (module 1 named AEG), then again organizes emails of emails of each folder into subfolders (module 2 named APEG) where subfolder will contain emails sent by only a particular person and lastly creating the index, containing name and link to the folders and sub-folders, and also a summary annotation about the content of the respective folders. The model AEG (Automatic Email Grouping), document frequency based feature selection method named associative term frequency has been introduced. Also a novel Semantic Non-parametric K-Means++ clustering method for folder creation is proposed, which overcomes the problem of random seed selection and pre-defined cardinality of clustering as in K-Means and K-Means++ clustering algorithms. Our method selects the seed according to the email weight and decides the cardinality according to the similarity between the email content; along with adding semantics to the clusters. Additionally, index/view is created which contains the link to each folder and sub-folder and Apriori-based folder summarization which contains important keywords from the folder. The approaches proposed successfully can provide better quality clustering with less computational time as well as provide help when user needs to find particular email with its vague information.

The work of this thesis can be extended by improving the feature selection method which can be done using frequent pattern mining. By applying frequent pattern mining on the email dataset, we can get important terms from the whole dataset. There are two advantages of using frequent pattern mining, (1) we get the feature terms that can best represent each email and can reduce computational time because after feature selection we will only deal important terms rather than all terms in whole dataset, and (2) these feature terms can be directly used for the folder summary as those terms can represent the content of the folder. The AEMS module does not handle the processing of incoming messages; therefore a method can be developed to immediately process incoming messages using classification methods. Also, some recommendation system can be built based on the emails logs for deletion of unused email or cluster. Additionally, a method can be developed for spell check, when user is finding an email with misspelled topic or sender name.

Bibliography

Androutsopoulos, I., Koutsias, J., Chandrinos, K. V., and Spyropoulos, C. D. (2000). An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '00, pages 160–167, New York, NY, USA. ACM.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Ayodele, T., Zhou, S., and Khusainov, R. (2010). Email grouping method. In Ao, S., Gelman, L., Hukins, D., Hunter, A., and Korsunsky, A., editors, *The World Congress on Engineering 2010 Volume I*, number 2183 in Lecture Notes in Engineering and Computer Science, pages 315–320. Newswood Limited, London, UK.

Balog, K. and de Rijke, M. (2006). Finding experts and their eetails in e-mail corpora. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 1035–1036, New York, NY, USA. ACM.

Balog, K., Thomas, P., Craswell, N., Soboroff, I., Bailey, P., and De Vries, A. (2008). Overview of the trec 2008 enterprise track. Technical report, DTIC Document.

Bogawar, P. S. and Bhoyar, K. K. (2012). Email mining: a review. *IJCSI International Journal of Computer Science Issues*, 9(1).

Carenini, G., Ng, R. T., and Zhou, X. (2007). Summarizing email conversations with clue words. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 91–100, New York, NY, USA. ACM.

Cormack, G. V. and Lynam, T. R. (2007). Online supervised spam filter evaluation. *ACM Trans. Inf. Syst.*, 25(3).

Crawford, E., Kay, J., and McCreath, E. (2002). Iems - the intelligent email sorter. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 83–90, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Cselle, G., Albrecht, K., and Wattenhofer, R. (2007). Buzztrack: topic detection and tracking in email. In *Proceedings of the 12th international conference on Intelligent user interfaces*, IUI '07, pages 190–197, New York, NY, USA. ACM.

Diao, Y., Lu, H., and Wu, D. (2000). A comparative study of classification based personal e-mail filtering. In Terano, T., Liu, H., and Chen, A., editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, volume 1805 of *Lecture Notes in Computer Science*, pages 408–419. Springer Berlin Heidelberg.

Entlich, R. (2006). Youve got mailnow what? regulatory and policy dilemmas in email management part ii. us state environment. *RLG DigiNews*, 10(3).

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (1996). *Advances in Knowledge Discovery and Data Mining.* The MIT Press.

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, pages 309–312, New York, NY, USA. ACM.

Guan, R., Shi, X., Marchese, M., Yang, C., and Liang, Y. (2011). Text clustering with seeds affinity propagation. *IEEE Trans. on Knowl. and Data Eng.*, 23(4):627–637.

Han, J. and Kamber, M. (2000). *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, 1st edition.

Heidorn, G. (2000). Intelligent writing assistance. *Handbook of Natural Language Processing*, pages 181–207.

Islam, A. and Inkpen, D. (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data*, 2(2):10:1–10:25.

Katakis, I., Tsoumakas, G., and Vlahavas, I. (2006). *Email Mining: Emerging Techniques for Email Management*, pages 219–240. Idea Group Publishing.

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

Kulkarni, A. and Pedersen, T. (2008). Name discrimination and e-mail clustering using unsupervised clustering of similar contexts. *Journal of Intelligent Systems*, 17(1-3):37–50.

Kushmerick, N. and Lau, T. (2005). Automated email activity management: an unsupervised learning approach. In *Proceedings of the 10th international conference on Intelligent user interfaces*, IUI '05, pages 67–74, New York, NY, USA. ACM.

Lang, K. (1995). NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA.

