Identifying Word Categories for Diffusion Studies

in an Email Social Network

by

Timothy Choe


Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

August 2006

©2006 Massachusetts Institute of Technology
All rights reserved.
Identifying Word Categories for Diffusion Studies
in an Email Social Network

by

Timothy Choe

Submitted to the
Department of Electrical Engineering and Computer Science

August 22, 2006

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

To better understand the sources of information workers’ productivity in relation to email social network characteristics, we identify four distinct word categories in a corporate email corpus based on diffusive and informative characteristics. The four categories are “common – everyday”, “common – specialized”, “event”, and “clique” words. Common – everyday words exhibit no diffusive or informative characteristics and are used regularly in everyday communication. Common – specialized words also exhibit no diffusive characteristics, but are hypothesized to be specialized vocabulary used infrequently in everyday communication. Event words are diffusive pieces of information that reach a majority of the population rapidly and are likely triggered by events. Clique words are diffusive pieces of information that reach only a small portion of the population and exhibit characteristics of exclusive, specialized knowledge shared among groups. These four categories can help future studies assess whether social networks determine information diffusion and whether information diffusion is associated with productivity.

Thesis Supervisor: Erik Brynjolfsson
Title: George & Sandi Schussel Professor of Management

Thesis Co-supervisor: Sinan Aral
Title: PhD Candidate, Sloan School of Management
List of Figures and Tables

Figures

Figure 1. A figure from *Information Diffusion through Blogspace* 11
Figure 2. Initial Filters for deriving *Candidate* Words 16
Figure 3. Frequency Distribution of All Words 18
Figure 4. An Example One Cycle Word 19
Figure 5. Ideal One Cycle Word 20
Figure 6. People distribution for *Candidate* Set of Words 22
Figure 7. People distribution for *Words in Every Week* Set of Words 22
Figure 8. Event word 23
Figure 9. Potential Event Words? 23
Figure 10. Random Word Diffusion Patterns 25
Figure 11. Tree Graph Type I 26
Figure 12. Tree Graph Type II 27
Figure 13. Clique Words 29
Figure 14. Activity and Diffusion Patterns for common – everyday words 31
Figure A-1. Design Architecture 39
Figure A-2. *MyEmailDBHelp Design* 41
Figure A-3. Cycles Algorithm Flow Chart 46
Figure A-4. Filters to *Candidate* Words 47
Figure A-5. Top-Level Algorithm for Metrics Generation and Selection 49
Figure A-6. Activity Hasetable 53
Figure A-7. Histogram of nofroms/numemails 54
Figure A-8. Node Data Structure 55
Figure A-9. Tree Graph Data Structure Creation 56
Figure A-10. Bad Scenario for Recursion 58
Figure A-11. Infection based Algorithm 59
Figure A-12. Type I and II Tree Graph Difference 61
Figure A-13. Difference between previous SOT matrix and a correct one 64
Figure A-14. Shortest path example 66
Figure B-1. Histogram of Frequencies for each word categories 68
Figure B-2. Histogram of People for each word categories 69

Tables

Table 1. Summary of Knowledge Types 12
Table 2. Word Categories 32
Table 3. Metric Definitions 33
Table 4. Summary Statistics 33
Table 5. Word Categories, Knowledge Types, and Diffusion Models Summary 37
Table 6. Sample Data Set 38
Table A-1. Sample Rows in Email Data 40
Table A-2. Word/Frequency Pair 43
Table A-3. The metrics for word categorization 48
Table A-4. Tree Graph Metrics 60
Table A-5. Example Social Network Matrix 65
Table B-1. T-tests comparison results 71
1 Introduction

Information workers or knowledge workers, a phrase coined by Peter Drucker in his book, *Landmarks of Tomorrow*, are workers who primarily use and/or develop information for their output. They are already a large part of the economic landscape in the world and for advanced industrialized countries like the US, they are the overwhelming majority of the workforce. Some claim that unlike manufacturing production workers, whose output and productivity can be measured in terms of the costs and revenue associated with the products they generate, information workers' tangible output and productivity are hard to define [1]. Although using more information or generating more information may provide some degree of correlation between information workers and their performance, the relationship does not hold true all the time and is dependent on many other factors. Aral, Brynjolfsson, and Van Alstyne (2006) set out to explore this relationship by studying the task level practices of information workers at a midsize executive recruiting firm [2]. Other factors besides information that are analyzed in this study include information technology (IT) use and skills, the structure of information flows, and the hierarchical structure of the company. One part of this overall project is the analysis of emails in the company. Email data provide a clearly definable social network structure, information content, and a measure of IT use. Aral, Brynjolfsson, and Van Alstyne (2006) have already found that email social network structure is positively correlated with productivity and performance. This thesis will try to explore a second order effect in the email analysis and look at the actual content of the email and the diffusive characteristics of words or pieces of information flowing through the email social network. One critical problem in studying the diffusion of information through a social network involves separating words or language used by individuals as a matter of their own thoughts or ideas from words or language that individuals become aware of as a result of receiving that information from others in their social network through information flows in email. Because it is hard to verify if information is diffusing through the email network rather than users generating the information themselves individually, this research is focused on identifying sets or categories of words that exhibit diffusive characteristics and are pertinent to the social network structure of the firm. Previous studies have been done on social network analysis and productivity [3], but this thesis will try to examine in detail methods and measures for the identification and categorization of strategic information that is likely to be diffused through email communication. The end goal is to provide clearly definable and distinctive categories of words in email data sets that may provide insight into the social network characteristics of the firm and information worker productivity.

2 Theory/Motivation

2.1 Models of Diffusion in Social Networks
There are three main types of models for diffusion in social networks: diffusion of innovations, disease propagation, and game theory-based models. These models describe how the property being diffused (i.e. innovations, diseases, information, etc.) flows from one person to the next in a social network. Although each has its own parameters and assumptions regarding the behavior of people passing and receiving the diffused property, they all are based on social network theory. Therefore, understanding these models provides insight into the relationship between social networks and diffusion and how to determine which categories or sets of words in email would be useful in social network diffusion analyses.

2.1.1 Diffusion of Innovations

The network model of diffusion of innovation looks at how innovations are spread and adopted among groups of people and organizations by analyzing the relationships between the users, the structural position of users in relation to the whole network, and individual and system-wide critical mass thresholds necessary for innovations to diffuse [4]. An early and influential 1943 study [5] on the adoption of hybrid corn among two Iowa communities showed that innovation diffusion is a social process and dependent on individuals’ social types and characteristics. Although the study did not contain any network analysis, it prompted other studies to look at the social network aspects of innovation adoption rather than other factors, such as economic or profit-driven risk analysis. Some landmark papers and ideas in social network modeling for innovation diffusion are Roger’s (1970) [6] opinion leadership, Granovetter’s (1973) [7] strength of weak ties, and Burt’s structural equivalence (1987) [8] and structural holes (1992) [3]. Social network modeling helps describe the diffusion of innovation through the lens of the whole network of people and on an individual basis as well.

For the whole network level, some important characteristics of the social network for diffusion are the strength of weak ties, network centrality, the extent of centralized innovators, and the critical mass needed to cause continuous diffusion in the network. The strength of weak ties (SWT) theory highlights the importance of the degree to which unrelated subgroups of a social network are tied or connected to each other. SWT is probably one of the most important characteristics for innovations to diffuse across entire social networks [9]. Network centrality is the extent to which connections between individuals in the network are centered around certain members [10]. Centralized networks lead to faster rates of diffusion because once the central members adopt the innovation, the spread to other members in the network proceeds fairly quickly. Also, if a social network has early adopting central members, the rate of diffusion is hypothesized to be even faster. Critical mass is the minimum number of members in the social network needed to sustain the diffusion process. Lower minimum numbers mean faster rates of innovation diffusion. In the current popular press, the idea of the “tipping point” [11] is used to describe this phenomenon.
On the individual level (or ego-centric level), social network models of innovation diffusion characterize each member’s relationships to other members, their structural position within the network, and predict the degree to which innovations are adopted along these characteristics. Relationships between users can be described by measures of opinion leadership, personal network density and radicality. Opinion leaders are the users who receive communication from the most people. Personal network density is the degree an individual’s personal network is interconnected. A dense network has individuals who communicate a lot to each other in many permutations, but the individuals speak primarily to one another and not to a large number of unconnected others. A radial personal network has a non-dense structure and the contacts within that network do not talk to each other but spread out to other parts of the overall network. Members of a social network who are opinion leaders and have a radial personal network tend to be faster adopters and drivers of innovation.

The structural positions of users in relation to the whole network can be described by a variety of measures. Two major classes are centrality measures, such as closeness and betweenness, and structural and positional equivalence. Closeness centrality describes how close one individual is to others in the network in terms of how many people it takes to reach a particular user. Betweenness centrality describes the degree to which an individual is in between others in the network. It measures the probability that a user will be on the shortest path between any two other individuals in the network [4]. Both centrality measures have been shown to be positively correlated with an individual’s rate of adoption in some settings. Structural and positional equivalence is the degree to which two individuals have the same relations with others. Diffusion of innovation theory postulates that members with structural and positional equivalence adopt innovations at similar times and rates. In the context of email diffusion studies, central members are likely to see more strategic information, while structurally equivalent members may see similar information types and see them at relatively the same time.

On both the whole network and individual levels, the model of social network is based on connections or ties of communication between members of the social network. Another approach is to create social networks based on models of individual behavior in relation to innovation diffusion, such as independent cascade models, which look at the diffusion of innovation through the lens of people who influence other members in the network to adopt. For each discrete time step, all current members who have already adopted the innovation have certain probabilities to spread the adoption to immediate members and only have one chance to do so [12]. Independent cascade models have been used to try to find the right set of initial adopters to maximize the cascade of adoption and have been extended to other variations of the model [13].

2.1.2 Epidemic Models
Epidemic models describe how a disease propagates through social networks. In the context of emails, words or information can be substituted for the item being propagated. The most widely-known and standard class of models is the SIR model. The SIR model analyzes disease propagation through the lens of the disease cycle in individuals in the network. First, a person becomes (S)usceptible to a disease, which then leads to (I)nfection, followed by (R)ecovery. At the infected stage, individuals can in turn cause susceptible people to become infected. A recovered person cannot become infected again. SIR models are built around governing equations or processes that relate the three types of people [14]. The simplest and earliest of these models is the Kermack-McKendrick model [15]. The governing equations, which are coupled nonlinear ordinary differential equations, are:

\[ \begin{align*}
\frac{dS}{dt} &= -\beta SI, \\
\frac{dI}{dt} &= \beta SI - \gamma I, \\
\frac{dR}{dt} &= \gamma I
\end{align*} \]

where \( t \) is time, \( S(t) \) is the number of susceptible people, \( I(t) \) is the number of people infected, \( R(t) \) is the number of people who have recovered and developed immunity to the infection, \( \beta \) is the infection rate, and \( \gamma \) is the recovery rate. The Kermack-McKendrick model assumes that the population size is fixed, has no social structure, and that a disease is initially present. Another model, a mixed information source model, incorporates an additional common source outside the network to initially spread the disease to the network [16]. This enhancement to the Kermack-McKendrick SIR model makes a plausible assumption of how diseases start. One other model [17] replaces the homogenous social ties by varying the number of connections and the disease transmission probabilities.

2.1.3 Game Theory Models

Game theory models for diffusion in social networks treat members of the network as agents who act rationally and make decisions to maximize a utility function based on the actions of other members. This can be achieved through competition, cooperation, or a combination of both with fellow network agents. Two common types of games that model diffusion are adoption games and link formation games. Agents in adoption games have a certain threshold in which they will eventually adopt something, such as new technology or information. Adoption games can have agents with different levels of utility associated with adoption. The value of the utility can also change depending on other agents actions and decisions as well as the utility associated with adoption [18]. The word-of-mouth spread of an innovation can be described using this model [19].

Link formation games try to model information flows rather than adoption. Agents in these games form ties with other agents in a network, but link formation costs the initiating agent some negative utility
value. Forming ties with other agents provides benefits to both parties but to varying degrees depending on the value of the information the other party has [20]. An agent's decision making process in forming new links can be modeled from simple cost-benefit analysis to complex, adaptive strategies [21].

2.2 Information/Knowledge Types

In addition to models of diffusion, understanding how and why information is categorized into different types is important in determining sets of words in an email corpus that are diffusive and may correlate with social network characteristics and individual productivity. Information types are different than specific knowledge categories, such as biology, computers, or finance. They try to divide an information space into sets that are subject-agnostic but distinctive on other dimensions. There are two main classes of information type sets: non-social and social. The non-social class consists of information type sets that come from a variety of fields, such as linguistics [22], philosophy [23], cognitive psychology [24], or Artificial Intelligence [25]. They are mainly determined by characteristics of the information itself and not by extra meta-data about who used it or how many people used it in a social setting. Of the non-social knowledge types, the most widely known categorization of knowledge is from cognitive psychology. Knowledge can be divided into “knowing-how” and “knowing-that” or what is known today as procedural and declarative knowledge. The social class of information types tries to characterize information based on usage, the usefulness of the information for specific groups of people and other social factors involved with the information. This class is prevalent in knowledge management theory and is described in more detail below.

2.2.1 Tacit - Explicit

In the context of knowledge management theory, knowledge in organizations can be divided into at least two dimensions: tacit and explicit knowledge [26]. Tacit knowledge is internalized, subconscious knowledge that includes both cognitive and technical expertise. A person may not be consciously aware of this kind of knowledge but may use it to perform a variety of tasks. Cognitive tacit knowledge represents an individual's beliefs or common assumptions in processes and relationships. It is a mental model of how things work. Technical tacit knowledge is specific know-how or skills applicable to work. An example is great batting skills. The basics of hitting a baseball can be explicitly communicated, but great batting skills require a personal, internalized knowledge that can be attributed to good instincts and years of experience. On the other hand, explicit knowledge refers to knowledge that is written, formalized, and/or codified in some way. This type of knowledge can be easily communicated to other members of the organization and can be stored in some permanent form. In the context of our research, because the knowledge is taken from an email corpus, which is a store of recorded, written communication, most
knowledge represented in the emails would be explicit. It could be possible that tacit knowledge is mentioned in the written communication without any explicit explanations, but because of the nature of the communication, most knowledge represented would be explicit.

The reason behind the tacit-explicit classification of knowledge in organizations is to motivate knowledge management systems to transform tacit knowledge into explicit knowledge and to encourage the internalization of explicit knowledge so that organizations can benefit from the totality of the knowledge base of their workers [27].

