DETECTION AND EXTRACTION OF DISCUSSION THREADS IN A CHAT SESSION

A MASTER'S THESIS REPORT

BY

GANESH SHANKARAN

2004

Dedicated to the memory of my grandmother

ABSTRACT

Textual chat services such as instant messengers and chat servers have emerged as a popular mode of communication among people for both business and recreational uses. In recent times, there has been a high incidence of participating speakers indulging in posting or typing messages inappropriate to the chat group. Due to the high volume of messages involved, it is not feasible to offer real-time solutions to analyze each chat message. However, the problem is greatly simplified if we can group together only those utterances from a chat session that comprise a thread. Once similar utterances have been grouped together, these individual discussion threads could then be used to perform several operations such as:

1. Build a profile of speakers based on the thread in which they participate.
2. Identify the topics being discussed in the chatroom.
3. Retrieve only those messages from a chat session which are related to a particular thread.

In this research, I present an automatic method to identify discussion threads. The approach is based on clustering together utterances that are similar to each other. Since the chat utterances are stored and archived as individual files, I treat them as documents of extremely small size. Stopword removal is applied to each of these utterances to eliminate words that do not convey any particular information. This in turn leads to a further reduction in the size of documents being clustered; in most cases, eliminating the document from consideration. In order to compensate for this loss, we apply post-processing techniques to group these utterances with identified threads that surround them. The results from a series of experiments are presented

demonstrating that postprocessing can raise the accuracy of thread detection from 0.86 to 1.

ACKNOWLEDGEMENTS

This master's thesis would not be what it is without the support and assistance of several people. First and foremost, I express my sincere gratitude to Dr. Susan Gauch, my thesis advisor and committee chair, for her understanding, experience, support and patience in what has seemed an eternity to accomplish my work. I have greatly benefited from the numerous hours of consultation she has willingly offered, in guiding me through this thesis. I am also deeply indebted to Dr. Daphne Fautin from the Division of Biological Sciences for supporting me through her research grant. I thank Dr. Jerry James and Dr. Jerzy Grzymala-Busse for serving as my Thesis committee members. Thanks to Dr. George Karypis, from the University of Minnesota, Twin Cities, for having contributed the CLUTO clustering package to the research community. Thanks also to Sriram for his help in reviewing the draft, and to the entire ChatTrack team, both past and present, for their contribution in developing the framework for archiving and storing data, and generating the input required for my work. Rajan, Jason, Eera, and Jim – great work guys!

And lastly, but certainly not in the least, I thank my parents for their support and encouragement without which none of this would have happened.

TABLE OF CONTENTS

LIST OF FIGURES

1. Introduction

Instant messaging (IM) and chat services have become an important means of communication among people around the globe, providing an alternative to telephone and email communications. The number of people using instant messaging applications and chat services has increased steadily over the last decade. The total time spent using instant-messaging applications at home in the U.S. increased 48%, from 9.2 billion minutes in September 2000 to 13.6 billion minutes in September 2001. At the same time, the number of unique users of instant-messaging applications at home increased 28%, from 42.0 million in September 2000 to 53.8 million in September 2001[2].

Coupled with the rise in use of these applications is the rise in chat data. There are few techniques available to store, explore, or analyze this flood of information. In this paper, we present an approach to extract discussion threads from chat session, thus reducing the analysis space to a few clusters as opposed to the thousands of utterances that are typically seen in a single Internet chat session.

1.1. Background and Motivation

With increased usage of chat rooms and instant messengers as a mode of communication comes an increased risk that the speakers are exposed to inappropriate contact, especially young children. Pedophiles could be posing as children on chat groups and chat rooms meant for children, targeting impressionable youngsters with inappropriate messages. The United States Internet Crime Task Force, Inc. (www.usict.org) projects that this year alone, 1 out of 5 children will receive an online approach by a predator. Last year, 19.5 million children went online and the USICT reports that 65% were solicited in chat rooms and 10% of children were asked to meet someplace. 77% of those children were under 14 years of age[1]. There are also security concerns that criminals and/or terrorists are using chatrooms as places to meet and disseminate information, and their conversations are camouflaged by the flood of other chat data.

Instant messaging has also become a popular choice of communication in corporate environments as well. Employees and clients alike are using IM to quickly communicate important office details, technical plans, and even meetings through chat sessions. The total time that people at work used publicly available IM products from AOL Time Warner's America Online (NYSE:AOL), Microsoft's (NASDAQ:MSFT) and Yahoo (NASDAQ:YHOO) jumped 110%, from 2.3 billion minutes in September 2000 to 4.9 billion in September 2001. However, unlike traditional meetings, there is often no record of these electronic discussions.

Chat rooms are also used to disseminate and share knowledge. At the University of Kansas, students enrolling in the following two courses, EECS 745 and

EECS 888 (at the time of this writing) use chat rooms for discussions, replacing the traditional discussion sessions involving the physical presence of the instructor and students. Programmers at the Mozilla development center also use IRC clients to keep others informed about code changes, and version control.

With such a vast target audience, it is imperative that methods be developed to archive, summarize, and analyze information from chat sessions. For example, a chat session involving the participation of 20 people could easily generate more than a 1000 utterances over a period of couple of hours. During this time, several topics may be discussed by several people, and these again, may not be all contiguous. Consider for example, a particular group of people discussing about enriched plutonium in a programming languages group. An occasional statement interspersed with utterances on programming languages may seem innocuous, perhaps erroneous, but it is difficult to keep track of the pattern of this suspicious discussion. However, if threads can be extracted, off-topic discussions can be found. In addition, summarization and analysis of single-topic threads, rather than diverse, unrelated utterances is likely to be more efficient and more effective.

Most systems aimed at analyzing chat messages do not perform a thorough analysis; rather they rely on filtering messages based on predefined keywords. In this work, we have aimed to create a more comprehensive analysis of chat sessions. The goal is to identify threads by grouping together similar utterances. A significant number of these utterances do not convey any particular information, for example, “Hi,” “Bye,” “brb,” etc. We therefore need a more intelligent solution to retrieve only those utterances which convey information. These clusters of documents can then be

used to perform several other tasks, such as identifying topics being discussed, profiling speakers based on their utterances/cluster of utterances, or retrieving only those utterances pertaining to a group/topic.