Li, H., Shen, D., Zhang, B., Chen, Z., and Yang, Q. (2006a). Adding semantics to email clustering. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 938–942, Washington, DC, USA. IEEE Computer Society.

Li, Y., McLean, D., Bandar, Z., O'Shea, J., and Crockett, K. (2006b). Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on*, 18(8):1138 –1150.

Lloyd, S. (2006). Least squares quantization in pcm. *IEEE Trans. Inf. Theor.*, 28(2):129–137.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55.

Manco, G., Masciari, E., and Tagarelli, A. (2008). Mining categories for emails via clustering and pattern discovery. *Journal of Intelligent Information Systems*, 30:153–181.

Markov, Z. and Larose, D. T. (2007). *Data mining the web - uncovering patterns in web content, structure, and usage.* Wiley.

Meyer, T. A. and Whateley, B. (2004). Spambayes: Effective open-source, bayesian based, email classification system. In *Proceedings of the First Conference on Email and Anti-Spam (CEAS)*, volume 98.

Muresan, S., Tzoukermann, E., and Klavans, J. L. (2001). Combining linguistic and machine learning techniques for email summarization. In *Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, pages 19:1–19:8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nagwani, N. and Bhansali, A. (2010). An email clustering model using weighted similarities between emails attributes. *International Journal of Research and Reviews in Computer Science (IJRRCS)*, 1(2).

Oliver, C. S., Ringger, E., Gamon, M., and Campbell, R. (2004). Task-focused summarization of email. *Proceedings of the Text Summarization Branches Out ACL Workshop.*

Pazzani, M. J. (2000). Representation of electronic mail filtering profiles: a user study. In *Proceedings of the 5th international conference on Intelligent user interfaces*, IUI '00, pages 202–206, New York, NY, USA. ACM.

Porter, M. F. (1997). Readings in information retrieval. chapter An algorithm for suffix stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, 1:81–106.

Radicati, S. and Hoang, Q. (2011). Email statistics report, 2011-2015. *Retrieved May*, 25:2011.

Rambow, O., Shrestha, L., Chen, J., and Lauridsen, C. (2004). Summarizing email threads. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 105–108, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998). A Bayesian Approach to Filtering Junk E-Mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin. AAAI Technical Report WS-98-05.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing Management*, 24(5):513 – 523.

Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Schuff, D., Turetken, O., and D'Arcy, J. (2006). A multi-attribute, multi-weight clustering approach to managing e-mail overload. *Decision Support Systems*, 42(3):1350 – 1365.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Surendran, A. C., Platt, J. C., and Renshaw, E. (2005). Automatic discovery of personal topics to organize email. In *Proc. of the 2nd Conference on Email and Anti-Spam, Stanford University*.

Tedmori, S., Jackson, T. W., and Bouchlagem, D. (2007). Optimising the email knowledge extraction system to support knowledge work. In *ECIS*, pages 681–691.

Tkach, D. S. (1998). Information mining with the ibm intelligent miner family. *An IBM Software Solutions White Paper*, pages 1–29.

Ulrich, J., Carenini, G., Murray, G., and Ng, R. T. (2009). Regression-based summarization of email conversations. In Adar, E., Hurst, M., Finin, T., Glance, N. S., Nicolov, N., and Tseng, B. L., editors, *ICWSM*. The AAAI Press.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Whissell, J. S. and Clarke, C. L. A. (2011). Clustering for semi-supervised spam filtering. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, CEAS '11, pages 125–134, New York, NY, USA. ACM.

Whittaker, S., Bellotti, V., and Gwizdka, J. (2006). Email in personal information management. *Commun. ACM*, 49(1):68–73.

Whittaker, S., Jones, Q., Nardi, B., Creech, M., Terveen, L., Isaacs, E., and Hainsworth, J. (2004). Contactmap: Organizing communication in a social desktop. *ACM Trans. Comput.-Hum. Interact.*, 11(4):445–471.

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, pages 276–283, New York, NY, USA. ACM.

Xiang, Y., Zhou, W., and Chen, J. (2009). Managing email overload with an automatic nonparametric clustering approach. In Li, K., Jesshope, C., Jin, H., and Gaudiot, J.-L., editors, *Network and Parallel Computing*, volume 4672 of *Lecture Notes in Computer Science*, pages 81–90. Springer Berlin Heidelberg.

Yang, H., Luo, J., Yin, M., and Liu, Y. (2010). Automatically detecting personal topics by clustering emails. In *Education Technology and Computer Science (ETCS), 2010 Second International Workshop on*, volume 3, pages 91 –94.

Zeng, C., Lu, Z., and Gu, J. (2008). A new approach to email classification using concept vector space model. In *Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia - Volume 03*, FGCNS '08, pages 162–166, Washington, DC, USA. IEEE Computer Society.

VITA AUCTORIS

NAME:	Gunjan Soni
PLACE OF BIRTH:	Sagar, (M.P.), India
YEAR OF BIRTH:	1986
EDUCATION:	University of Windsor, M.Sc., Windsor, ON, 2013
	University of Pune, M.C.A, Pune, India, 2010
	Dr. Hari Singh Gour University, B.C.A., Sagar, India, 2007