2.2.2 Stickiness

Information can also be categorized by its level of stickiness. Stickiness is defined as “the incremental expenditure required to transfer … [a] unit of information to a specified locus in a form usable by a given information seeker” [28]. The term expenditure in the definition can mean a variety of things, including actual financial costs to intangible human effort. In general, highly technical information is often very sticky, but the degree in which the information is explicit or tacit is also important. Typically, tacit information has high stickiness because of the extra expenditure needed to make it explicit. Other origins of stickiness in information include characteristics of the source, recipient, and context of the information being transferred [29]. The source or recipient of the information may lack motivation, may have problems with communication and learning, or may be situated in the wrong social, organizational context to understand the information. Information that requires greater and more specific levels of background context is considered to be stickier. In the context of email diffusion studies, the level of stickiness may correspond negatively to the degree of confidence in the diffusiveness of a word. If a word is part of a highly sticky cluster of information, then, by definition, it is unlikely to be transferred from one person to the next. Therefore, in the email data set, we would see the word in few peoples’ inboxes and we would conclude that the word in not diffusing.

2.2.3 Chatter and Spikiness

Another way to categorize information in email is through the levels of its activity across topics, which is described in Information Diffusion through Blogspace [30]. In the paper, Gruhl, et al. (2004) define a word to represent a topic when it exceeds certain TF(i) and TFCIDF(i) thresholds, and in many cases they are handpicked. The formulation of TFCIDF(i) is as follows:

\[
2) \quad TFCIDF(i) = \frac{(i - 1)TF(i)}{\sum_{j=1}^{i-1} TF(j)}
\]
TF(i) stands for the term frequency of a word in a period of time, where \( i \) refers to the current period and anything smaller than \( i \) refers to previous periods. The authors also define the terms ‘chatter’ and ‘spikiness’ to correspond to word usage activity levels associated in blogspace. Topics that are mostly ‘chatter’ have low variation of activity around their mean usage. Topics that are just ‘spikes’ have some form of inactivity followed by a burst of activity and then back to inactivity. Combining these two levels of activity is ‘spiky chatter,’ which is chatter plus some spikes. In the figure below, taken from the paper, “Alzheimer” is an example of a word characterized by chatter, “Chibi” is an example of a word that exhibits ‘just spikes,’ and “Microsoft” displays spiky chatter in its usage because it is constantly being talked about (chatter) and exhibits great changes in activity (spikiness) due to external events, such as new product offerings. The authors dig deeper into what constitutes spiky chatter and realize subtopics related to the main topic can account for the extra spikiness associated with spiky chatter. An example of a subtopic in the “Microsoft” example would be “longhorn” or “xp.”

![Figure 1: A figure from Information Diffusion through Blogspace: Activity levels for three example words](image)

Figure 1: A figure from Information Diffusion through Blogspace: Activity levels for three example words

### 3 Methods and Process

#### 3.1 Purpose and Goals

The purpose of this research is to gain deeper insight into the correlation between favorable social network structural positions and information worker productivity. Although analysis has been done that
establishes a correlation between productivity and characteristics of the social networks of employees in a mid-sized executive recruiting firm [2], the exact empirical mechanisms for the cause of the correlation still need to be explored. One hypothesis is that the diffusion of informative and strategic information through the email social network correlates positively with favorable structural positions and therefore, with productivity. For example, in our recruiting firm, if a recruiter is often privy to rare client information, such as candidate names or new job openings, then he may be more likely to be productive in recruiting transactions. Another example would be if a recruiter is first to hear about a new recruiting practice or new trends in an industry, then he may have a competitive advantage in gauging client demand, and hence be more productive. In both these examples, the larger hypothesis of this stream of research is that workers in favorable structural positions in the flow of information in the firm will be “first to know” and more often “in-the-loop” with regards to these types of informative and strategic pieces of information and will therefore be more productive.

<table>
<thead>
<tr>
<th>Knowledge Types</th>
<th>Definition</th>
<th>Characteristics</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedural and declarative knowledge from cognitive psychology</td>
<td>Procedural knowledge is “knowing-how” and declarative knowledge is “knowing-that.”</td>
<td>Declarative knowledge is about objects, events etc. and their relationships and states given. Procedural knowledge is control information, how to execute tasks on the objects in declarative knowledge.</td>
<td>Procedural – knowledge of how to brush one’s teeth&lt;br&gt;Declarative – Toothbrushes are usually made of plastic.</td>
</tr>
<tr>
<td>Tacit / Explicit</td>
<td>Tacit knowledge is internalized, subconscious knowledge. Explicit knowledge is formalized, written, and/or codified in some way.</td>
<td>A knowledge management system’s goal is to convert tacit knowledge into explicit knowledge and for explicit knowledge to be internalized.</td>
<td>Tacit – great baseball batting skills&lt;br&gt;Explicit – organizational chart of company</td>
</tr>
<tr>
<td>Stickiness</td>
<td>Stickiness refers to the degree of difficulty in transferring information from one person to another.</td>
<td>Stickiness can be caused by tangible and intangible sources such as transport costs or human intellect.</td>
<td>High Sticky – How to succeed in a consulting engagement. Low sticky – last night Red Sox’s score.</td>
</tr>
<tr>
<td>Spam</td>
<td>Spam are unsolicited emails that contain spurious ads and sent to a very large group of people.</td>
<td>There are a variety of spam block algorithms, the main ones are supervised machine learning techniques that requiring human attention.</td>
<td>Spam – those unwanted Viagra ads.</td>
</tr>
<tr>
<td>Chatter / Spikiness</td>
<td>Chatter is consistent level of activity using a word. Spikiness is jumps in usage from no or low levels of activity to high levels of activity.</td>
<td>Most interesting topics consist of chatter then some spikiness thrown in that is usually caused by a subtopic and external event.</td>
<td>Chatter – “Microsoft” subtopic spikes – “xp”, “longhorn”</td>
</tr>
</tbody>
</table>

Table 1 – Summary of Knowledge Types
To help validate this hypothesis, the goal of this thesis is to identify words that are highly diffusive and informative in the email data set of the recruiting firm. In the context of our research, a highly diffusive word means that we are more confident that the word is being passed from person to person. This is in contrast to the typical definition of high diffusiveness, which is analogous to being contagious. The reason why we want to identify highly diffusive words is so that we can be sure that tracking these words would result in following a diffusion process in the information flows of the firm. Without a diffusion process, the flow of words from person to person may mean nothing in terms of validating our hypothesis of the correlation between the diffusion of strategic information and favorable structural social network positions. One caveat in identifying highly diffusive words is that we must not choose the words based on the email links that connect them. This is because we will then be selecting words that mimic the email social network structure and finding correlations with favorable structural positions would be tautological.

Once we identify words that are highly diffusive, the second requirement is to identify words that are also informative. This is because not all highly diffusive words are informative or strategic. An example of this would be soccer highlights. People within the firm may pass that information along from person to person and it may provide indirect productivity benefits from employee bonding and social cohesion, but it probably has no direct value in recruiting efforts. To gauge the informative level of a word, we want the words to either show a) usage among a significant number of people or b) back-and-forth usage among the users of the word. The criterion of showing usage among a significant number of people would eliminate words pertaining to subjects like soccer highlights because it is unlikely that the majority of people in the firm would be interested in the same highlights or even soccer. It is possible that general news may pass this criterion, but if it reaches a significant number of people, then the news is more likely to describe an important event and would be considered informative. One counterargument against this criterion is that it is still possible for uninformative words to reach a significant number of people, like spam or American Idol chatter. For the spam example, because spam related-words do not diffuse through network, but are sent en mass to the receivers, they would not be among words we would be interested in. For the American Idol chatter example, it is possible that the related words diffuse through the network and reach a lot of people. In these cases, we would have to deem the words diffusive and informative, not because words may not be informative related to work but because the words are informative on information flow and interests of the firm.

The criterion of showing back-and-forth usage enables us to include words that do not reach lots of people but are exclusive pieces of information that are shared among smaller groups, like a project team. The counterargument for this criterion as a measure of informative level is that uninformative words can also be talked about among a small group of people back-and-forth. For example, an on-going
discussion about who is the best boxer ever can also fit this criterion. Although the words may be directly unrelated to recruiting efforts, the particular set of people involved in the best boxer discussion will be indicative of a clique and will provide insight into the social ties within the firm and the ties’ indirect effects on productivity. Combining both the diffusiveness and informative criteria, we come up with two different sets of words: event words, which are diffusive words that reach a lot of people, and clique words, which are diffusive words that show back and forth usage.

In addition to identifying words that are diffusive and informative, we also want to identify words that are non-diffusive, so that we can compare use of the non-diffusive words with our diffusive, informative word sets and to validate our process of identifying diffusive, informative words as distinct from non-diffusive, non-informative words. Non-diffusive words are words that originate from an individual’s thought processes instead of from another person’s communication. Some words we are confident are non-diffusive are words used in everyday email writing. This includes the articles “the”, “a”, etc. as well as words like “question” or “are”. We label these words as common-everyday words. Because common-everyday words are likely to have high frequency of use, we also want to identify words that are non-diffusive but have a lower frequency of use, such as high level specialized vocabulary. An example of a common – specialized word would be the word “fastidious.” This is in contrast to the more frequently used word “fussy.” We label these non-diffusive, low frequency words as common-specialized words.

With event, clique, common-everyday, and common-specialized words, we have the necessary categories of words with which to test our hypothesis of the positive correlation between favorable social network structural positions and the diffusion of informative and strategic information. The following sections describe several alternate methods for identifying words in these categories and the final process we use to create these word sets. We also provide justifications for why we choose certain methods over others.

3.2 Data Set

Before going into the processes involved in determining the word sets, it is important to document the nature and constraints of the data set used in this study. The email data set used in our research has some restrictions, mainly due to privacy considerations. The data collection process was designed to maintain the anonymity and privacy of the human subjects in our study and therefore has several unique characteristics related to the de-identification and hashed masking of the actual email content. The data was first processed through the Hashing/Capture module of EmailNet [31], an email extraction and hashing program, to add extra privacy protection. EmailNet was developed as part of an NSF project on information productivity and is a software system that automatically mines, hashes, and
analyzes email traffic for use with information productivity research. The process of the Hashing/Capture module has very important features that need to be explained in order to understand the approaches used in this research. First, the very common words, such as “the” are deleted from the email corpus. This feature is actually good for our analysis because we only care about informative, diffusive words and strategic information. Next, the words themselves are hashed into non-recognizable numbers. Although the mapping is consistent across the email data set, we lose the ability to check the accuracy of our research by human cognition of the data. This limits us to using automated and computational methods to determine and verify our results. Also, the order of words in the body of emails is removed and just the frequency of words is known. Therefore we cannot check a set of words in a particular order. The normal from, to, cc, date, and subject information is available, although the subject information is also hashed. The other fields are not hashed but de-identified during the research. The email data was collected at a mid-sized executive recruiting firm, headquartered in a large mid-western city. The period of analysis covers 41 weeks, approximately 10 months consisting of two periods from October 1, 2002 to March 4, 2003 and October 1, 2003 to February 11, 2004. These periods were chosen because of the validity of the number of emails recorded during these months. Also for this data set, there are some “duplicated” emails, emails that have exactly the same from, to, cc, date, and subject information. The duplicates are filtered and are not used as part of the results and methods in this paper or in Aral et. al. (2006).

3.3 Initial Filters for Diffusive Words

The overall design in identifying words that are diffusive and informative is to create a table of words with associated metrics and then to use those metrics to determine which of the words are diffusive and informative. As there are about 1.5 million different words in the email data set described above, calculating metrics for each of the 1.5 million words would be impossible because of the computation time and power necessary to finish the calculations. Therefore, we created quick (in terms of computation time and speed) initial filters on the 1.5 million words to reduce the possible set of words that need metrics to be computed. The initial filters try to quickly eliminate words that are likely to be non-diffusive (words that originate from an individual’s thought processes instead of from another person’s communication). This most likely includes words that have very high frequency. We would also want to eliminate extremely low frequency words. In the marketplace of ideas, words that are in demand are more interesting. Once we eliminated non-diffusive words (which are likely to be high frequency words), we want more frequent words because they proxy for an interest in or demand for the word. In the end, the filters’ main purpose is to reduce the set of possible words to a manageable level so that it is computationally possible to create relevant metrics for each word. We used three initial filters: a TF (term frequency) threshold, an exclude words in every week filter, and a TFCIDF (term frequency – cumulative
inverse document frequency) threshold. Each of these filters are described and explained in the following subsections as well as some alternative filters. After passing through the three filters, the original 1.5 million words are reduced to about 120,000 words, which we characterize as comprising the Candidate word set.

3.3.1 TF Initial Filter

Like all the initial filters, the purpose of the TF (term frequency) filter is to eliminate words that are non-diffusive and/or uninformative. This filter eliminates words that occur less than 11 times. This cutoff is based on previous literature [30], which does not explicitly state why 11 times is preferred to any other cutoff. However, it is inferred that 11 times is sufficiently high enough so that all words lower than this cutoff are not potential topics words and is sufficiently low enough so that the filter does not miss potential topic words. The filter eliminates over a million words from the original 1.5 million word set.

One reason in using this filter is to first eliminate words that only occur once. Obviously, words that occur only in one email do not diffuse; therefore we can immediately discard these words. A second reason why we eliminate words with TF $< 11$ is because words that are used so infrequently do not fit our two criteria of what a diffusive, informative word should be: words that reach a lot of people or show back-and-forth usage. Words that occur less than 11 times by definition are not used by the majority of people in the firm. This is also true for any values of TF around 11 (like 9, 10, 12, or 13), meaning our results are not sensitive to minor deviations in the neighborhood of this threshold. In identifying back-and-forth usage, words that occur less than 11 times may fit this criterion, but are less likely to be back-and-forth usage words than words that occur more frequently. For example, if two words both show that users sent and receive the words, and all else being equal, the word with the higher frequency is more likely to be diffusive and/or informative (more usage means higher possibility that diffusion did occur and that people are interested in the word). Our top clique words are not sensitive to low TF cutoffs such as 11 or any values around 11. Another reason we use the TF $< 11$ threshold is that it is a quick, effective mechanism for reducing the possible word set dramatically without sacrificing too many words that we would want to use. In one swoop, the filter reduces the computational footprint in calculating and determining the word sets by two-thirds.
3.3.2 Exclude Words in Every Week Initial Filter

The purpose of the exclude words in every week filter is to eliminate words that are used in
everyday email writing, such as the articles “a” and “the” or other common words like “are” and “reply”. As mentioned above, words that are used in everyday communication are non-diffusive since they usually originate from an individual’s thought processes rather than from another person’s communication. The exclude words in every week filter simply eliminates all words that are used week in and week out. This means that all words that are in at least one email for every week in the data set are excluded. The reason behind using a week instead of a day is that it could be that a common word, like “reply”, may not be used in a given day (like on the weekend), but nevertheless be consistently used throughout the email corpus. The week time frame ensures that any random daily non-occurrences would not cause the filter to miss a typical everyday word. From a 500,000 word set after the TF<11 filter, the exclude words in every week filter reduces the possible word set to only about 495,000.