1.2. Related Work

My research investigates whether similar approaches could be applied to detect threads in a chat session based on the clustering of chat utterances, wherein each utterance may be considered as a very short document. Document clustering has been, and still is, an area of tremendous research interest in information retrieval. Applications of document clustering range from organizing retrieval results in web search engines [5], to topic detection and tracking in news stories [17][29][21][8], and event tracking [6][18].

1.2.1. Document Clustering

Document clustering has been investigated as a means of improving the performance of information retrieval systems by pre-clustering the entire corpus [3]. However, clustering has also been investigated as a post-retrieval results browsing technique [6]. My work follows the former paradigm.

There are numerous document clustering algorithms that appear in the literature, of which *Agglomerative Hierarchical Clustering* (AHC) algorithms are commonly used because they provide robust results across many applications and data sets. These algorithms are typically slow when applied to large document collections. *Single-link* and *group-average* methods typically take $O(n^2)$ time, while *complete-link* methods that recompute similarities to each item in the clustering superscript typically take $O(n^3)$ time, where n is the number of input documents[9].

As shown by experiments, these algorithms are too slow to meet the speed requirement for one thousand documents [5].

Linear time clustering algorithms are often the best choices in terms of clustering performance. These include the K-Means algorithm - *O(nkT)* time complexity where *k* is the number of desired clusters and *T* is the number of iterations [16] and the Single-Pass method - *O(nK)* were *K* is the number of clusters created. One advantage of the K-Means algorithm is that, unlike AHC algorithms, it can produce overlapping clusters. Its chief disadvantage is that it is known to be most effective when the desired clusters are approximately spherical with respect to the similarity measure used. There is no reason to believe that documents (under the standard representation as weighted word vectors and some form of normalized dot-product similarity measure) should fall into approximately spherical clusters. The Single-Pass method also suffers from this disadvantage, as well as from being order dependant and from having a tendency to produce large clusters [10]. It is, however, the most popular incremental clustering algorithm, particularly in the event detection domain [24].

1.2.1.1. Clustering Techniques

Hierarchical clustering is often portrayed as the better quality clustering approach, but its applicability is limited because of its quadratic time complexity. In contrast, K-means and its variants have a time complexity which is linear in the number of documents, but they are thought to produce inferior clusters [26]. Sometimes K-means and agglomerative hierarchical approaches are combined to form a new class

of clustering algorithms called *constrained agglomerative algorithms* that combine features from both partitional and agglomerative approaches. This allows them to reduce the early-stage errors made by agglomerative methods and hence improve the quality of clustering solutions [32].

In this section, I provide a brief description of hierarchical, partitional, and hybrid algorithms.

1.2.1.1.1. Hierarchical Clustering Techniques

Hierarchical techniques produce a nested sequence of partitions with a single, all inclusive, cluster at the top and singleton clusters of individual points at the bottom. Each

intermediate level can be viewed as combining two clusters from the next lower level (or

splitting a cluster from the next higher level). The result of a hierarchical clustering algorithm can be graphically displayed as tree, called a dendogram. This tree graphically displays the merging process and the intermediate clusters. Hierarchical clustering techniques fall into two broad classes:

a) **Agglomerative**: Start with the points as individual clusters and, at each step, merge the most similar or closest pair of clusters. This requires a definition of cluster similarity or distance.

b) **Divisive**: Start with one, all-inclusive cluster and, at each step, split a cluster until only

singleton clusters of individual points remain. In this case, we need to decide, at each step, which cluster to split and how to perform the split.

Agglomerative techniques are much more commonly used. A Simple Agglomerative Clustering Algorithm is described below:

1. Compute the similarity between all pairs of clusters, i.e., calculate a similarity matrix

whose ij^{th} entry gives the similarity between the i^{th} and j^{th} clusters.

2. Merge the most similar (closest) two clusters.

3. Update the similarity matrix to reflect the pairwise similarity between the new cluster and the original clusters.

4. Repeat steps 2 and 3 until only a single cluster remains.

1.2.1.1.2. Partitional Clustering Techniques

In contrast to hierarchical techniques, partitional clustering techniques create a one-level

(un-nested) partitioning of the data points. If *K* is the desired number of clusters, then partitional approaches typically find all *K* clusters at once. Contrast this with traditional hierarchical schemes that bisect a cluster to get two clusters or merge two clusters to get one. Of course, a hierarchical approach can be used to generate a flat partition of *K* clusters, and likewise, the repeated application of a partitional scheme can provide a hierarchical clustering.

There are a number of partitional techniques, but because the K-means algorithm is widely used in document clustering, we will focus on it. K-means is based on the premise that a cluster can be well-represented by its center point, or centroid, the mean or median of a group of points. Note that a centroid almost never corresponds to an actual data point, but rather a virtual location. One limitation of the K-means approach is that it requires an initial value, *k*, for the number of clusters to produce. For some applications, this is known in advance, but for others it is unavailable and a variety of values for *k*, must be evaluated.

The basic K-means clustering technique is presented below.

Basic K-means Algorithm for finding *K* clusters.

1. Select *K* points as the initial centroids.
2. Assign all points to the closest centroid.
3. Recompute the centroid of each cluster.
4. Repeat steps 2 and 3 until the centroids remain unchanged [26].

1.2.1.1.3. Constrained Agglomerative Clustering Techniques

Constrained agglomerative algorithms generate the clustering solution by using an agglomerative algorithm to build a hierarchical subtree for each partitional cluster and then agglomerate these clusters to build the final hierarchical tree. Experimental evaluation by Zhao and Karypis [31][32] has shown that these methods consistently lead to better solutions than agglomerative methods alone and for many cases they

outperform partitional methods, as well. To understand these improvements, they studied the impact that the constraints have on the quality of the neighborhood of each document and found that the constraints led to purer neighborhoods since they identify the good subspaces for the various classes.