3.3.3 TFCIDF Initial Filter

The TFCIDF (term frequency –cumulative inverse document frequency) filter also eliminates words that are non-diffusive and is based on previous literature [30], but unlike the TF <11 filter, its purpose is to eliminate words that occur very frequently. Although the exclude words in every week filter eliminates words with high frequencies, it does not exclude the majority of them and it only eliminates about 5,000 words in total. The TFCIDF filter adds an additional safeguard in eliminating non-diffusive, high frequency of use words. TFCIDF is defined as a word’s current period term frequency over the average frequency in previous time periods and the formula is formalized above in Equation 2. The approach is to include words with a high TFCIDF. This yields words that become ‘interesting’ in a particular time period, instead of words that are used, on average, consistently in every time period. This fits our goals of choosing words that are not used in everyday writing and words that show a degree of informative value. For example, a low TFCIDF word could be “education.” If the firm has an education recruiting practice, it is highly probable that the word “education” would be used frequently. However, if during the winter time, there is no demand for education recruiting, then the word may not be used. The exclude words in every week filter would not eliminate this common word because of the absence of use during the winter. However the TFCIDF filter would probably eliminate it because on average, the TF never increases dramatically compared to the previous week’s levels (the word drops in average usage in the winter and would not spike up enough after winter to pass the TCIDF. If it does, then this would show unusual demand for education recruiting and thus make the word important). The authors of the previous literature using TFCIDF used a cutoff greater than three, which is high enough so that words that
occasionally go in and out of time periods would not necessarily get triggered as an ‘interesting’ word. For our computations, we use the same cutoff and a time period of a week. From the 495,000 words resulting from the TF <11 filter and *exclude words in every week* filter, we reduced the word set to about 120,000.

To compute the TFCIDF measures word by word from the resulting TF and *exclude words in every week* filtered word set, comprising about 495,000 words, would take an unreasonable amount of computation time. Instead, we compute the TFCIDF of all words, week to week, at the same time. This means holding thousands of words in memory, sifting through the email data week to week, and checking simultaneously if they pass the TFCIDF cutoff. The advantage to this approach is that the amount of time needed to filter all 495,000 words is much less, but the tradeoff is that more computational memory is needed. Using a high memory capacity computer system, we were able to execute the TFCIDF filter in a reasonable amount of time (6 hours).

In addition to the TFCIDF filter, we looked at other approaches that we thought would eliminate high frequency of use, non-diffusive words. This included simply using another TF filter that excludes words greater than some threshold. To come up with this threshold, we plotted the word frequency distribution to see if a pattern emerged that provides clues for determining the appropriate threshold. The frequency distribution plot is shown in Figure 3 below.

![Figure 3 – Frequency Distribution of All Words](image-url)
The first pattern that jumps out in the graph is that the Zipf Law relationship is not perfect. Higher frequency words seem to deviate from the straight log-log relationship. This is because the difference in the number of words at higher frequencies is in absolute terms small, but on the graph very large because of the log-log nature of the axis. Second, looking at the extreme cases, there are over 800,000 words that only occur once and there exist words that occur over 600,000 times. Given that there are approximately 450,000 emails, it is possible to imagine words that occur more than once on average per email. This set of words could include the company name, which may be part of employees' email signature, and some common words as well. Looking at the graph, there is no clearly definable cutoff for a threshold. One problem in using a frequency cutoff alone to determine which set of words are informative and diffusive words is that there is not a clear justification as to what constitutes a “high” frequency. Are 70,000 occurrences too much or 7,000? A reasonable solution might be to take some word count weighted average and then to take only words below that average. However, comparing this arbitrary TF upper threshold approach to the TFCIDF filter, we realized that the TFCIDF filter eliminates high-frequency, non-diffusive words without eliminating all high frequency words. It is definitely possible that there are words which are highly diffusive and also have high frequency of use. The TF upper threshold approach would eliminate these words as well. The TFCIDF filter also has a clearer justification of how it removes words. It does not automatically assume that high frequency words are bad, and instead includes words that become “interesting” at some point in time through relative levels of frequency of use, not absolute levels. Since most high frequency words usually do not become dramatically interesting in particular time periods, the TFCIDF filter would eliminate most high frequency words, but not all.

![Example One Cycle Word](image)

**Figure 4— Example One Cycle Word**

- 19 -
Another approach we did not take that we thought would eliminate high frequency of use, non-diffusive words was to look at the ‘cycles’ of word usage activity over time. We use the term ‘cycle’ to refer to a pattern of word usage such that a word is not used at all in previous periods, then starts being used over a set of consecutive periods, and finally stops being used in a subsequent period. Because people may not use some important words consecutively on a daily basis, the period used to capture this data is set to a week. An example of a one cycle word is shown below in Figure 4. Because words with constant frequency of use are more likely to be words that are used in everyday email writing and hence not diffusive nor informative, the hypothesis in using cycles is to exclude words that do not have any cycles, which are the everyday use words, and to retain words that have few cycles. Words with few cycles are likely to be uncommon words that are used only in a specific context or situation, and thus occur only in certain consecutive time period(s). An example of an everyday use word would be the word “question” and an example of a word with few cycles would be a candidate name for recruiting. Our ideal word, shown in Figure 5, would be a word that has exactly one cycle and shows increasing usage in activity and among people with activity dropping off gradually or immediately. Our notion of an ideal word comes from the assumption that word usage is directly correlated with the number of people involved with a word. As a word flows through the network and more people see or use the word, the word frequency will increase, implying the possibility of diffusion.

In considering this approach, we determined that there were multiple problems with the hypothesis that words with no cycles should be eliminated and only words with few cycles should be kept. First, not all common words are consistently being used week in and week out. Such common words may stop being used in a given week for random reasons and then afterwards, be used consistently from then on. In this scenario, common words that exhibit this pattern of usage have two cycles. As words with two cycles, they would be kept as possible diffusive, informative words, even though they are common words.
Second, diffusive informative words may be used to varying degrees over different periods and thus exhibit multiple cycles. Because the cycle approach would only keep words with few cycles, the approach would eliminate a possibly good set of diffusive, informative words. The last problem with a potential method using cycles as initial filters can be described as the difference between Figure 4 and Figure 5. The original thought process behind the use of cycles was the notion that diffusive, informative words have this semi-smooth, increasing ramp up in activity followed by either a quick drop or a gradual decline. Figure 4 clearly does not show this and after examining many other words with one cycle, we found that the constant increasing activity level idealized in Figure 5 seldom exists.

A third and final approach not taken that we thought would eliminate high frequency of use, non-diffusive words was to include analysis of the topics of our email data corpus. Topics, as defined by the field of text mining and clustering, are sets of words that are frequent among groups or clusters of documents. Topics summarize the content of clusters. The problem with this approach is in text mining and clustering, topics are calculated after the commonly used words are filtered out. This is because if they are not, the topics of a text data set would end up being the commonly used words since the high frequency associated with these words would make the words common in all clusters of documents. Since the purpose of initial filters is to eliminate these non-diffusive, high frequency words, topic filtering provides no mechanism to eliminate the words we want to exclude.

Having filtered out words that are unlikely to be diffusive or informative based on usage characteristics, we turned to the problem of identifying words in our four categories: event, clique, common-everyday, and common-specialized words. These words and the methods used to identify them are described in the next four sub sections.

3.4 Event Words

*Event* words are diffusive, informative words that reach a lot of people and are likely triggered by discrete events, such as external, ground-breaking news. From the *candidate* word set, which are words that passed through the initial filters, *event* words are identified by choosing words on two metrics: first, the words are used by more than 30 people and second, the words have a high (one standard deviation above the mean) coefficient of variation of activity (CVA) across weekly usage. The reason behind the first metric is obvious. *Event* words are words that reach "a lot of people." However, the exact value of 30 is not obvious. We chose this cutoff by examining the distribution of number of words versus the number of people a word has reached for the *candidate* word set, which is plotted below in Figure 6. From this graph, it is hard to say what constitutes "a lot of people". Like the frequency distribution graph of Figure 3, there is no obvious way to determine a good cutoff. An arbitrary cutoff can again be some median statistic of the number of people. Another can be that when a word reaches the majority of the
people in the network, it can be said to reach "a lot of people." A more solid justification of what constitutes "a lot of people" is matching the frequency to the number of people who use common words in everyday communication. Surely, "a lot of people" use everyday words. Words used in everyday communication are the words used week in and week out as described in Section 3.3.2 above. To find a value for the "a lot of people" cutoff, we plotted the histogram of the distribution of everyday words over the number of people who used the words, shown below.
First, we see a pattern that seems relatively normally distributed but slightly skewed with a mean of about 58. The most important insight from the distribution is the fact that the number of words starts to ramp up (conservatively speaking) at around 30 people. This is a clear, empirically determined demarcation of how many people use these everyday words and a solid justification for using a value around 30 as an initial value for what constitutes “a lot of people” in this network.

As mentioned before, the second metric used to determine event words is having a high coefficient of variation (CVA). CVA is defined as:

\[
3) \quad CVA = \frac{\sigma_{activity}}{\mu_{activity}}
\]

where

\[
\mu_{activity} = \frac{1}{41} \sum freq_{week}
\]

\[
\sigma_{activity} = \sqrt{\frac{\sum (freq_{week} - u_{activity})^2}{40}}
\]

CVA is the standard deviation of the number of emails per week over the mean number of emails per week (there are total of 41 weeks in our data set). This formulation normalizes words that appear in a large number of emails versus those that appear in a small number of emails and helps us to compare their frequency variation with each other. The motivation behind using CVA was taken from the chatter/spikiness distinctions used to describe different topics in blogspace [30]. We use high CVA as a selection criterion to make sure that the words that reach “a lot of people” are also highly diffusive. A word with a high CVA is a word that has burst(s) of activity in some weeks relative to other weeks. The likelihood that a lot of people suddenly use a word more frequently than usual from their own thought processes and at the same time is small. These bursts of activity are an indication of a word being spread through a diffusive process. For example, the word “Yahoo” is a possible non-diffusive high level vocabulary word that “a lot of people” in the firm might use once-in-while in their email. If, however, “Yahoo” is all of a sudden being used by the majority of the firm on the same day, it is highly improbable that the people are using the word in their own individual usages all at the same time. It is more likely the word is being passed around from person to person, describing some event concerning Yahoo, such a new product announcement or company earnings.

After combining both the criterion - that the number of people that use the word is greater than 30, and that the words have a high CVA - an event word set of 3,275 words is identified from within the 120,000 word candidate word set. An example of an event word is shown below.
One approach not taken that we thought would be a good filter in choosing event words is to take all the words from the candidate set of words and keep only the words that reach the most people. This is similar to the mechanism described above in choosing event words, but it does not have the high CVA component. Conducting sanity checks on words chosen in this manner, we plotted numerous frequency graphs like the one shown below.
At first glance, the words seem very indicative of diffusive, informative words. First, the number of people the word reaches is almost all of the people in the social network. Second, the rate of diffusion is fairly consistent; there is no sudden increase in the number of new people seeing the word. This is in contrast to an everyday usage word, which would see a dramatic increase in its use across individuals early on. For example, we expect to see the word “the” used by all individuals in the firm on the first day of observation. However, these graphs also look similar to a random word generating pattern, where individuals in the network use words over time in a manner such that the likelihood of using a word resembles an exponential or Poisson distribution. We next simulated what random exponential generated words would look like (two are shown below in Figure 10) and found that the similarities to Figure 9 are striking. This means that words represented by Figure 9 (words from the candidate set that reach the most people) are non-diffusive since they may also be ‘generated’ from individual thought processes instead of another person’s communication. Therefore, words chosen based solely on reaching large numbers of people are most likely not the diffusive words we want for our event word set.

![Random Word Diffusion Patterns](image)

**Figure 10 – Random Word Diffusion Patterns**

### 3.5 Clique Words

Not all words that are diffusive and informative also reach a lot people. In the context of the firm we are researching, some specific client names and industry knowledge are likely to be known only to small groups of people in the firm for whom that information is most relevant. For example, the name of a new candidate for a nursing position may only be relevant to employees looking to fill nursing positions but not to other employees. Because these words occur infrequently and are likely to involve the few people for whom the information is directly relevant, it is hard to distinguish these words from the average candidate word set. Figures 3 and 6 clearly show that the majority of the words in the email data
set are both infrequent and reach very few people. Our approach for identifying this set of informative, infrequent words emerged from a previous set of analyses (briefly described below) that was originally used to find diffusive words but ended up simply informing our search for these informative, infrequent words which we call *clique* words.

3.5.1 From/To

A direct way to find words that are diffusive is to follow the path of each word as it flows through the social network. Because email data records the exact time and people involved in an email exchange in which a given word is being used, it is possible to recreate the precise paths a word goes through. However, what we do not know is if a receiver of a word sends the word on as a consequence of having received it or whether instead they are using that word as part of their own usage (i.e. they would have sent it whether or not they received it from the original sender). A good criterion to distinguish between the two is to limit the time between receiving and sending a word. If the time parameter is small enough, then we can be more certain that the word was adopted when received and was then sent on as a consequence in part of the original receipt (rather than someone's own choice of word usage). Once we determine a good time parameter, the next step is to create tree graphs of words being sent and received under the time parameter constraint.

![Figure 11 - Tree Graph Type 1](image-url)
If a word is sent by person A and received by person B in an email X, and then person B sends the word on to another person in email Y within the time constraint, then a link is created between two nodes, emails X and Y, in our tree graph. If person B does not send an email with the word then no link is created (and node Y is not created either). If person B sends an email but after the time constraint, then a node Y would be created but no link between X and Y would be created. All nodes and links for a tree graph of a word are created in this manner. The links show which emails are “connected” through a diffusion process as defined here for a given word. An example tree graph of a word is shown in Figure 11.

Intuitively, a word that has lots of nodes with links and has good tree heights would be indicative of a highly diffusive word. Tree height is the largest number of consecutive links down a tree in a graph. For example, in the figure above, the height of the tree is four. Words with high tree heights mean that the words are continuously being diffused in their usage. This is in contrast to words with small tree heights, where words are being used in sporadic, separate diffusion processes. However, one problem with this type of tree graph is that although the word reaches only three people, there are nine nodes measuring how many emails the word is being used in. Any measure of diffusion with respect to depth, width, or the number of nodes in a tree graph for a word could be biased toward words with lots of emails. A word can be used in hundreds of emails and have links between them in the tree graph, but if the word only reaches two people, the word is still not diffusive. Another problem with the tree graph is that if a person (Person Q) receives a word, uses it immediately, and then also uses it again out of the blue a long time later, the latter usage would be shown as a node with no link to it. However, in this scenario, we believe that Person Q received the word from another person and used it in a diffusive manner. Any subsequent usage of the word from Person Q should be denoted as being used diffusively. The Type I graph does not indicate this correctly.
Because of these problems, another type of tree graph needs to be constructed that has links representing diffusion but gets rid of multiple reoccurrences of the same people in the graph (Tree Graph Type II). In this graph, the nodes should represent people, not emails. One idea is that every node in the graph represents the first use of the word by a person and is time stamped when this occurs. Next, if a person, represented by one node, sends an email to another person, who has a node in the graph with a subsequent time stamp within the time limit parameter, then a link is created between the two nodes. This algorithm eliminates any back and forth conversations people may have about the word and gets rid of the false no-link nodes described above. An example of this kind of graph is shown above in Figure 12.

After creating tree graphs of Type II for all the words in the data set, the next step was to find the words that are highly diffusive based on some characteristics of the graphs. One good measure is the average height of the trees for a given word. In both Figure 11 and 12, the words have exactly one tree. However, in many cases, words have multiple trees, with many trees consisting of just one node. The average tree height metric would force words with lots of one-node trees to be considered less diffusive. This is because these words are mostly being used by people who use the word based on their own usage rather than after having received the word in an email from someone else. After sorting on this metric, there are only ~3000 words with average tree height of greater than 1.5 (a one-node tree is considered to be height of zero). The more surprising results are that of these diffusive words, the average number of people involved in a word is only 3.24, the max number is only 6, and the average number of emails per word is only 7.78. These results are a departure from the number of people involved with the event words above. These intermediate findings indicated to us that there was some other category of words missing from our analysis and originally inspired our search for what we call clique words.

3.5.2 Endogeniety and Approaches

Armed with our new metrics of diffusion and the list of highly diffusive words above, we wanted to see if the flow of words adhered to the email social network characteristics of the firm. By construction, words identified by the method described in section 3.5.1 will have some degree of correlation to the social network structure of the firm. Since these words are chosen in part based on the characteristics of their email links between individuals (rather than on non-network based usage characteristics such as their frequency of use or their coefficient of variation), this process in essence ‘selects on the dependent variable.’ Therefore any results that come from analyzing correlations between these words and the structure of the email social network would be tautological. A good analogy to this problem is studying the innovation of the telephone only along the lines of telephone communication between people. A typical hypothesis is that a central person in this telephone social network would be the driver of telephone innovation. Because by definition he is in-between most people who have telephones, it may
seem that he is a driver. However, his centrality is not the source of the diffusion of innovation but is the result of it. Because of this endogeneity problem, we did not use characteristics of tree graphs to choose candidate diffusion words. However, as a result of this analysis, we were motivated to find words that are informative and diffusive, but that exist only among a small group of people and occur infrequently.

The approach to finding these infrequent, diffusive informative words is to choose words where people involved in the diffusion of the word both sent and received the word in email. Looking back on the tree graph analysis, one characteristic that is common to all of the top diffusive words is that each person sent and received the word. Using this as our metric where there is more than one person on our candidate set of words, we find that there are about 4100 words, which is a good sized list. The metric also makes sense because this means that the word is being used back and forth among the people involved with the word, and hence that the word is likely informative for that group and is being diffused among them. An example word of this set is shown below in Figure 13. Although the word was not selected on its continued usage or on the small number of people using the word, the graph shows that the word exhibits the desired characteristics along these two dimensions. After examining multiple graphs, we realized that the words identified using this new simple metric are indeed representative of the clique words we intended to identify. Further analyses into this set’s characteristics and statistics are explained in the next section titled Results.

![Figure 13 - Clique Words](image-url)
One approach not taken that we thought could also identify *clique* words was to filter the candidate set of words such that they exhibited the characteristics we want. This means choosing words that have a small number of people (say from 3 to 7), occur infrequently (11-50 times), have a high number users per email (greater than 3) and show long periods of usage after reaching all the people the word is going to reach. The last two metrics suggest that these words are being used in conversation and are still informative after being seen once. From this set of constraints on the *candidate* words, the number of words we find is about 2200. Although this is a good sized list, there are two problems with this approach. First, the choice of limit values associated with each constraint is arbitrary. Is there a reason for the number of people to be between 3 and 7? Why not 2 to 9 or some other range? Although the values may make some intuitive sense, there is no logical justification for a particular set of limits. We want to identify words that are only used by a small group of people, but how many people constitutes a ‘small’ number? Another problem with this approach is that even if we could justify the exact values for metrics such as frequency or number of people (like the TF Initial Filter), we would then not be able to state that *clique* words are distinct on those measures. For example, the reason why *clique* words would occur infrequently in comparison to *event* words is because we defined them that way, not because of some unique property of *clique* words. Choosing words only on the fact that people involved in the diffusion of the word both sent and received the word is a much better approach since the words selected exhibit the characteristic of low frequency but are based on a property of *clique* words that is unrelated.

### 3.6 Common – everyday Words

*Common – everyday* words are words that are used in everyday communication. These words could be used as a baseline data set for comparing *event* and *clique* words to make sure any correlations between their diffusion and social network characteristics are meaningful beyond those of normal, everyday words. In section 3.3.2, we described how to identify these words. In addition to choosing words that occur week in and week out, we want these words to have early rapid use (within the first month) among lots of people to show that the words are used by everyone on their own almost all the time. An example of a common –everyday word is shown below in Figure 14.

Notice how much more email activity this word has and how it quickly reaches 60 people in the social network. Although the activity varies from day to day, there is a constant use of the word (there is a period of inactivity at around 150 because of the separation of the two time periods used in the study). Words that reach the majority of people quickly are likely being used by everyone most of the time. This is indicative of words that used in everyday usage. High frequency and constant usage are also characteristics of words in everyday use.
3.7 Common – specialized Words

*Common – specialized* words are words that are also non-diffusive words, but unlike *common – everyday* words, these words occur with less frequency. Therefore, *common – specialized* words should come from the *candidate* set of words, in which most of the high frequency words are already taken out. *Common – specialized* words are likely to be high level specialized vocabulary words that a lot of people use only once-in-a while. Because “a lot of people” use these words, the first metric used to identify *common – specialized* words is the same cutoff as *event* words for the number of people, i.e. choosing words that are used by more than 30 people. However, unlike *event* words, *common – specialized* words should be non-diffusive. This means instead of a high CVA, *common – specialized* words should have low CVA. Low CVA words are words that are consistently used over time. If people use a set of words once-a-while but on own their own volition and “lots of people” use these words, then the aggregate word usage for this set of words for the whole firm should be fairly consistent over time. This is exactly what low CVA is and so *common – specialized* words should have low CVA. An example of *common – specialized* words (chosen on number of people greater than 30 and low CVA) is shown in Figure 9 above.

3.8 Summary of Word Categories
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Construction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common – Everyday</td>
<td>Words that are used everyday very frequently such that most people use the</td>
<td>Words that occur in at least one email every week and have reached the</td>
</tr>
<tr>
<td></td>
<td>words most of the time in the course of everyday language.</td>
<td>majority (over 36 people) of the network population within a month.</td>
</tr>
<tr>
<td>Common – Specialized</td>
<td>Words that are used consistently over a period of time but that are used</td>
<td>Words that reach greater than 30 people and have low (one standard deviation</td>
</tr>
<tr>
<td></td>
<td>infrequently by any given individual.</td>
<td>below the mean) coefficient of variation of activity (CVA).</td>
</tr>
<tr>
<td>Event</td>
<td>Words that seem to be triggered by an event such that use of the word</td>
<td>Words that reach greater than 30 people and have high (one standard deviation</td>
</tr>
<tr>
<td></td>
<td>spikes after a given date and reaches many people in the organization.</td>
<td>above the mean) CVA.</td>
</tr>
<tr>
<td>Clique</td>
<td>Words that are used only among a small group of people exclusively and show</td>
<td>Words that are used among more than one person where each user both receives</td>
</tr>
<tr>
<td></td>
<td>back and forth (&quot;conversational&quot;) usage.</td>
<td>and sends the word.</td>
</tr>
</tbody>
</table>

Table 2 – Word Categories

3.9 Note - Words in Each Category

For each of the four categories, we have identified words that exhibit the desired characteristics as described above and throughout the paper. The list of words in each category is not, however, a comprehensive list of possible words in any given category. It is important to stress that it is not necessary for our methods to include all words of a given category in the email corpus, but that the number of words in each category is reasonable enough to perform further statistical studies and that we are confident that the words in each category are the most representative of their respective categories.

4 Results

The next step after defining and justifying the four categories of words (e.g. common – everyday, common – specialized, event, and clique words) is to make sure these categories are distinct from each other and exhibit their defined characteristics independent of the metrics used to define them. Below, Table 3 describes all the metrics with their associated name and Table 4 presents summary statistics for the words in these four categories and highlights both the representativeness of the words in each category and the distinctions between categories.
Table 3 - Metric Definitions

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numemails</td>
<td>The average number of emails containing the words in each category.</td>
</tr>
<tr>
<td>Numpeople</td>
<td>The average total number of people to have used the words in a given category.</td>
</tr>
<tr>
<td>CVA</td>
<td>The weekly coefficient of variation of activity of the words in a given category.</td>
</tr>
<tr>
<td># Emails/Person</td>
<td>The average number of emails per person associated with a word.</td>
</tr>
<tr>
<td>Diffusion Time/Total Time</td>
<td>The average time that the words take to spread to all eventual users of the word over the total time the words are used</td>
</tr>
<tr>
<td>Max People Added</td>
<td>The maximum number of people that see or use a word for the first time in a given day.</td>
</tr>
</tbody>
</table>

Table 4 - Summary Statistics

4.1 The common – everyday set

The common – everyday set represents words that are used consistently on a weekly basis as part of everyday language use. Although many of the common words were taken out of the data set through the use of EmailNet [31], there are still some that remained. Some examples could be words like “because” or “thanks.” The 488 words in this set are the words that reached half of the social network population the fastest (after exclusions based on initial filters) and were used in every week of our observation window. Because these words are chosen in this way, the max people added (the maximum number of new people to have used the word in a day) statistic is high by definition. Other statistics, independent of the selection methodology, can also be used to distinguish this word set. For frequency of use, number of people reached, and the number of emails per person, the words in this set are expected to have high numbers to show that they are commonly used. The averages in the results demonstrate that this is in fact the case. Frequency of use is almost four times more than the nearest word set, while the number

---

*These words are few because before hashing the email data, common words (such as “the” or “a” etc) were already taken out and there are not many of them to begin with (Zipf Law).

**Numbers in parenthesis represent standard deviations; the rest are all means.
of emails per person is more than 25 times the nearest set. The average number of people that used the word is about 50% higher than the nearest set. One other statistic of distinction is the coefficient of variation of activity (CVA). Words that are used on an everyday basis are used by all (or most) people often and on a consistent basis over time. Low CVA words are words that are consistently used across time. Because common – everyday words are words used on an everyday basis, they would have consistent frequency of use over time and therefore should have low CVA. This word set clearly shows a low CVA in comparison to the other sets. To make sure that all of the four distinctions are statistically valid, we performed t-tests (exact results in Appendix B.2) and Wilcoxon rank sum tests to compare summary statistics across word sets and found that all the averages display statistically significant differences. All of these statistics also imply that common – everyday words are the least ‘sticky’ pieces of information in comparison to the three other word sets. This is because a) almost all of the people in the firm use the words more frequently relative to the other sets and b) the words exhibit usage patterns that are consistent with words generated from individual thought processes rather than as a result of receiving the word from another person’s communication. The first reason implies that the information is common and easily understood or usable, and the second implies that there is no need for high or specific levels of social background context to understand the information being represented. Also, because of the low CVA, common – everyday words exhibit the chatter characteristic described in Information Diffusion through Blogspace [30] paper.

4.2 The common – specialized set

The common – specialized set of words is also thought to be non-diffusive but should be less frequently used than the common – everyday words. The words in this set represent high-level vocabulary that people use once in a while in their communication patterns. Examples of such words are “[job] satisfaction” or “[value added] matrix.” Common – specialized words are slightly stickier than common – everyday words because they occur less frequently, although both word sets would be classified as non-sticky because of their non-diffusiveness. Because these words are chosen based on the number of people (greater than 30) that used word, the fact the words have greater numbers of users than clique words is not a valid dimension of distinction across word sets. However, if the words have more people than the words in the event set, it is a valid criterion on which to evaluate whether our methods are producing distinct categories of words in each set as both were chosen on this same criterion. The main hypothesis behind common – specialized words is that they are in contrast to the event words, which also reach a lot of people, but are more likely to do so in a non-random, diffusive pattern of information flow through the social network. One way to show this is that non-diffusive words, like common – specialized words, on average, should be used by more people and more frequently than diffusive words because diffusive
words are constrained by the fact that a person has to receive the word before passing it on. On both counts, this assumption is valid and statistically significant (greater than 99% confidence) with T-statistics assessing the differences of mean values for the number of emails at 7.56 and the number of people at 42.31. Another distinction that needs to be demonstrated is that common – specialized words should not have large daily jumps in the number of people using or seeing the words for the first time. Use of these words should roughly follow an exponential or Poisson distribution and should, by our definition, be randomly used across people, and so it is highly improbable that there are big increases in the number of people using the word on the same day. The maximum people added statistic shows this is the case in comparison to the event words. In comparison to clique words, although the maximum people added statistic is greater in common – specialized words, if the statistic is normalized over the average number of people using the words, common – specialized words show lower percentage jumps, which validates the hypothesis that common specialized words are more likely to be words used randomly by people in their everyday language. In comparison to the common – everyday words, common – specialized words also show a lower average of maximum people added, but because the words in the common – everyday set are chosen on how fast they reach the majority of the people in the social network, this is a less informative distinction.

4.3 The event set

The words in the event set are words that should be highly diffusive in nature and are thought to be triggered by some event, which could be some external news, internal firm-wide changes in management practices, or a variety of other catalysts. The words are defined as those that reach a significant number of people and have a high CVA, so using these statistics as measures of distinction when comparing event words to other words is not necessarily valid. The main characteristic that should hold in this word set is that there should be ‘jumps’ or spikes in the number of people who use the word on a particular day. Such spikes in activity indicate that these words are not necessarily used by people in the course of their everyday language and instead are received then spread to others after an event in time. Ideally, events words should have the highest max people added average out of all the word sets because event words are chosen based on spikes in usage. However, event words have only the second highest maximum people added average. Common – everyday words have the highest average, but because common-everyday words were chosen based on the rapid increase in people added and at the same time have a low CVA (which implies that they are not diffusive), the fact that the common – everyday word set has a higher maximum people average than event words is less meaningful. To make sure the distinctions on max people added are statistically valid when compared to the other word sets (clique words and common – specialized words), we performed t-tests and Wilcoxon rank sum tests and found that all the
averages are statistically significantly different. Given the diffusiveness of event words, the words are stickier than either common word sets. This is because, event words are less frequent than either common sets and exhibit the constraint that people need to receive the words before passing them on. Also, because event words are characterized by their spikes in activity and people usage, event words can be considered to be spiky as described in [30]. In performing future diffusion studies with event words, the words should be used as a source of diffusion order information. On average, event words reach 36.31 people, thus being first out of thirty people is more significant and informative than being first out of three as with clique words. Therefore, using event words in first adopter as well any dyadic relationship innovation diffusion studies can provide more significant results. Also for future diffusion studies, event words are more representative of the disease-like propagation within the social network than clique words. This is because event words spread rapidly among the population and exhibit contagiousness.