One of the advantages of partitional clustering algorithms is that they use information about the entire collection of documents when they partition the dataset into a certain number of clusters. On the other hand, the clustering decisions made by agglomerative algorithms are local in nature. This local nature has both its advantages as well as its disadvantages. The advantage is that it is easy for them to group together documents that form small and reasonably cohesive clusters, a task in which partitional algorithms may fail since they may split such documents across cluster boundaries early during the partitional clustering process, especially when clustering large collections. However, a disadvantage to agglomerative algorithms, is that, if the documents are not part of particularly cohesive groups, the initial merging decisions may contain some errors that tend to be magnified as the agglomeration progresses. This is especially true for the cases in which there are a large number of equally good merging alternatives for each cluster [31][32].

One way to eliminate this type of error is to use a partitional clustering algorithm to constrain the space over which agglomeration decisions are made by only allowing each document to merge with other documents that are part of the same partitionally discovered cluster. In this approach, a partitional clustering algorithm is used to compute k clusters. Then, each of these clusters, referred as *constraint clusters*, is treated as a separate collection and an agglomerative algorithm is used to

build a tree for each one of them. Finally, the *k* different trees are combined into a single tree by merging them using an agglomerative algorithm that treats the documents of each subtree as a cluster that has already been formed during agglomeration. The advantage of this approach is that it is able to benefit from the global *view* of the collection used by partitional algorithms and the local *view* used by agglomerative algorithms. An additional advantage is that the computational complexity of constrained clustering is $O(k((n/k)2\log(n/k))+k2\log k)$, where *k* is the number of constraint clusters. If *k* is reasonably large, *e.g.*, *k* equals $\sqrt{n}$, the original complexity of $O(n2\log n)$ for agglomerative algorithms is reduced to $O(n2/3\log n)$ [31][32].

1.2.2. Topic/Event Detection and Tracking

Monitoring chat topics over time is related to Event Detection and Tracking efforts. Event Tracking is the task of monitoring a stream of news stories to find those that discuss the same event as the one covered in a few sample stories. Since it is difficult, many systems rely on manually supplied examples of new events. For example, having read one or two stories about a bombing, a user might tag those stories and ask that the system notify him or her when new stories on the same event are broadcast [6].

The Event Detection and Tracking problems are part of a broader initiative called Topic Detection and Tracking (TDT). The domain of TDT's interest is all broadcast news – i.e., written and spoken news stories in multiple languages. As such the problem is substantially broad encompassing automatic speech-to-text efforts,

finding the boundaries between news stories for archival and presentation purposes, locating new events within the stream, tracking located events, and doing all of that in a multi-lingual environment with degraded information. As its name implies, TDT is also ultimately concerned with ways of organizing information that are broader than "events" [18].

1.2.2.1. The Tasks

The input to the Topic Detection and Tracking process is a stream of stories. This stream may or may not be pre-segmented into stories, and the events may or may not be known to the system (i.e., the system may or may not be trained to recognize specific events). This leads to the definition of three main technical tasks to be addressed in the TDT study. These are namely, the segmentation of a news source into stories, the detection of unknown events, and the tracking of known events. Following is a brief description of the different tasks [17].

1.2.2.1.1. The Segmentation Task

The segmentation task is defined as the task of segmenting a continuous stream of text (including transcribed speech) into its constituent stories. The segmentation task tries to, for all stories in the corpus, correctly locate the boundaries between adjacent stories.

Segmentation must be done before further processing is possible, it is therefore an "enabling" technology for other applications, such as tracking and new event

detection [17]. We can view our research as developing techniques to segment chat sessions into stories. The task is more difficult because the granularity of the stream contents is much finer than in news transcripts.

1.2.2.1.2. The Detection Task

The detection task is characterized by the lack of knowledge of the event to be detected. In such a case, we may wish to retrospectively process a corpus of stories to identify the events discussed therein, or we may wish to identify new events as they occur, based on a stream of stories.

1.2.2.1.3. Retrospective Event Detection

The retrospective detection task is defined as the task of identifying all of the events in a corpus of stories. Events are defined by their association with stories, and therefore the task is to group the stories in the study corpus into clusters in which each cluster represents an event and the stories in the cluster discuss the event. Our work can also be viewed as retrospective event detection in that the segmentation produces a group of related utterances, each of which is a thread or event, from an archived chat session.

1.2.2.1.4. On-line New Event Detection

The online new event detection task is defined to be the task of identifying new events in a stream of stories. Each story is processed in sequence, and a decision is made whether or not a new event is discussed in the story, after processing the story

but before processing any subsequent stories. A decision is made after each story is processed. It requires two main components: event detection and the ability to determine whether or not the event detected is sufficiently dissimilar to previous events to be considered "new".

1.2.2.1.5. The Tracking task

The tracking task is defined as the task of associating incoming stories with events known to the system. An event is defined "known" by its association with stories that discuss the event. Thus, each target event is defined by a list of stories that discuss it. In the tracking task, a target event is given and each successive story must be classified as to whether or not it discusses the target event.

1.2.2.2. A Comparison of the Design Approaches

Topic Detection and Tracking has been investigated by many, but we will report on three of the most active group – Carnegie Mellon University (CMU), University of Massachusetts at Amherst (UMass), and Dragon Systems. A brief description of the approaches adopted by each of the investigators follows.

1.2.2.2.1. CMU Approach

The CMU approach to retrospective event detection is to cluster stories in a bottom-up fashion based on their lexical similarity and proximity in time. The CMU approach to on-line detection combines lexical similarity (or distance) with a declining influence look-back window of '*k*' days when judging the current story, and

determine NEW or OLD based on how distant of the current story from the closest story in the '*k*' days window. [17]

1.2.2.2.2. UMass Approach

UMass has developed two largely complementary segmentation methods. The first method makes use of the technique of local context analysis (LCA). The second segmentation method uses a Hidden Markov Model (HMM) to model "marker words," or words which predict a topic change.