4.4. The clique set

Clique words are words that are used among a small group of people and are usually infrequent, esoteric words. These words represent in-the-know information and could be important client names or specific industry knowledge known only to people on projects or in teams that have a particular reason to know about that given piece of information. For example, if a team of recruiters were trying to find a triage nursing candidate in Paris, TX, members of this team are more likely to know or care about nursing candidate names in Paris, TX than members of a team seeking CIOs in New York. Clique words are defined by the fact that all individuals that use the words must both send and receive them and that the number of people using them must be greater than one. Clique words can therefore be distinguished on all of the statistics in Table 4 as none of these statistics are part of the identification criteria for this class. However, the important distinctions are that a) clique words should have a smaller number of people using the words, b) they should be used infrequently, and c) they should be used in conversations among the word’s users. This last distinction demonstrates that clique words are diffusing back and forth among users in a small group or clique. The statistic that represents this distinction is the diffusion time / total time measure. This statistic is the average time that the words take to spread to all their users over the total time the words are used. The lower this ratio is, the more the word is being used back and forth among its users. On all three of these distinctions, clique words are shown to be distinct according to all of our assumptions. To further illustrate how distinct clique words are according to their frequency of use and the number of people involved, histograms of the measures are shown in Appendix B.1 and B.2 for each word set. By looking at the x-axis dimension alone, one can immediately tell that clique words are clearly distinct on these two measures.
Table 5: Word Categories, Knowledge Types, and Diffusion Models Summary

Given the infrequent and diffusive nature of clique words, the words can be considered the stickiest out of the four word sets. For future diffusion studies, clique words are probably better suited for ego-centric innovation diffusion studies because clique words represent highly specific, in-the-know information among small groups. This means that knowing which individuals are involved in a greater number of clique words is more significant than knowing which individuals are involved in greater number of event words, since most people would have used or seen event words. One hypothesis with clique words can be that central social network members would be more likely to see clique words. Also for future diffusion studies, the way clique words (more so than event words) are received or used seem to follow the adoption and link formations games described above in terms of game theory models. This is because from individual member viewpoint, a member will only adopt or create links with only certain other members of social network. This decision is highly selective and probably made to maximize the expected value of specific project dependent information. Also like adoption and link formation games, the back-and-forth activity among the users is vital to assessing adoption (word usage).

5 Extensions

Although the four categories of words above are shown to be distinct on diffusive and informative measures, the next step is to use these sets of words with individual productivity and social network analysis. For example, one hypothesis could be that people who are highly central in the firm's social network are more likely to see clique words. Another hypothesis might be that structurally equivalent people in the social network see event words in about the same order. These as well as a myriad of other possible hypothesis about the email social network and productivity are beyond the scope of this thesis, but we anticipate that these four sets of words can be used to answer questions related to the diffusion of information through social networks. To enable empirical analysis on these questions, we created data sets for words in each set category which show the order in which the word is diffusing.
among the people in the network. The data sets also highlight which person is the first person to use the word, so that any hypothesis about first person usage can also be tested. A sample of a table in the data sets looks like this:

<table>
<thead>
<tr>
<th>Word</th>
<th>First Person</th>
<th>Person A</th>
<th>Person B</th>
<th>Person C</th>
<th>Person D</th>
</tr>
</thead>
<tbody>
<tr>
<td>-107005113615236</td>
<td>Person Q</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-117639381805308</td>
<td>Person E</td>
<td>0</td>
<td>36</td>
<td>29</td>
<td>27</td>
</tr>
<tr>
<td>-126864802000474</td>
<td>Person A</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-129071568354704</td>
<td>Person F</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-133800693218714</td>
<td>Person M</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

*the dots mean there are more possible data points to the right and below, but not shown

Table 6 – Sample Data Set

There are 3 kinds of tables, one with order information (like the one shown above), one with just a binary variable indicating usage or no usage, and one with the exact times of first usage of the word. Since there are four categories of words, this makes 12 tables in all. For more information about this and the other data generation processes used in paper, please see Appendix A.

6 Conclusion

After analyzing and filtering all the words in the email data set, we established four categories of words that are statistically distinct on diffusive and informative measures. The common—everyday words are the base set in which words are used in a non-diffusive manner and are words used in the course of everyday language in most emails. The common—specialized words also exhibit no diffusive characteristics, but are hypothesized to be specialized vocabulary used infrequently in everyday communication. Event words are words that are most likely to diffuse and reach many people. Although these words are probably important and informative, because they reach, on average, the majority people of the firm, they are not scarce pieces of information. The final set of words, clique words, are words that are diffusive and scarce pieces of information that are used by people only in-the-know for a particular project, team, or specific industry knowledge. Using these sets, further studies can be done with social network characteristics and information worker productivity. The end goal for these categories of words is to provide useful insights into the diffusion of information in an email network and the role of information in the context of the firm.
Appendix A: Data Generation Processes – Software Architecture, Programs, and Algorithms

A.1 Overall Program Design and Architecture

The majority of the data extraction and analysis of the email data was done via programs created by the author in the Java programming language, version 5.0. The data sets used for graphs and statistics were created by individual programs tailored to created data specific to each graph and/or statistic. However, an overall architecture exists for all the programs so that the code could be reusable and extensible. Matlab was used to create all of the graphs used in the paper. The program architecture was split into three parts, data extraction, data filtering/analysis, and data output. In some cases, the data outputs produced by the programs were then re-used in the data extraction process for further analyzing and filtering. A simplistic diagram of the architecture of the programs is shown below.

![Diagram of Program Architecture]

**Figure A-1: Design Architecture**

A.1.1 Data Extraction

The main data set used in this paper was on a MySQL database server and taken only from one table. This table was organized so that each row recorded the information for each email that was obtained from the firm. The columns for the table are: `emailID`, `dateStr`, `subject`, `froms`, `tos`, `ccs`, `body`, `size`, `attachNum`, and `attachType`. The last three columns, `size`, `attachNum`, and `attachType`, were not used for any part of the data generation process. The example some rows of data looks like this:
Table A-1: Sample Rows in Email Data

<table>
<thead>
<tr>
<th>emailID</th>
<th>dateStr</th>
<th>subject</th>
<th>froms</th>
<th>Tos</th>
<th>ccs</th>
<th>Body</th>
</tr>
</thead>
<tbody>
<tr>
<td>0A2A50C645...</td>
<td>10/23/2002</td>
<td>...651101709952</td>
<td><a href="mailto:a@b.com">a@b.com</a></td>
<td>Person A</td>
<td><a href="mailto:g@h.com">g@h.com</a></td>
<td>...3454&lt;3&gt;</td>
</tr>
<tr>
<td></td>
<td>6:57:41 PM</td>
<td>53436:...</td>
<td></td>
<td></td>
<td></td>
<td>;2342...</td>
</tr>
<tr>
<td>C6433531A8...</td>
<td>10/24/2002</td>
<td>...544231114799</td>
<td><a href="mailto:c@d.com">c@d.com</a></td>
<td>Person C;</td>
<td><a href="mailto:i@f.com">i@f.com</a>;</td>
<td>...1348&lt;1&gt;</td>
</tr>
<tr>
<td></td>
<td>7:46:04 AM</td>
<td>7378:...</td>
<td></td>
<td>Person E;</td>
<td>Person Z;</td>
<td>;67834...</td>
</tr>
<tr>
<td>3531438019D</td>
<td>10/24/2002</td>
<td>...203720221378</td>
<td>Person Q</td>
<td><a href="mailto:e@f.com">e@f.com</a></td>
<td></td>
<td>...8238&lt;7&gt;</td>
</tr>
<tr>
<td></td>
<td>10:47:14 AM</td>
<td>686:...</td>
<td></td>
<td></td>
<td></td>
<td>;49423...</td>
</tr>
</tbody>
</table>

There are a couple things to note. The `emailID` can be up to 255 characters in length and uniquely signs each email with an ID. The `dateStr` is the string with the format shown above. Because this is a string, and not a `datetime` field that can be used in MySQL, any data extraction and analysis done on strings has to be converted into a `datetime` object so that the emails can be compared and analyzed sequentially. Both the `subject` and `body` fields are both shown with "..." and lots of numbers. The "..." represent further possible numbers and data, but were not included in the table above to conserve space. Each set of numbers represent a word that is hashed and the `<?>` after the numbers represents how many of the words exist in the data for that field. For the `froms`, `tos`, and `ccs`, the real data set has valid email address and name information but were de-identified in our study for privacy reasons. Also for `froms`, `tos`, and `ccs`, individuals’ addresses are delimited by semicolons and for in-firm emails addresses, the name of the person was substituted. Based on this format of the email data, we created a separate Java class, `MyEmailDBHelp`, to interact with and extract from the MySQL server and provide one more step of security and privacy of the data.

A.1.1.1 `MyEmailDBHelp

This class uses two other classes called `DriverInfoBean` and `DriverUtilities` to correctly identify and call the MySQL database. The `DriverInfoBean` class is a simple Java bean that stores relevant database server information, which includes the `vendor`, `description`, `url`, and `driverClass` strings. The vendor string holds which vendor the database is from; in our case, it is MySQL. The description provides a general storage area to hold any pertinent data regarding how the database is suppose to be used; the url is the exact URL needed to access the database, including the port number and protocol. The `driverClass` string indicates the name of the Java class that will be used to call the various SQL queries on the database. This class implements the JDBC 3.0 specification for database access. For our data set, we used the `mysql-connector-java-3.1.7` set of drivers.

The `DriverUtilities` class provides a convenient method for converting the data in `DriverInfo`, so that access to the database would be automatic for our `MyEmailDBHelp` class. It has three main functions...
loadDrivers(), getDriver(String vendor), and makeURL(String host, String dbName, String vendor). The loadDrivers() method uses the current SQL vendor info and creates DriverInfoBean objects for a variety of standard SQL systems. This includes the MS Access, MySQL, and Oracle. The getDriver(String vendor) method returns a specific DriverInfoBean object that has all the relevant information on the input vendor. The makeURL(String host, String dbName, String vendor) method uses the input strings to automatically produce the appropriate URL so that another object can access the database without the need to know the exact procedures to do so.

Using both DriverInfoBean and DriverUtilities classes, MyEmailDBHelp object creates the initial parameters to access the MySQL database. It does not, however, make any connections to the database until another client object requests that it does. This is because MyEmailDBHelp wants the client objects to have full access to the amount and types of the connections there are to the database. This type of access provides micro-management capabilities in handling the memory allocation for the each of the connections, which may require an excessive amount of memory due to the amount of information stored in the database and the potential size of a resulting query. Once the MyEmailDBHelp is initiated by a client object, the client object can request a connection to the database and execute and corresponding method call to the database for particular way of data extraction. Because each filter and analysis has their own specific needs in terms of data extraction, methods are created within the MyEmailDBHelp class to facilitate the variety of types of transfers. More details on the different methods and algorithms are explained below in their corresponding type of filter and analysis.

![Diagram](image)

**Figure A-2: MyEmailDBHelp Design**

After a connection to the database is used by a client object of MyEmailDBHelp, the client object should close the connection down again via MyEmailDBHelp because it contains useful methods to do so.
that take care of exceptions that occur during the process. This also includes any run-time errors and the dumping of the memory used during the connection. To summarize, a diagram of how MyEmailDBHelp fits to the overall programming architecture is shown above.

A.1.1.2 Other Data Formats and Utilities

Although the main data set and hence the main data extraction utility is related to the email records, there are other formats of data and their extraction utilities that are used in the programs to create the statistics and graphs used in the paper. The main non-Java one is CSV text files. The CSV format is used to denote table information like the ones in Microsoft Excel. Each line represents a row and the commas delimit the cell information. Because this format is used for both spreadsheet programs and Matlab, the author decided that a fully functional utility to handle and extract CSV was called for. Therefore, the author used pre-made, free, Java utilities called CsvReader and CsvWriter. CsvReader extracts data in CSV text-based format with options to extract the headings of the columns. It does not allow any access via row and columns numbers and reads the data one line at a time. This forces the program and methods that used these utilities for data extraction to implement creative algorithms to analyze the data quicker and efficiently.

In addition to the data extraction of CSV text files, the programs also extract information from Java objects. The main class type of the Java objects is the Hashtable class. There are also other user-created Java classes that follow the Java bean design specifications. In both types of classes, the programs can simply access the data through methods calls on the respective Java types. More specific information on the class specifications are explained in the corresponding data analysis and filters below.

A.1.2 Data Filtering and Analysis

The data filtering and analysis portions of the system architecture consists of either individual Java classes or methods within those classes that execute the algorithms necessary to do the data manipulations. Each class has a set of methods that are related together in terms of their contribution to the thesis project. In addition to the Java classes and methods for data filtering and analysis, some of the statistics were directly derived from the MySQL server. This is because MySQL has built-in functions that efficiently compute the statistics. More detail on each of the programs and methods used for data generation is described below.

A.1.3. Data Output

The types of data outputs, which are the MySQL database, CSV textfiles, and Java objects, are used interchangeably based on the needs of the data filtering and analysis programs and methods. The
major data metrics used to create the word categories were outputted to a MySQL table so that easy retrieval and storage are possible. Some data outputs were again used as inputs for other programs and methods to continue the data analysis and filtering. The specific instances when this happens as well as the exact formatting of the data are described below for each of the programs.

A.2 Specific Programs' Design and Algorithms

The section covers the programs used to create all the data sets and statistics from the email data. The first sub-section Initial Filters discusses all the relevant steps necessary to perform all the computations in the Initial Filters section in the body of the thesis. The next sub-section, Statistics Generation, describes how all the relevant statistics and measures were created for all the words in the email data set so that the four categories could be identified and verified. The third sub-section, Tree Graphs and related programs, describes the programs and algorithms used to create the tree graphs mentioned in the Clique section in the body of the thesis. Although this work has no direct relation to the statistics and measures used for this thesis, the work, as mentioned in the body of the thesis, inspired the creation of the clique word category. The last sub-section, Social Network-related data generation, describes all the work done in creating any useful social network statistics and measures, including the data set described in the Extensions. Most of this work is for future social network analysis and hypothesis testing. As mentioned above, all of these programs and algorithms follow the overall architecture design laid out in Figure A-1.