The UMass approach to event detection is similar to CMU's in that it uses a variant of single-link clustering to build groups of related stories to represent events. New stories are compared to the groups of older stories. The matching threshold is adjusted over time in recognition that an event is less likely to be reported as time passes. UMass's retrospective detection method focuses on rapid changes by monitoring sudden changes in term distribution over time. [17]

1.2.2.2.3. Dragon Systems Approach

The Dragon Systems approach is based on observations of term frequencies using adaptive language models developed for speech recognition. Dragon Systems' segmentation treats each story as an instance of some underlying topic and models an unbroken text stream as an unlabeled sequence of these topics. In this model, finding story boundaries is equivalent to finding topic transitions. A novel event is hypothesized when the prediction accuracy of the adapted language models drops relative to the background model(s) [17].

1.2.3. Thesis Goal

The above approaches are applicable to news stories in which there is a steady stream of information with little noise. In addition, the number of topic transitions is higher for chat data and the amount of data between transitions is generally much smaller. In the case of chat data, where the amount of noise to information is prohibitively high, we have to adopt a modified approach.

First, we must combine segmentation, event detection, and event tracking. We must preprocess the data to remove noise. Then, similar to event detection, we cluster utterances to identify threads. Finally, similar to event tracking, we must incorporate temporal information to improve this identification of related utterances. In short, our goal is similar to that of segmentation or retrospective event detection – find the stories from a stream of chat. However, we must incorporate techniques from event detection (clustering), and event tracking (temporal heuristics).

1.2.4. Existing Systems

There are several commercially available systems that monitor the sites visited and log utterances from chat sessions on a personal computer. They provide a variety of functionalities such as grabbing text from messenger services(s) and getting window titles of all web pages open on a system. However, none of the systems provide a comprehensive analysis tool to identify topics or track change of topics in a chat session.

1.2.4.1. IamBigBrother

IamBigBrother is one of the leading Internet monitoring software available for both home and business. It runs in stealth mode and captures everything from chats and instant messages to email, titles of visited web sites, among other things. IamBigBrother records all of the Internet activity for many programs including America Online, MSN, and Outlook Express. It also logs all keystrokes typed in every program along with screen shots [13]. The software has the ability to playback chat conversations based on username chatted with and it allows the user to search keywords in the archived conversations. The search functionality is very basic, essentially a linear scan through the archived text, similar to Microsoft Windows' search feature [25].

1.2.4.2. SPY for MSN Messenger

Very similar to IamBigBrother, SPY For MSN Messenger captures and records all conversations in a MSN Chat session. Features include: sending all captured IMs to a specified email address or uploading them to a specified web-site through FTP [15].

1.2.4.3. CLUTO

CLUTO is a software package for clustering low- and high-dimensional datasets and for analyzing the characteristics of the various clusters. CLUTO is well-suited for clustering data sets arising in many diverse application areas including information retrieval, customer purchasing transactions, web, GIS, science, and biology.

CLUTO's distribution consists of both stand-alone programs and a library of functions through which an application program can access directly the various clustering and analysis algorithms implemented in CLUTO.

Features included are:

- Multiple classes of clustering algorithms: partitional, agglomerative, and graph-partitioning based.
- Multiple similarity/distance functions: Euclidean distance, cosine, correlation coefficient, extended Jaccard, and user-defined.
- Numerous novel clustering criterion functions and agglomerative merging schemes
- Traditional agglomerative merging schemes: single-link, complete-link, UPGMA
- Extensive cluster visualization capabilities and output options: postscript, SVG, gif, xfig, etc.
- Multiple methods for effectively summarizing the clusters: most descriptive and discriminating dimensions, cliques, and frequent itemsets.
- Scalability to very large datasets containing hundreds of thousands of objects and tens of thousands of dimensions.

2. Research Approach

In this section, we discuss the approach used to identify utterances that belong to a certain thread of discussion. We will first describe how the chat utterances are archived in our current system and then discuss how we use this system. Then, we

describe the use of document clustering to extract documents that belong to a particular thread. Finally, we will outline the evaluation metrics and the postprocess algorithm that uses the temporal sequence information of an unclustered/incorrectly clustered utterance to identify its most probable cluster.

2.1. Message Archival System

Archiving, or logging, is the process of storing the chat utterances to persistent storage. The ChatTrack system being developed at The University of Kansas, stores the original, unfiltered messages. The messages are stored, one per file, in a directory structure that encodes the channel identifier and date, and the filenames encode the utterance transmission time. These messages are retrievable by date and time, by speaker, by listener, by keywords, and by combinations of the above.

Currently two versions of ChatTrack exist, one each for a server-based chat archive, and one for a client-based chat archive. However, in both versions, the storage functionality is the same. The messages from each session are stored in a separate directory named after the session id. The subdirectory names encode the year, month, and day of the chat utterances in that session. The filenames encode the timestamp, down to the millisecond. Figure 1 diagrams the directory and file structure.

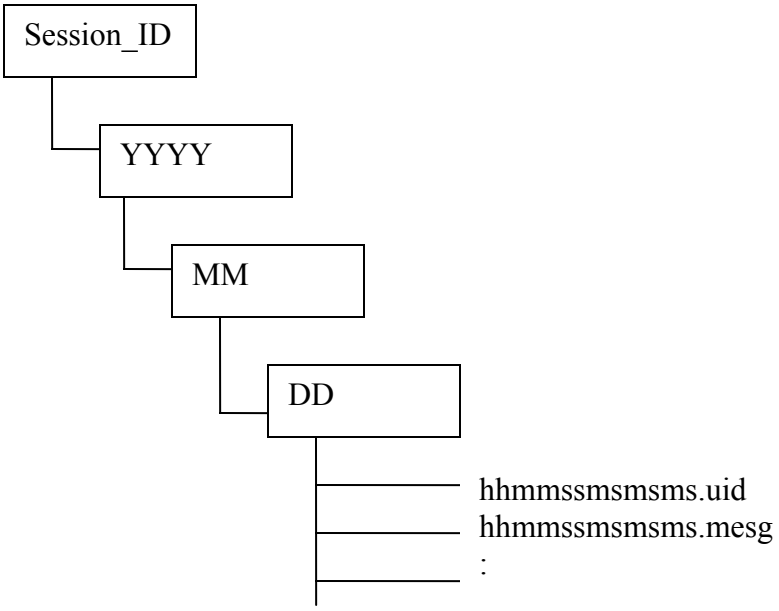

Figure 1 - ChatLog Message Archiving Format

Where,

- Session_ID refers to the ID of the session, assigned by ChatTrack
- YYYY is the current year in 4 digit representation
- MM is the current month in 2 digit representation
- DD is the current date in 2 digit representation.