A.2.1 Initial Filters Design

Before any of the initial filters were identified and created, the first step in getting a feel for the email data was to create Figure 3, a histogram of frequencies of the words in the data set. Considering that there are over 1.5 million words, the task is not as simple as the figure represents. The first thing to do is to create a list of all words in the email data and then record how many times they occur. Because the frequencies of the use of a word were recorded in the body of an email, the second task was to simply add these up. For the first task, to record all possible words, required to create new data structure consisting of the word, and frequency pair, like one show in the figure below:

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>3320873844605058910</td>
<td>102</td>
</tr>
<tr>
<td>6319955323910293655</td>
<td>794</td>
</tr>
<tr>
<td>1964486070286230619</td>
<td>34</td>
</tr>
</tbody>
</table>

Table A-2: Word/Frequency Pair
The data structure was implemented as a Java Hashtable object. The algorithm to fill this data structure was to simply go through each email, get the body, parse through to get at the words and their corresponding frequencies, check if the word already existed in the Hashtable, and if it did, then add the current frequencies with the new one in the current email, and if it did not, add the word and the frequency to the HasTable as a new row. Pseudo-code of the algorithm is shown below:

```java
Hashtable-structure hs;
for each e:email in <Email Data Set>
begin
    body = e.getBody();
    for each:w word in <body>
    begin
        freq = getfreq(w);
        if hs.contains(w)
            pf = hs.getfreq(w)
            hs.put(w,pf+freq);
        else
            hs.put(w,freq);
        end if
    end
end
```

Once we had the Hashtable filled up, the next step was to flip the keys and the values, so that we can have for each frequency, all the words that are associated with it. For this step, we had to be a little more creative because the Hashtable with word as keys and frequencies as values already took up most of the memory available on our computer system. So what we did is divide the 1.5 million words into 30 parts, with each part holding 50,000 words. Then with each part, we flipped the keys and values so that it would be in the form we can used to create Figure 3. The algorithm to flip the keys and values is similar to the one above which is:

```java
for each of the 30 parts of hs
begin
    Hashtable-structure flipped;
    for each w:word in <hs-part-i>
    begin
        pf = hs.getfreq(w);
        if flipped.contains(pf)
            cs = flipped.getwordsize(pf);
            flipped.put(pf,cs+1);
        else
            hs.put(pf,1);
        end if
    end
end
```
Now, because the data was divided in 30 parts from the original Hashtable to save memory, the new Hashtables that has frequencies as keys and the list of words as values does not have correct values. The next step is to combine the 30 new Hashtables into one data structure so that we do have the correct values. To do this, we simple load the 30 Hashtables, cycle through each frequency and then match any same frequencies with any other Hashtable and then add the numbers together. Finally, we output this data into a CSV file, where each row is a frequency and number of words pair, so that Matlab can read it. Then, use the plot command on Matlab to make Figure 3. Remember, although the outputted CSV file has only frequency and number of words pairs, the final Hashtable has frequency and a list of words pair. This data can now be used for the Initial Filters described next.

A.2.1.1 Cycles Filter

One of initial filters described in the paper is the cycles filter. Although we eventually did not use this filter, the process in which it was created required a good deal programming and an intermediary output was used for the next filter TFCIDF. As the reader may recall, a cycle is when a word is not used at all in a previous period, then starts being used over set of consecutive periods, and stops being used again in a subsequent period. The period used was a week. The first step to figure how many cycles are there for each word is to record the activity of each word on a weekly basis. This is the intermediary output that can be used for both cycles and the TFCIDF analysis. This requires sorting the email data in sequential order then going through each email for a week, storing the words in those emails and counting how many emails those words were in. All of processing required constant access to the MySQL server so computing this intermediary output of activity was mostly done in a method in the MyEmailDBHelp class. An outside client object simple just called this method.

Using this activity data, the next step in the algorithm is to create two Hashtables, one for the current week and one for the previous week. Because the first week of our data set does not have a previous week, the algorithm starts from the second week. As the algorithm progresses week by week, the current Hashtable becomes the previous week’s Hashtable. In each Hashtable, what is stored is the respective week’s list of words and their activity count. The algorithm compares the previous week’s Hashtable activity versus the current week’s and determines if there is a jump up or down in activity (from zero to some activity or from some activity to zero). If any jump is detected, the word goes into another Hashtable called Cycles that records the word and how many half-cycles it currently has. Remember, a jump (either way) only indicates the start or end of cycle, not the whole cycle itself. That is why the Hashtable is recording only half-cycles. If there are no half-cycles recorded, then the algorithm adds in the word and records its first half-cycle. In addition to half-cycles, the Cycles Hashtable also records the time in which the jump occurs. Knowing the exact times for the jumps allows for further
computation of the average time for each cycle for a given word. As weeks pass on, more and more words are recorded into the Cycles Hashtable. Because of the extent of the size of possible words (1.5 million), the Cycles Hashtable gets to take up a lot of memory space. So in order to conserve memory, once a word reaches 6 half-cycles, the word is removed from the Hashtable and put out of memory because all we care about are the words that have few cycles. After all the weeks are analyzed, the remaining list of words in Cycles is analyzed for the average time of the cycles for each word. This simply means computing the difference in times for the appropriate jumps, adding up the differences and then dividing by the number of cycles. Once this information is recorded, the words and the cycle statistics are then stored as Java objects. Because this list would have been used so that other filters can analyze and reduce the words to useful categories, the list was divided up into multiple parts for splitting up the work to multiple computers. However, because of the problems with using cycles as a filter to take out common words, this list was never used in the final statistics. Below is a flow diagram of the algorithm that summarizes how the cycles were computed for each word in the email data set.

![Figure A-3: Cycles Algorithm Flow Chart](image)

A.2.1.2 TFCIDF and putting it all together

After analyzing and discarding the cycles filter, the next filter we created and used is the TFCIDF filter. In the body of the paper, a brief explanation on the algorithm was already explained. This section
will review some of it and then go into all of the initial filters we came up with to produce the starting candidate set of words.

The TFCIDF filter finds words that record a jump in weekly usage activity that is 3 times more than the previous weekly average. To do this, the activity Hashtable described above is used. The algorithm for the filter examines the weekly activity data and compares previous weekly activity to the current week's activity for all the words. Because TFCIDF only cares about jumps in activity (and not sudden stops, like the cycle filter), the algorithm only needs to check all the words in the current weekly activity Hashtable. The cycle filter algorithm had to also check the previous week's word for sudden drops in activity to the current week. To save memory and computation time, once a word records any jump in activity pass the 3 times threshold, then the word is put into another location in memory and the future occurrences of the words ignored. This makes TFCIDF a faster filter than the cycles filter, but at the same time, it is not a constrictive. Any word that can make through the cycle filter can make it through the TFCIDF filter because a jump from no activity to some activity is considered to be a jump of more than 3 times.

Once the TFCIDF filter algorithm was created, the next two sets of filters, the TF > 3 and taking out all of the commonly used words was simple. For the TF > 3 filter, the Hashtable of the histogram of all frequencies paired with the list of all the words with the frequencies was used. The filter simply recorded any word that have a frequency greater than 3 as its key. For taking out the commonly used words, which are words that are used every week, a careful analysis of the number of words per week revealed that in the weeks between the first set of 5 months and the second set, activity levels drop dramatically. Although the exact reasons are unknown, the author assumed that was an error in the email data set, so any words that mysteriously disappeared during the “transition” weeks was ignored. This produced the ~4000 words cited in the body of the paper. If the “transition” weeks were not ignored than the number of the commonly used words drop by half. Making sure that this half was correctly included as commonly used words, we looked at their corresponding characteristics of frequency of use and number of users of the words and found out that they were in line with the other half.

Combing the three filters, TF > 10, TFCIDF >3, and taking out commonly used words in the order shown below produced the candidate set of words. Although any order would have worked, this particular order is computationally the fastest.

![Figure A-4: Filters to Candidate Words](image-url)
A.2.2 Statistics Generation

Once the list of the candidate set of words was created, the next step was to create four categories from the list. To do this efficiently, a table was created in MySQL to store each word and a set of metrics that might potentially help in separating the words out into the categories. Below is the list of metrics with the corresponding description. Note that most of the metrics were not eventually used to determine the final four categories of words, but were created as a toolbox to see any interesting patterns in the data. The names of the metric are the exact names the programs and the database used to refer to them.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Avg (Stddev) in Candidate Word Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>numemails</td>
<td>The number of emails the word is in. This is a measure of the frequency of word.</td>
<td>45.63 (114.53)</td>
</tr>
<tr>
<td>nofroms</td>
<td>The number of emails the words is in that is not from someone within the firm. This can be used a measure of the likelihood the word is from spam emails.</td>
<td>36.48 (94.95)</td>
</tr>
<tr>
<td>numpeoplefrom</td>
<td>The number of the different people in the firm that only sent the word in email.</td>
<td>2.63 (4.96)</td>
</tr>
<tr>
<td>numpeopleto</td>
<td>The number of the different people in the firm that only received the word. This includes the people cc’ed.</td>
<td>9.38 (10.72)</td>
</tr>
<tr>
<td>numpeopleboth</td>
<td>The union of the two metrics above. The total number of people to have used and/or seen the word.</td>
<td>9.7389 (10.9624)</td>
</tr>
<tr>
<td>startdate</td>
<td>The exact time (month/day/year, time) in which the word first occurred.</td>
<td>N/A</td>
</tr>
<tr>
<td>enddate</td>
<td>The exact time (month/day/year, time) in which the word last occurred.</td>
<td>N/A</td>
</tr>
<tr>
<td>peopleend</td>
<td>The exact time (month/day/year, time) in which the word reached all of the people who were ever going to see/use the word. This time is always equal to or less than enddate.</td>
<td>N/A</td>
</tr>
<tr>
<td>fromtoratio</td>
<td>The average ratio of the number of people in the from over the number of people in the tos and ccs for each email the word is used.</td>
<td>0.907 (0.137)</td>
</tr>
<tr>
<td>stddev</td>
<td>The coefficient of variation of activity of the words over a weekly basis.</td>
<td>3.14 (1.656)</td>
</tr>
<tr>
<td>numActive</td>
<td>The number of weeks the words has been used. This is used to check how consistent the word usage is.</td>
<td>9.95 (9.44)</td>
</tr>
<tr>
<td>maxpeopleadd</td>
<td>The maximum number of people that first seen or used a word in a day.</td>
<td>4.15 (3.98)</td>
</tr>
</tbody>
</table>

Table A-3: The metrics for word categorization

For each of the metrics above, there is a corresponding algorithm to compute them. Those are mentioned in detail below. Because the metrics required knowing all information about the emails a word is in, the method to compute the metrics is done one word at a time. This makes computing the metrics very costly in terms of time, but there is no other more efficient way. However, once the metrics are
computed one time and stored, they are never needed to be computed again unless there is some
programming error that is found in the original computations. The top-level algorithm used to compute
the metrics starts with loading up the candidate set of words. Next, for each word, MyEmailDBHelp
searches through the whole email corpus and finds a list of all the emails that the word occurs in. Once
the list is retrieved (which could take some time for words that occur in lots of emails) each algorithm to
derive their respective metrics is computed. Once the metrics are calculated, they are stored along with
the word in the MySQL server. Using various queries on the server, different combination of filters and
conditions could be computed to come up with categories of words. Eventually, the four final categories
were discovered. Below is a summary of the top-level algorithm to get the different categories from the
candidate set of words.

```
Load words from candidate set

List of words stored in Java objects

For each word in set, get list of emails contain the word

End Goal: Query DB to get categories of words

MySQL DB

Perform algorithms to get metrics for each word

Store Metrics in MySQL
```

Figure A-5: Top-Level Algorithm for Metrics Generation and Selection

A.2.2.1 People-related Metrics: numpeoplefrom, numpeopleto, numpeopleboth, peopleenddate

Before getting any of the people-related metrics, the first piece of information needed is the
mapping between email addresses and names of people in the firm. In most cases, because for all
internally generated firm-based email, the names of the person is substituted for the actual email
addresses, the mappings are self referential. However, there are a few special cases where the people in
the firm use an outside email address or less formal names (like Mike instead of Michael), so a Hashtablename
is needed to map all possible combinations. Previous work has been done on this project by other students
in creating these mappings to account for a variety of combinations, so we used this work and proceeded
on toward calculating the people-related metrics. Once this mapping is created, the program can read in
each email for a word and then can recognized if the email is from and/or to someone inside the firm.
Because the metrics want to only count the number of people rather how many times a person within the firm has seen or used a word, the algorithm hold in memory the current names of all of users of the word as the algorithm sifts through each email through time.

To create the numpeoplefrom metric, the algorithm looks at the froms field in each email record. Because there can only be one person in the froms, the algorithm simply checks the current list of people it has seen in previous emails, and if it has seen the email address before, then it does nothing. If it has not seen the email address before, then the algorithm adds the person to the current list of people who already seen the word. After going through each of the emails for a word, the number of people in the list is the metric numpeoplefrom. The algorithm is written in pseudo-code below for easy reference.

```
lcpf = current list of people who have seen the word (initially blank)
edsw = get_emails_for_word (current word)
for each e:email in <edsw>
begin
    thefromaddr = e.getfroms();
    theperson = mapping.convert(thefromaddr);
    if lcpf.doesnotcontains(theperson)
        lcpf.add(theperson);
    end if
end
numpeoplefrom = getsizeof(lcpf);
```

Creating the numpeopleto metric is a bit trickier than the numpeoplefrom metric because for each email, there can be multiple people who receive the email. So instead of checking for one person for each email, the algorithm checks a list of people. To get the list, we must parse all the data in the tos and ccs field. Below is the pseudo-code for the calculating the numpeopleto metric.

```
lcpi = current list of people who have seen the word (initially blank)
edsw = get_emails_for_word (current word)
for each e:email in <edsw>
begin
    tcl = e.gettos_list();
    tcl = tcl + e.get_ccs_list();
    for each p:person in <tcl>
    begin
        If lcpi.doesnotcontains(p)
            Lcpi.add(p);
        end if
    end
end
numpeopleto = getsizeof(lcpi);
```

For the numpeopleboth metric, the algorithm is to simply combine the ones for numpeoplefrom metric and numpeopleto, but use another list of people that contains both the people who have received
the word and sent the word and the corresponding date and time when they were added to the list. This is so that the next metric, \textit{peopleenddate}, can be calculated. Once the algorithm is finished with all of the emails for a word, the last entry in the list will have the information for \textit{peopleenddate}.

\subsection*{A.2.2.2 Basic Email Information Metrics: \textit{numemail}, \textit{nofroms}, \textit{startdate}, \textit{enddate}, \textit{fromtoratio}}

The next set of metrics is collected a lot more easily than the people-related metrics. This is because they do not require extra memory storage for people who have already seen or used the word. For \textit{numemails}, as the top-level algorithm loops around all possible emails for a word, there is a counter that increments by one every time a new email is sifted through. Although not mentioned before, when the algorithm gets the list of possible emails for a word, it has to filter out the duplicates and the time frames in which the emails were sent. To filter out the duplicates, every time the algorithm loops around and encounters a new email, it checks whether the new email's \textit{froms}, \textit{tos}, \textit{ccs}, and \textit{dateStr} email data matches any previous seen emails exactly. If it has, then the new email is a duplicate and it is ignored in all calculations for any metric. If it has not, then the new email is processed and its \textit{froms}, \textit{tos}, \textit{ccs}, and \textit{dateStr} data is stored in memory. Because there are so many emails, it is not efficient, in terms of memory space and computational time, to store all previous seen email data. So as the algorithm sifts through emails, it dumps the previous seen email data for each new day. This makes sense, because if a new day has occurred, any future emails will not match any previous seen emails. Of course, it is possible to reset the memory store in smaller increments in time, but that would mean more frequent resets and more computational requirements. The time frame filter is just checking the \textit{dateStr} of an email, converting it to a Java \texttt{GregorianCalendar} object, and then making sure the object is within two sets of time frames that are also denoted by \texttt{GregorianCalendar} objects.

For \textit{nofroms}, the algorithm looks at the \textit{from} field in the email data, checks it against the mappings described above in the people-related metrics, and sees if there are any matches. If they are not, then a counter will increment by one as the algorithm sifts through the emails. When the algorithm finishes with all the emails, the counter will have the number of emails that were not from someone within the firm. For the \textit{fromtoratio} metric, the algorithm assumes that each email is sent by only one person. Although this is valid assumption, they are some cases (because of computational error) where there is no \textit{from} information. The algorithm ignores these and then moves on to the next email. In order to count how many people the email was sent to, the algorithm looks at both \textit{tos} and \textit{ccs} fields, delimits them by a semicolon, and then counts how many tokens there are. The number of tokens is the number of strings that are around a semicolon and thus the number of people the email is sent to. Next, the algorithm divides one by this number and adds it to a counter. When all the emails are sifted through, the counter is then divided by \textit{numemails} to get the average \textit{fromtoratio} for all emails associated with the word.
For getting the startdate metric, the algorithm simply records the date of the first email that passes through the duplicate and time frame filters. For the enddate metric, it is a little more complicated. Because the algorithm is sifting each email of a word one email at a time, it does not know how many emails there are total or what the last one is until it sifts through all of them. This is because processing the emails in this manner reduces the memory footprint to one email at a time. If the algorithm did try to get the whole set of emails for each word at once, then for some words, the system will run out of memory. So, as the algorithm goes through each email one by one, there is another variable that holds the previously seen email’s date information. Once the algorithm finishes looping through all the emails, this variable will contain the date of the last processed email. Below is a summary of the algorithm in pseudo-code for all of the basic email information metrics.

```plaintext
numemails = 0;
nofroms = 0;
fromtoratio = 0;
whiche = 1;
edsw = get_emails_for_word (current word) // remembers this filters duplicate and the time frame
for each e: email in <edsw>
begin
    numemails++;
    if whiche = 1
        startdate = getdate(e);
    else
        enddate = getdate(e);
    end if
    whiche++;
    if mappings.does_not_contains(e.getfroms());
        numfroms++;
    end if;
    counttos (e.gettos(), e.getfroms());
    fromtoratio += (1/counttos);
end
fromtoratio /= numemails;
```

A.2.2.3 Activity-related metrics: stddev, numActive, maxpeopleadd

Unlike the previous two sets of metrics, the activity-related metrics requires an addition loop after going through the set of emails for each word. This addition loop is sifting through data implemented as two Hashables that contains the daily and weekly activity numbers for the set of emails for the word. The weekly numbers data is just a collapsed version of the daily data. The exact data structure is described below in figure A-6.

To fill up the two Hashables, the algorithm first goes through the set of emails, looks at each email and its date, confirm that the date is either the same day or the next day from the previous email, and then adds the count to activity number for the appropriate day. This process is also done for the
number of new people that first seen or used the word for each day. Once the algorithm is finished going through all the emails, it is done with creating the Hashtable with daily numbers. The next step is to collapse the daily data into weekly data. This is done simply by doing through each day, adding up the values until a week has passed, and then storing the added value and the week number into the Hashtable. Because maxpeopleadd is a metric based on daily data, this collapsing procedure is done only for the activity numbers.

<table>
<thead>
<tr>
<th>Day</th>
<th># emails</th>
<th>People Added</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week</th>
<th># emails</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure A-6: Activity Hashtable

Once the two Hashtables are created and filled, the algorithm then starts to calculate the stddev metric. The stddev, unlike the name implies, is not the standard deviation of activity but the coefficient of variation of activity indicative by weekly email counts. The first step to compute this is to find the mean of the weekly activity, which is done by adding up all of the values in the weekly activity Hashtable and dividing by 41, the number of total weeks in our data set. Once this mean is calculated, the next step is to compute the standard deviation, which requires the mean, and then divide this number by the mean. This resulting number is the coefficient of variation for the word. In order to calculate the numActive metric, the algorithm searches for any week with any activity and then sums up the total number of these weeks. To find the maxpeopleadd metric, the algorithm goes through each day of the daily activity Hashtable and checks to see if the value is higher than a current assessment of the maximum value (which starts at 0). If the value is higher, a new assessment is assigned with the value and then the algorithm continues on through each day. By the time the algorithm finishes with all the days, the assessment of the maximum value will be correct.

A.2.2.4 Queries for Words Categories

Once all the metrics above are calculated and stored in the database, queries are made to divide the words into different categories. Initially, because we did not know what the categories were or how to define them, we needed to plot histograms of each metric over the total numbers of words. This is so that we could get a feel for the data and see if there are any patterns we can utilize. To create histograms, we queried the database on the metric of interest and then exported the result into CSV files. We then loaded...
the files into Matlab, used the Matlab function `hist`, and then histograms were automatically generated. An example histogram that was not in the body of paper but was used during the research process is shown below. This is a histogram of the metric `nofroms/numemails`.

![Histogram of nofroms/numemails](image)

**Figure A-7: Histogram of nofroms/numemails**

Although there is a general increase in the number of words as `nofroms/numemails` becomes higher, there are three main deviations from this pattern, which are at 0.0, 0.5, and 1.0. For 0.0 and 0.5, there are a lot more words that are used only within the firm or about half the time than expected of the trend. For 1.0, although not shown properly on the graph, about half of all possible words are only being sent from someone outside the firm. This is indicative of spam and that email use is not only used for communication within the firm. Although no interesting categories were made from these results, this is a typical example what kind of information we could get after creating histograms on the metrics.

After analyzing the data set in many different combinations of metrics and constraints, we came up with our four categories of word. The thought process is described more in detail in the body of the paper. Once the four categories were defined, we computed the average and standard deviation statistics for the majority of the metrics. This was done by querying the database on the metric, but also using the SQL functions of `average()` and `stddev()`. The t-tests were calculated through Matlab, after we exported the metrics' data to CSV from MySQL.

**A.2.3 Tree Graphs and related programs**

Although the tree graph analysis that was done during the research process of the paper was not used directly in coming up with the four categories of words, the results of the analysis proved indirectly
useful in coming up with one of the categories, the clique words set. This section describes the programs and algorithms used in performing the tree graph analysis.

A.2.3.1 Data Structure - Description and Creation

As mentioned in the body of the text, we created two types of tree graphs. Type I tree graph represented the nodes as emails while type II represents nodes as people. Although it was argued that Type II was more useful in diffusion analysis, type I has its advantage in tracking down conversations, or the back and forth word usage among people, and may provide different insights from type II. In both cases, the basic class for the data structure is the node.

The node is represented by a Java class called EmailNode. Although the name implies that it represents an email, it can also represent a person as well. Each EmailNode has list of fields, shown in the figure below, that helps track what is represents and what it is connected to. Most of the fields are self-explanatory, except for the children list and parents list. The children and parent list are Java Vectors that simply point to other EmailNodes. Each member of the Vectors represents what other nodes the EmailNode diffuses to (the children) and diffuses from (the parents). These links are useful in computing the tree graphs metrics, like average height, and will be explained in the next section. Because the links are contained within each EmailNode object, there is no need for an overall tree graph object. The list of EmailNodes for a word is sufficient to describe the whole tree graph for the word.

<table>
<thead>
<tr>
<th>EmailNode:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- EmailID</td>
</tr>
<tr>
<td>- From</td>
</tr>
<tr>
<td>- To (includes cc)</td>
</tr>
<tr>
<td>- Date</td>
</tr>
<tr>
<td>- Frequency of Word</td>
</tr>
<tr>
<td>- Children List</td>
</tr>
<tr>
<td>- Parents List</td>
</tr>
</tbody>
</table>

Figure A-8 : Node Data Structure

Once the data structure is defined for the nodes, the next step in tree graph analysis is to create the specific tree graphs for each word. The algorithm to create the Type I tree graph is slightly different from the Type II tree graph, though both follow the same methodology. To create a Type I tree graph, the algorithm first queries the database to retrieve all emails associated with a particular word. It then sorts the emails in order by time (remember, because the date field in the email data set is a string, this is needed) and puts them into a Vector of EmailNodes. Each EmailNode represents one email and all fields are appropriately inputted. Starting for the top of Vector list, the algorithm sets a base pointer to the
earliest EmailNode. It then loops through all the following EmailNodes and sees if any are within the time limit parameter and the sender is one of the base pointer's email receivers. If both conditions are met, then the base pointer's EmailNode adds to its children list the corresponding EmailNode. The corresponding EmailNode adds to its parents list the base pointer's EmailNode. Then the base pointer is moved up to the next email in the list and the loop repeats. This process is continued until the base pointer reaches the next-to-last email. After that, the vector of EmailNodes will be linked so that they will form the Type I tree graph for the word. A flow diagram of this algorithm is shown below for further clarification.

![Flow diagram of the algorithm](image)

**Figure A-9: Tree Graph Data Structure Creation**

To create the type II tree graph data structure, the process is similar to creating the type I. However, the first step is different. Instead creating just one Vector of EmailNodes for all emails, the algorithm creates a second Vector of EmailNodes for the first occurrence of a word by a sender. This second vector will be the tree graph data structure. Next, the algorithm performs the same base pointer comparing and linking mechanism, but instead going through all the emails, it goes the only the first occurrence emails. To create a link, all of the base pointer's EmailNode sender's emails (stored in the first Vector) are checked to create a link with an EmailNode further down the second Vector. If a link is established, then the child will not accept any more parents. This is to ensure that the first person who emails an adopter of the word is the only who gets credit for it. Next, the base pointer moves down the second Vector list to the following EmailNode and the process continues until it reaches the next-to-last EmailNode. Then the type II tree graph data structure is completed.

A.2.3.2 Metric Computations and Analysis

The main purpose of creating the tree graphs was to compute diffusive metrics for each word. Because the tree graphs follow a word's path through the social network and indicate adoption by links,
the graphs had many metrics that were explored to determine diffusiveness. The first set of these concerned the height of the trees in the graph.

The algorithm for determining the height of tree was recursive in nature. For any node that has no children, the height of that node is 0. If the node does have children, then height of it is one plus the height of its tallest children. The height of a tree is the height of the head node. Now because the height function is called upon many times during the course of algorithm, if a height of node is already calculated, there is no need to recalculate it (i.e. go through all the children, children’s children, etc). Therefore, after one call to the height function of node, the value is saved for future reference.

Determining the height of graph this way is simplistic in code, but the trade off is that it takes more computation time. This is because recursive algorithms in Java are not as optimized as other programming languages, like Lisp. The pseudo-code for determining the height of a node is below:

```
function int heightlevel (EmailNode node)
    if (node.getHeight() != -1) { // if height value already stored then return it
        return node.getHeight();
    } else
        if (node.noChild())
            node.setHeight(0);
        return 0;
    else
        Vector<EmailNode> children = node.getChildren();
        int cmax = 0;
        int cl;
        for each EmailNode:kid in <children>
            begin
                cl = heightlevel(kid) + 1; // note recursion call
                if (cl > cmax)
                    cmax = cl;
            end if
        end
        c.setMax(cmax);
        return cmax;
    end if

    Once we have a method for determining the height of a tree, the next step is to come up with various metrics concerning this characteristic. The metric mentioned in the body of the paper, average height, is just the average heightlevel() of all the nodes in a tree graph with no parents, but with children. Two other metrics calculated but not used is the maximum height and minimum height of the trees in a graph. Because the average height metric penalizes words which have one-node trees even though they also have very long trees, the maximum height metric was created as a substitute. The minimum height
metric allows us to check whether a given word has one-node trees. If it does not, the word is always used in diffusive processes.

In addition to the height of trees, the number of emails and people involved in a tree is also indicative of diffuseness. It weighs the height of tree together with the width of the tree to form possibly a more comprehensive measure. To calculate these metrics, the first step is split a word’s tree graph into components trees. Although it is possible to determine how many trees there are in a graph and which nodes are the head of those trees, it is not immediately possible to determine which nodes belong to which trees. This is because a node only links with other nodes one step removed. The first way we thought to do this was to, again, implement a recursive algorithm that starts with the head node of a tree, searches though all of its children, then the children’s children and so on, and come up with a list of nodes in its tree structure. Although the code for this would be short and simple, the time is takes to get a list of nodes is tremendous long. This is because (only in the type I graph) there can be multiple parents for each node. For each additional parent of a child, the algorithm will over check the child and its children as a member of a tree. For example, examine the figure below.

![Figure A-10: Bad Scenario for Recursion](image)

This could be a typical tree for a word with multiple parents. Now, because the dashed node has multiple parents with multiple parents, a recursive algorithm going through each node’s children will check on the dashed node four times. This may not sound that bad, but if the dash node’s linage was long, then the algorithm will go through this linage four times as well. If there were further multiple parents down this tree, the checks for the nodes will grow exponentially for each level down. With heights of some trees (for type I) sometimes being 12-16, the amount of checks would be enormous. Therefore, a new algorithm was devised that was non-recursive and quicker on average, especially when there are multiple parents.
The new method for figuring out the sets of nodes belonging to each tree in a graph was inspired by the spread of infection among a network. First, the node population was divided up into infected and non-infect nodes. Initially, the head of a tree was infected and the rest of the nodes in the graph were healthy. Then the algorithm loops through all the healthy nodes and checks whether they have immediate contact with an infected node. This means that if any of parents of the healthy nodes were infected, then the healthy node becomes infected. Once a node becomes infected, it removed from the healthy node list and onto the infected list. This spreading of infection process is continued for multiple times, the minimum being about the height of the tree to guarantee full infection. After this is over, the infected list of nodes is the list of nodes associated together in one tree. A pictorial diagram of the process is shown below.

![Figure A-11: Infection based Algorithm](image)

Notice after each cycle, a new height level gets infected. In some cases, if the time parameter limit is long enough, some nodes will link to other nodes many levels down. This means that using the infection based algorithm will process nodes on many different height levels at the same time. This as well as eliminating the redundant checks for multiple parents makes the infection based algorithm much faster than a simple recursion one. The downside to the approach is the need to create variables to store which nodes are infected. However, this is a simple boolean variable that costs very small amounts of memory, so the trade-off is worth it.

With the set of nodes associated with each tree in a graph, the metrics for the number of emails and people involved for each tree is simple to compute. The number of emails is just the number of nodes per tree (for the type I tree graph). To compute the people metric, the algorithm has to sift through the list of nodes, check the email addresses in the from, to, and cc fields to see if they are within firm, and then add up the total number, making sure not to double count. With both these metrics, we can create
corollary metrics, which are the maximum and minimum number of emails and people per tree. The uses of these metrics correspond with the equivalent height metrics explained above.

The next set of tree graph metrics is time-related. They are the average length of time between the nodes that are connected with links, and the maximum and minimum of these times. The algorithm to compute these is to go through each node that is associated with a word. For every node, go through each of its children, calculate the difference in time in the dates, add all the times up and divide by the number of links. To figure out the maximum and minimum metrics, the algorithm just goes through all the differences of time and finds the corresponding values. One problem with these time metrics is that for the type I tree graphs, there can be multiple parents with a wide range of times associated with adoption. As the time parameter limit gets longer, these metrics will turn out to get longer as well. This is because all the sources of adoption, from the farthest way (in terms of time) to the closest one, are included in the calculation. For the type II tree graph, this is not a problem.

The last set of tree graph metrics are the number of stubs and branches. Stubs are just nodes with no parents or children. Branches are nodes that have at least a parent or child. The branches and stubs metrics can be used as measures of diffusiveness as a percentage to the total number of nodes. The higher percentage of branches, the more a word is used in diffusive processes. And vice versa, the higher percentage of stubs, the more a word is non-diffusive. A review of all the tree graph metrics is in the table below.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>average/min/max height</td>
<td>A height of a tree is longest count of the consecutive links between nodes. The average/min/max is over all trees in a given graph.</td>
</tr>
<tr>
<td>average/min/max people</td>
<td>The number of people in a tree is how many people that are within the firm and either senders or receivers of email in each tree. The average/min/max is over all trees in a given graph.</td>
</tr>
<tr>
<td>average/min/max emails</td>
<td>The number of emails in a tree is how many nodes there are in a tree. This metric is only valid for type I tree graph. The average/min/max is over all trees in a given graph.</td>
</tr>
<tr>
<td>average/min/max diffusion time</td>
<td>The diffusion time is the difference between a parent node’s timestamp and node’s in question time stamp. The average/min/max is over all links in a given tree graph.</td>
</tr>
<tr>
<td>branches</td>
<td>The number of nodes that have at least one child or a parent (can be both) in a graph.</td>
</tr>
<tr>
<td>stubs</td>
<td>The number of nodes that have no children and no parents in a graph.</td>
</tr>
</tbody>
</table>

Table A-4: Tree Graph Metrics

A.2.3.3 Visualizing the Graphs
One of the best ways to analyze and understand diffusion through the tree graphs is to visualize them. The type II tree graph is actually a result of looking at type I tree graphs and noticing how convoluted the graphs look, even though there are only few people who are actually using the word. An example of the difference between a type I and II tree graph is shown below. Both represent the same word, but at very different perspectives.

![Type I (left) and II (right) Tree Graph Difference](image)

The graphs were rendered using yEd Graph Editor, but any graphing package would have been fine. The hard part was to translate the tree graph data structure defined as Java objects into a formatted XML file so that yEd or any other graphing package could read and display the results. The XML file format we choose to use is the GraphML format located at http://graphml.graphdrawing.org/. This format has two parts, the list of nodes and then the list of edges of the nodes. Because the data structure we used in Java did not separate the two types of the data, we had to create another program to do so. This program takes in a word and outputs out the resulting GraphML file. In between, it creates the list of
EmailNodes, described above, and then sifts through the list to separate the nodes from the links. To create list of nodes, the program goes through the Vector list of EmailNodes, and then write out in the output textfile, the corresponding text, like the one below, for each node. Note that each node has to have a unique id so that the set of edges can be defined.