Each utterance being logged, results in two files being created:

- a message file that contains the actual utterance (ending in .mesg)
- a user information file that stores the chat ID of the speaker (ending in .uid)

These files share a common prefix of the form: “hhmmssmsmsms” where

“hh” is the hour in 2 digits

“mm” is the minute in 2 digits

“ss” is the seconds in 2 digits

“msmsms” is the milliseconds in 3 digits.

Also, there is a file that stores the times at which participants join and leave the session, so that we can track the listeners for each message.

2.2. Document Clustering – CLUTO

Since the utterances are stored as documents, we decided to use document clustering as a method to cluster similar documents, i.e. chat utterances. We must note that on an average each utterance does not take more than 10 words; therefore we treat all utterances from a chat session as a large number of very small sized documents. The clustering procedure occurs in several steps.

2.2.1. Algorithm Selection using test data

In order to apply clustering to chat utterances, we must first select an algorithm that handles documents of very small size well. For this, we create a test data set by interleaving utterances from two or more chatrooms discussing different topics. The 'true thread' for each utterance is already known; it is the chatroom from which the utterance originally came. We then run each of the clustering algorithms in the CLUTO package on this data set, and observe the clustering result. The algorithm that produces the best clusters is then selected for our thread extraction work. The quality of the cluster is given by a measure called the ***FScore Measure*** introduced by [2]. Given a particular class L_r of size n_r and a particular cluster S_i of size n_i , suppose n_{ri} documents in the cluster S_i belong to L_r , then the FScore of this class and cluster is defined to be

$$F(L_r, S_i) = \frac{2 * R(L_r, S_i) * P(L_r, S_i)}{R(L_r, S_i) + P(L_r, S_i)}$$

Equation 2.1 - FScore Measure

Where,

- L_r is the class of size n_r documents (the 'true thread')
- S_i is a cluster of size n_i documents
- $R(L_r, S_i)$ is the recall value defined as n_{ri} / n_i
- $P(L_r, S_i)$ is the precision value defined as n_{ri} / n_r for the class L_r and the cluster Si

The FScore of the class L_r , is the maximum FScore value attained at any node in the hierarchical clustering tree T . That is,

$$F(L_r) = \max_{Si \in T} F(L_r, S_i)$$

Equation 2.2 – FScore value at any node

The FScore of the entire clustering solution is then defined to be the sum of the individual class FScores normalized by the class size, yielding values in [0.0…1.0].

$$FScore = \sum_{r=1}^{c} \frac{n_r}{n} F(L_r)$$

Equation 2.2 – FScore of entire clustering solution

Where

c is the total number of classes.

A perfect clustering solution will be the one in which every class has a corresponding cluster containing all and only the documents for that class, in which case the FScore will be 1.0. In general, the higher the FScore value, the better is the clustering solution [31].

2.2.2. Document clustering using actual data

2.2.2.1. Stopword Removal

The first task is to remove all the stopwords from the utterances, thereby reducing the analysis space to only those words that convey information. Stopwords are provided in a stoplist file. The stoplist file was generated by running a word count on the entire chat corpus to identify words that occur with a high frequency during a chat session. These words were then identified as whether they were stopwords, or if they were information. Some of the identified stopwords from such an analysis were common chat acronyms such as “brb”, “ty”, “oic” and so on, in addition to the usual words that occur most often in the English language. This process of stopword removal actually renders several utterances empty, such as when an utterance contains only stopwords. Under such circumstances, the blank utterances are removed from consideration for clustering purposes.

The processed input utterances are then clustered using the algorithm chosen from the previous step. The resulting clusters are then sent as the input to the post-processing phase.

2.2.2.2. Selection of cluster size

The processed input utterances are clustered using various clustering algorithms and clustering criterion functions. This is done in two phases. In the first phase, we conduct experiments to determine the best clustering algorithm. Once the best clustering algorithm has been chosen, we then proceed to the second phase wherein we conduct experiments on this algorithm using different clustering criterion functions. Thus the idea is to determine which combination of clustering algorithm-

criterion function gives us the best clustering results, and to use those resulting clusters for the first phase of our thread extraction algorithm.

2.2.2.2.1. Selection of clustering algorithm

For evaluating the clustering algorithm, we use a sample test data of input utterances for which we know the truth. We run several clustering algorithms on this data (with all other parameters being the defaults for the CLUTO program)., and compute the FScore for the resulting clusters. The clustering algorithm which gives us the highest FScore is considered to be the best. The experiments are detailed in section 4.2.1.

2.2.2.2.2. Selection of clustering criterion function and clustering quality metric (CQM)

Once the best clustering algorithm has been obtained, we run a series of experiments with several clustering criterion functions for this algorithm, leaving all other parameters set to their default values, on a set of test data, for which the truth is already known. For each clustering criterion function, we run the clustering algorithm for a specified number of clusters. The resulting clusters are then evaluated using a clustering quality metric to find out the optimal number of clusters for which the goodness measure is a maximum. The idea here is to find out if there is any interplay between the clustering quality metric and the clustering criterion function, i.e., to investigate if a particular combination of a clustering quality metric and clustering criterion function gives us consistently good results. The number of clusters is then

increased to the next higher value, and the process is repeated, until we have computed the goodness measure for ‘k’ number of clusters.

For a chat session consisting of several discussions, we vary the value of ‘k’ from 1….N/2, where

‘N’ is the total number of utterances in the session

The experiments are detailed in section 4.3.1.