```
<node id="0">
  <data key="d0">
    <y:ShapeNode>
    </y:ShapeNode>
  </data>
</node>

<node id="1">
  <data key="d0">
    <y:ShapeNode>
      <y:NodeLabel> B : Wed Mar 12 14:53:01 EST 2003</y:NodeLabel>
    </y:ShapeNode>
  </data>
</node>

<edge id="0" source="0" target="1">
  <data key="edgeinfo">diffusion link</data>
</edge>
```

As the program runs through each node, it also creates a list of pairs of a node and all of its children. Then the program runs down through the list of pairs and outputs the edge information, like one shown above, for each pair. The edges have to follow the exact name ids of all the nodes in which they are linking. Once the program writes out all the edges, the XML file is complete and then yEd can load it up to display the results.

**A.2.4 Social Network-related data generation**

Although this thesis does not cover any analysis on the relationship between the categories of words outlined and the social network characteristics of the firm, the research behind the process produced a variety of data sets that may help out in that endeavor. The first data set, consisting of tables, is the diffusion characteristics of the word categories, described in the body of the text in section *Extensions*. The next data set is the social network matrices: strength of ties, friends in common, and shortest path. After these matrices created, then equivalent tables that correspond with word categories’ diffusion tables were created.

**A.2.4.1 Word Categories’ Diffusion Table**
For each of the categories of words, three different types of diffusion table were created to provide various levels of detail for analysis. The base level consists of 1 or 0 markers that indicate which persons within the firm are involved with the same word. The next level shows the ordinal ordering in which the word was passed around in the network. The last level shows the exact time in which the word was first seen or used by each person. Each row of these table types consist of one word and its corresponding diffusion information. Also, although the tables will show who and how they received a particular word, the algorithm to create these tables does not distinguish between words. This means an input list of commonly used words will be approached in the same manner as event words or any other category. It is up to the user of the algorithm that creates these diffusion tables to input the correct set of words. An example of the format of the tables is shown in Table 3.

In order to create these tables, the first piece of information needed is the mapping of people and their email addresses within the firm. This mapping is already described in section A.2.2.1 People-related metrics. Next, because the tables are eventually going to be used to compare with a social network matrix, the order in which the people are labeled across the columns has to match exactly with order of the social network matrix. This is so that dyadic analysis can be performed easily and automatically. Once the proper ordering of people is established, the next part in the algorithm consists of outputting the appropriate markers for each table type. For the base level 1 or 0 table, this is simply indicating all the people who have seen or used the word. However, for the ordinal ordering table, the algorithm is a bit trickier. This is because whether a person is a sender or receiver of the word matters in determining the order in which the word was used. The algorithm first sorts the list of emails associated with a word in order. Then it reads the from and to information. The person who sends the word is always before the people who receive the word. On top of this ordering, though, the time in which a person first sees a word is more important. For example, although Person B is the tenth person who has sent an email with a word, if he is the first person to receive the word, then he is the first user. For the time-stamped diffusion table, ordering of people is different than the cardinal table because it does not take into account the from/to distinction. All the people on the second email of the word are considered to be the second group of people who have used the word.

For all three types of diffusion tables, the from/to distinction is used to determine the first person of the word. This means that there can be multiple first persons of a word. For example, if a person outside the firm first sends an email with the word to multiple people within the firm, the set of receivers are the first persons of the word. To make future social network analysis easier, if a word has multiple first persons, then the word is repeated on consecutive number of rows in the table, where each row has only one of the first persons. Since each row has only one first person, each row can be compared to a corresponding row in another table with the first person and his dyadic social network measures.
A.2.4.2 Social Network Matrices and Tables

Social network matrices are the preferred (and highly concise) form in which social networks are represented (other forms include a list of nodes and edges, like the graph format above). Each column and row represents a person in the social network and the cells contain the measure in which the people in the corresponding row and column are related. All possible combinations of relationships between two people in the social network are represented. The base social network matrix is the strength of ties (SOT) matrix. In our social network, the ties are the number of emails sent to and from two people. The other two social network matrices, friends in common (FIC) and shortest path (SP), are derived from the SOT matrix to compute their relationship measures.

A SOT matrix for our email data set was already computed in Excel with VBA macros in previous work. However, it had two errors: the inclusion of duplicate emails and a scripting bug within a macro. Duplicates are emails that have the exact same from, to, cc, date, and subject information as other emails. Because it highly likely that these emails are just duplicates of other emails, using these emails toward the SOT between two people is not correct. The scripting bug consisted of a misinterpretation of the use of a function within Excel. The function was thought to have returned the first emails in which a particular person sent or received. However, the function skipped the first one and returned the second one. Finding these sources of error took a very long time, but the best way to approach solving this
problem was plotting the difference between what we thought was a correct SOT matrix and what the previous computed one was. Figure A-13 above shows the difference of the correct one minus the previous one. The abundance of negatives numbers indicates that the correct one did not include many of the emails the previous one had. This led to the discovery of duplicate emails or “duplicates”. Although originally the amount of duplicates was thought be inconsequential (we assumed the data to be very robust and clean), it turned out duplicates were more than 15% of the emails recorded. This led to important differences between social matrices with duplicates and without duplicates. The scripting bug error also accounted for some differences between the correct SOT matrix and the previous one. We figured out this error by running the macros one line at a time and to see how it counted emails between people.

After fixing the errors from the previous SOT matrix, we outputted the final matrix data in CSV file format, readable by Excel. An example template of this matrix is provided below. The person in a given column sends x emails to the person in a given row. Specifically, person A sent 5 emails to person B.

<table>
<thead>
<tr>
<th>SOT Matrix</th>
<th>Person A</th>
<th>Person B</th>
<th>Person C</th>
<th>Person D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person A</td>
<td>13</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Person B</td>
<td>5</td>
<td>16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Person C</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Person D</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>24</td>
</tr>
</tbody>
</table>

Table A-5: Example Social Network Matrix

To create the other matrices, FIC and SP, requires the SOT matrix and defining what determines a connection from one person to another. There are potentially three ways to define what an immediate connection is. For example, using the example data set above and looking at the Person A column, it looks like Person A has a connection with three people, himself, Person B, and Person D. Or looking at the Person A row, it looks like he may only have two connections, himself and person C. Or Person A has connections for all four people shown in the network. The difference between these definitions is how one counts emails that are only sent from a person, received by a person, or either one. In addition to these three types of connections, connections can also be defined on a threshold of minimum number of emails. For example, if the threshold is set to 10 emails, then Person A does not have any connection to anyone else except for himself. Three different thresholds were used in the outputted data set, 0, 5, and 10. Because of the variety of different ways to define a connection, both the FIC and SP matrix has 9 different versions, corresponding to all combinations of types and thresholds.
The algorithm for computing the FIC matrix is to go through each two pair relationship in the network, get two lists of connections for each person in the pair, and then count how many people overlap in those lists. Because the pair ordering does not matter, the FIC matrix is symmetric about the diagonal. The format of the FIC matrix is the same as the SOT matrix, except that the numbers represent shared connections, instead of number of emails.

The algorithm for computing the SP matrix is little more complex than the FIC. SP is the minimum number of connections steps from one person to another. For the example below, the shortest path from A to B consists of three steps. Although it is possible to go to B from A by four steps, SP is the minimum number of steps. Also, unlike FIC, the pair order does matter in the calculation. As stated before, the shortest path from A to B is three, but the path from B to A is only one. Here in the graph, the arrows represent the connection from one person to another. Since connections can be defined with many different ways, for each set of definitions, the arrows and directions may be pointed in different directions.

Although using a graph with nodes and edges makes SP easy to comprehend visually, the underlying data is not in that format, but instead it is in the SOT matrix format. Therefore, the algorithm has to be a little more sophisticated in dealing with the data. First, the algorithm scans the row (or column or both) of a person, which we will call Person X, in the SOT matrix. Then, it writes down a 1 in the SP matrix for every person with a value in relation to Person X that is over the threshold that defines a connection. Next, it finds all the connections of all the people now marked with 1 in the SOT. This list of people is then marked with 2 in the row of Person X in the SP matrix. The algorithm continues on like this step-by-step until all persons are accounted for or it reaches a maximum number of steps, which is 10. Then it moves onto the next person in the SP matrix and finds its SP with everyone else. This process is looped around until it is finished with everyone in the SP matrix. Pseudo-code of the algorithm is written below.

With the SOT, FIC, and SP matrices computed, the final social network-related data set that was created was the equivalent social network tables corresponding to the word categories' diffusion tables. These tables consist of rows that come from the social networks matrices of the first person in the word
categories' diffusion tables. This allows a row by row comparison of the first person using the word and the order in which other members of the firm used it versus the first person and his dyadic relationship with other members of firms in terms of SOT, FIC, and SP.
Appendix B: Other Supporting Graphs and Statistics

B.1 – Histogram of Frequencies for each word categories

![Histograms of Frequencies for each word categories](image)

Figure B-1: Histogram of Frequencies for each word categories

All graphs are clear distinct from each other. Of the four graphs, the two most similar to each other are the *common* – *specialized* and *event* words. However, both are still pretty distinct from each, judging by the averages and the way the *common* – *specialized* words is both more widely distributed and centered to the right. One thing to note is that the *common* – *everyday* words’ x axis is on order of $10^4$. 
B.2 – Histogram of People for each word categories

All four of graphs are, again, clearly distinct from each other. However, considering the common – specialized words and event words have specific # people cutoffs, it is not surprising they are different from the other two. What is interesting is the difference between those two word sets. One can see a semi-normal distribution for the common – specialized words centered around 50. This reminiscent of the commonly used words distribution and is hypothesize that individually generated common words follow this pattern, which falls in line with what common – specialized words are.
### B.3 – T-tests comparison results

<table>
<thead>
<tr>
<th><strong>Frequency</strong></th>
<th>Common Words – Everyday</th>
<th>Common Words – Specialized</th>
<th>Event Words</th>
<th>Clique Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Words – Everyday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Common Words – Specialized</td>
<td>[11069.84, 14610.10]; 18.758**</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event Words</td>
<td>[11185.55, 14725.83]; 18.874</td>
<td>[96.28, 135.14]; 7.654</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Clique Words</td>
<td>[11404.12, 14944.29]; 19.247</td>
<td>[320.75, 347.72]; 63.893</td>
<td>[204.40, 232.64]; 27.691</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Num People</strong></th>
<th>Common Words – Everyday</th>
<th>Common Words – Specialized</th>
<th>Event Words</th>
<th>Clique Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Words – Everyday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Common Words – Specialized</td>
<td>[21.44, 22.28]; 134.247</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event Words</td>
<td>[29.74, 30.46]; 170.066</td>
<td>[7.79, 8.68]; 42.3155</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Clique Words</td>
<td>[63.70, 64.17]; 701.038</td>
<td>[41.72, 42.43]; 308.683</td>
<td>[33.56, 34.11]; 213.280</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Coefficient of variation of activity:</strong></th>
<th>Common Words – Everyday</th>
<th>Common Words – Specialized</th>
<th>Event Words</th>
<th>Clique Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Words – Everyday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Common Words – Specialized</td>
<td>[.31, .33]; 86.39</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event Words</td>
<td>[.97, 1.01]; 92.920</td>
<td>[0.65, 0.69]; 71.448</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Clique Words</td>
<td>[3.58, 3.69]; 162.349</td>
<td>[3.26, 3.37]; 149.327</td>
<td>[2.58, 2.70]; 90.531</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Emails/Person</strong></th>
<th>Common Words – Everyday</th>
<th>Common Words – Specialized</th>
<th>Event Words</th>
<th>Clique Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Words – Everyday</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Common Words – Specialized</td>
<td>[163.02, 213.83]; 19.1789</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event Words</td>
<td>[164.53, 215.35]; 19.2263</td>
<td>[.0115, .9954] 2.1029 95% Significant</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Clique Words</td>
<td>[163.16, 213.99]; 19.189</td>
<td>Failed, p=56.7% 1.105</td>
<td>[.68, 2.06]; 1.105</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table B-1: T-tests comparison results

*All intervals represent the absolute difference between the two metrics. They are at the 99% confidence level, unless otherwise stated.

**Each cell has the t-statistic on the 2nd line and is significant at the 99% confidence level, unless otherwise stated.
References