2.3. Post-processing

Given the small size of the utterances, the statistical similarity between groups of related utterances is not guaranteed to be high. Furthermore, many utterances may be rendered blank due to stopword removal. The output of the clustering process is, therefore, not an accurate indication of the topic threads detected in the entire chat corpus. During post processing, we take into consideration the original chat corpus and, based on the results obtained from the document clustering phase, assign cluster ID’s to the blank utterances based on a heuristic algorithm. In addition, not all input documents will be clustered. Unclustered or erroneously clustered documents are assigned an invalid cluster ID, -1, by CLUTO.

In the first phase of post processing, we attempt to assign valid cluster IDs to the unclustered utterances. For each invalid cluster ID, or string of successive invalid cluster IDs, we identify the valid cluster ID that occurs most often in the surrounding neighborhood. In case of a tie, we choose the valid ID that occurs closest to the invalid sequence. The invalid ID is then replaced by the valid ID that has been identified as given above. We evaluate this approach using a sliding window scheme. In the case of even sized windows, we take an equal number of valid IDs into

consideration, from both the preceding section as well as the succeeding section of IDs. For example, if the window size is 2, then we take 1 cluster ID each from the preceding section as well as the succeeding section for comparison. If the same cluster ID occurs in both locations, this Id is assigned. However, if there is a tie (i.e., two different IDs), we assign the ID that occurred first in the sequence.

In the case of an odd sized window, we are slightly biased towards the preceding section, since we use one more utterance preceding the message than the one succeeding it. We chose this bias because the preceding section is likely to have more valid IDs since it has been processed already. The succeeding section, on the other hand, is yet to be processed and as such may contain more noise IDs which lead to erroneous identification. However, in cases where an invalid ID or a string of invalid cluster IDs occur either at the start of the sequence or at the end of the sequence, we are restricted to using only the succeeding or preceding section of valid cluster IDs respectively.

To illustrate this process, we will now present some example scenarios.

Scenario 1: *Invalid cluster ID's occurring at the start of the sequence.*

For a window size of 6, this would lead to 3 valid cluster IDs are being considered from each of the preceding as well as the succeeding sections. Since, in this case, the preceding section does not exist, we must deduce the valid cluster ID from the succeeding section only. In the example shown in Figure 2, the valid cluster IDs are 1 and 2. Since 1 is more frequent in the given window, we select 1 as the

valid cluster ID and replace all the occurrences of -1 with 1 in the current invalid section.

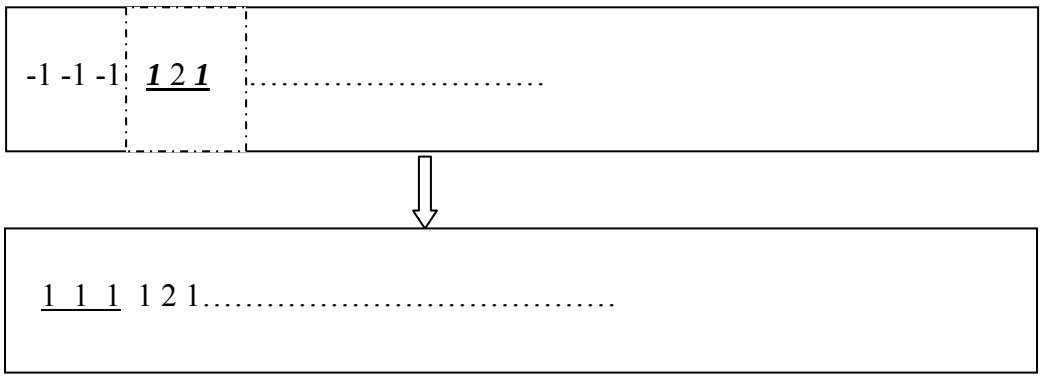

Figure 2 - Invalid cluster ID's at the start of the session

Scenario 2: *Invalid cluster ID's occur at the end of the session*.

For a window size of 6, this would lead to 3 valid cluster IDs being considered from each of the preceding as well as the succeeding sections. In this case the succeeding section does not exist so we must deduce the valid cluster ID from the preceding section only. In the example shown in Figure 3, the valid cluster IDs are 1 and 2. Since 1 is more frequent in the given window, we select 1 as the valid cluster ID and replace all the occurrences of -1 with 1 in the current invalid section.

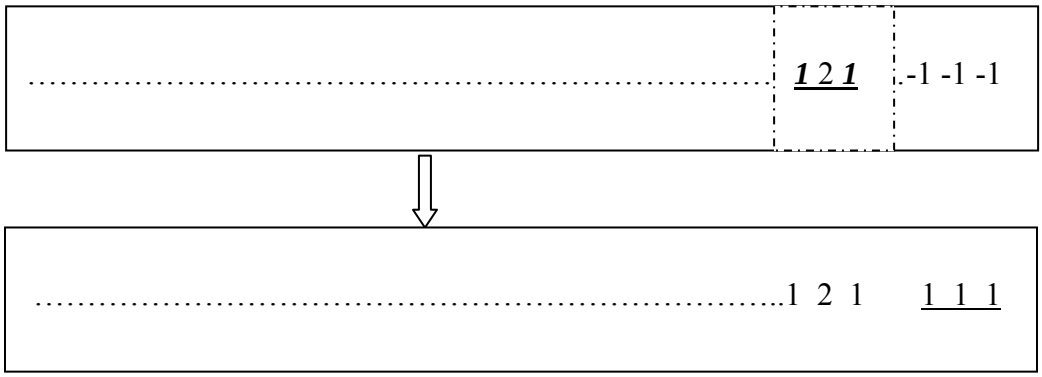

Figure 3 - Invalid cluster ID's at the end of the session

Scenario 3: *Invalid cluster ID's occur in the middle of the session.*

For a window size of 6, this would lead to 3 valid cluster ID's being considered each from the preceding as well as the succeeding sections. For the example shown in Figure 4, the valid cluster IDs are 1 and 2 and they are both equally frequent. However, 1 will be considered as the valid cluster ID, since it occurs in a closer proximity to the string of invalid cluster ID's (as we traverse from the left to right in a temporal sequence).

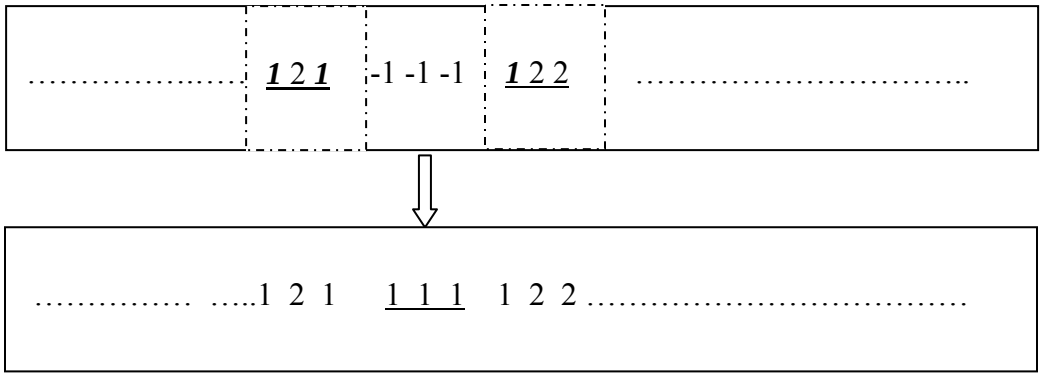

Figure 4 - Invalid cluster ID's in the middle of the sequence

3. System Architecture

This section describes the functionality and interaction between the different components of the system designed for this research. Figure 5 diagrams the tasks performed by these components. The Oval symbol represents a component and the rectangles represent an input/output to/from the system. Following the figure, each component is described separately.

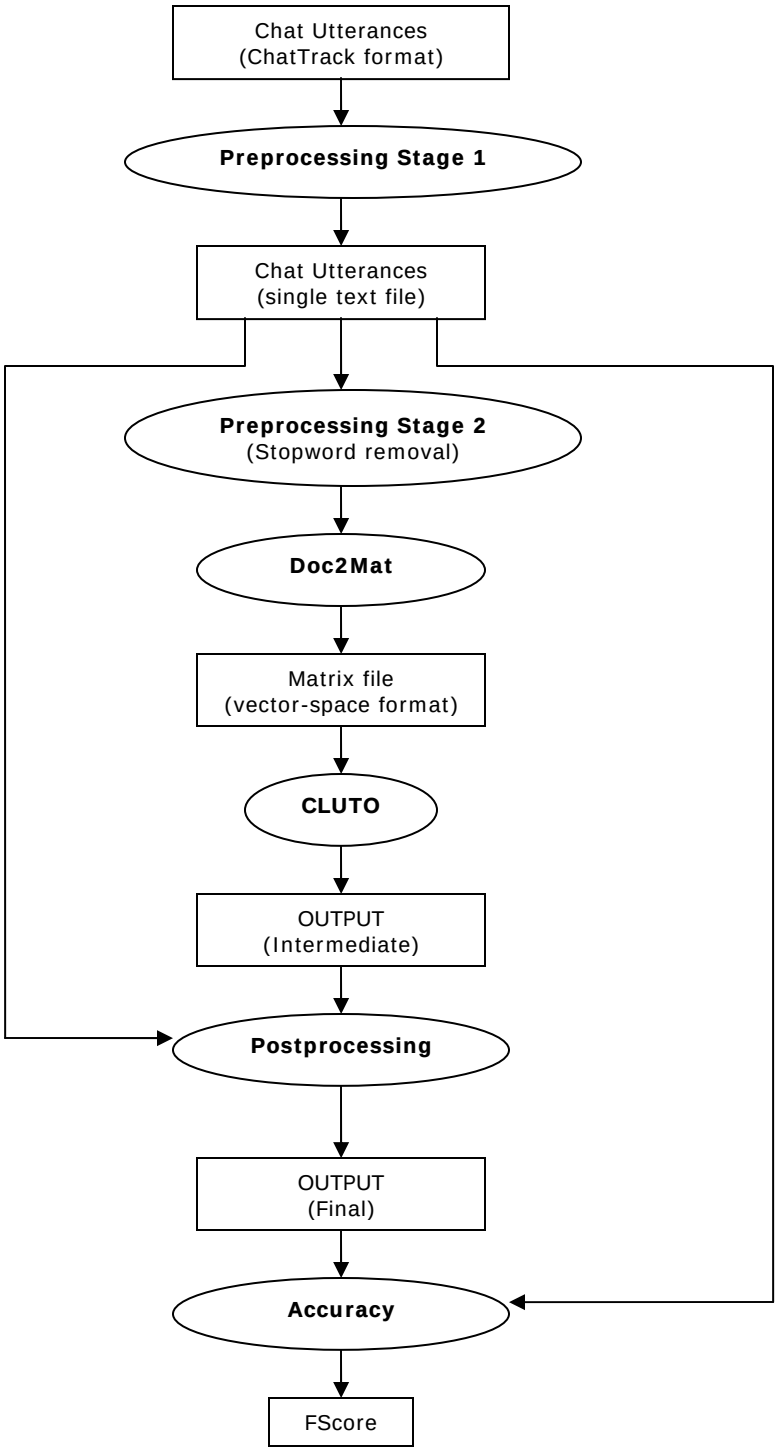

Figure 5 - System Architecture

Preprocessing Stage 1:

The chat utterances are initially archived in the ChatTrack format as described in Section 2.1. This format, however, must be changed to suit the clustering system. The required input format for the clustering system is that all the utterances must be in a single text file. The first preprocessing stage performs this task. Each line of the input file is a document from the ChatTrack format, with the first entry being the document or file name and the second entry being its contents as shown below:

```
1.txt      Hello Ganesh, How r ya ?
2.txt      Much fine, TY, how do you do BTW ?
```

Figure 6 - Chat Utterances in a single text file

Preprocessing Stage 2:

In this stage, we perform stopword removal on the input file, and remove all the words that may be considered as noise. This effectively retains only those words from the utterances that convey information.

Doc2Mat:

CLUTO provides access to its clustering and analysis algorithms via its two standalone programs – *vcluster* and *scluster*. *vcluster* takes as its input a multi-dimensional vector matrix representing the objects that need to be clustered. *scluster*, on the other hand takes a similarity matrix (or graph) between these objects, as its input. We use Doc2Mat to convert the chat utterance input file created by the previous stages into to a vector matrix representation. This matrix file can then be sent as an input to the CLUTO *vcluster* program.

Cluto:

The CLUTO program applies the clustering algorithms on the input vector matrix representation of the objects. The various parameters are passed through the command line interface of the *vcluster* program. The results of the stop-words-eliminated input file is now available in an output file, the name for which was specified as one of the input command line parameters (by default CLUTO prints the output to the screen). The results file contains the cluster ID of the correspond document in the input file, and it is printed one per line. For e.g., if we call the results file as *Results.txt*, then the following would be its output:

```
0
1
```

Figure 7 - CLUTO results – Results.txt

Comparing the cluster ID with the input file (as shown in Fig 6), we conclude that the input chat utterance 1.txt was assigned a cluster ID of 0, and 2.txt was assigned a cluster ID of 1. One thing to note about these ID's is that a negative cluster ID indicates that there has been an error in clustering, and the respective utterance was not clustered.

Processing using the CLUTO program consists of two stages. In the first stage, we wish to identify the algorithm most suited for clustering documents of very small size. This test is run on a data set for which we know the truth value. The algorithm that gives us the highest FScore is considered the most suitable one. In the second stage, we use the identified clustering algorithm to cluster documents on

several topics from a real chat session. Since we want to simulate the real chat session wherein the exact number of topics being discussed is not known apriori , we run the clustering algorithm with required cluster number (passed as one of the parameters to the *vcluster* program) ranging from 2…..n/2, where 'n' is the total number of utterances in the input. We compute the similarity measure of the clusters for each of these numbers. The value of 'n' for which we get the highest similarity measure is then considered as the best number of clusters for our post processing stage.

Post processing Stage:

Here, we augment the results file by considering the actual input file. That is, the results we obtained from CLUTO were for the stop-word-eliminated file, and this file may differ significantly from the original input file since several utterances may be blanked out because they contained only stopwords. We want to obtain the clustering results for the entire corpus of chat utterances. Therefore, we prepare a modified results file taking into consideration all the utterances, even those that had been eliminated due to stop word removal. The resultant file will now contain -1 as the cluster ID for all the files that were thus eliminated. After the post processing, all input files will now have a valid cluster ID, the procedure for doing so has been explained in Section 2.3.

Accuracy analysis:

We want to measure the accuracy of our clustering process. For this, we compare the output generated from the post processing stage with that of a truth file

that we already know for the given input utterances. The accuracy is indicated in the form of an FScore, the details for obtaining which have been described in an earlier section 2.2.1.

4. Evaluation

The objective of this research is to explore whether or not chat utterances can be treated as documents of very small size, and if document clustering techniques can be applied to group related chat utterances together. The general hypothesis of the research is that document clustering techniques can be used to cluster related chat utterances – with some clustering algorithms being more accurate than others. The accuracy can be further enhanced by applying heuristic analyses to group those utterances that may not have been clustered initially. In this section, I will introduce a series of experiments to test this hypothesis.

4.1. Chat Data Generation

Data was collected from IRC chat groups on the following topics: Baseball, Cricket, Politics, and Computers.

4.2. Experiment 1: Selecting a clustering algorithm

CLUTO has 6 clustering algorithms in all. They are:

rb - In this method, the desired *k*-way clustering solution is computed by performing a sequence of $k - 1$ *repeated bisections*.

rbr - In this method, the desired *k*-way clustering solution is computed in a fashion similar to the repeated-bisecting method but, at the end, the overall solution is globally optimized.

direct - In this method, the desired *k*-way clustering solution is computed by simultaneously finding all *k* clusters

agglo - In this method, the desired *k*-way clustering solution is computed using the *agglomerative* paradigm whose goal is to locally optimize (minimize or maximize) a particular clustering criterion function (which is selected using the *-crfun* parameter). The solution is obtained by stopping the agglomeration process when *k* clusters are left.

graph - In this method, the desired *k*-way clustering solution is computed by first modeling the objects using a nearest-neighbor graph (each object becomes a vertex, and each object is connected to its most similar other objects), and then splitting the graph into *k*-clusters using a min-cut graph partitioning algorithm.

bagglo - In this method, the desired *k*-way clustering solution is computed in a fashion similar to the *agglo* method; however, the agglomeration process is biased by a partitional clustering solution that is initially computed on the dataset.

4.2.1. Method

For the purpose of detecting the best clustering algorithm, we created a test chat session data with utterances from

i. A session in which baseball is the main topic of discussion

ii. A session in which cricket is the main topic of discussion

4.2.1.1. Experiment 1.1

The input file, Input.txt, had ten utterances from cricket, followed by 10 utterances from baseball, and then 10 more utterances from cricket, and so on, for a total of 218 utterances. Just to make things a little more challenging for the CLUTO package, I included 19 utterances from the cricket session (instead of the regular 10) and 9 utterances from the baseball session towards the end of the input file. There were in all 110 utterances from the baseball session, and 108 from cricket. Table 1 shows the performance of the different algorithms given this input.

The results were as given:

Algorithm	Precision	Recall	FScore	# of Errors
RB	1	1	1	0
RBR	1	1	1	0
DIRECT	1	1	1	0
AGGLO	0.931422	0.931313	0.93118	14
BAGGLO	0.995413	0.995455	0.995413	1

Table 1 - Clustering Algorithm Results

RB, RBR and DIRECT all produced an astounding 100% precision, recall, and FScore.

AGGLO produced a precision of 93.1422%, recall of 93.1313%, and FScore of 93.118% with 14 errors. (utterance #'s 19, 56, 87, 91, 103, 109, 123, 126, 145, 146, 171, 173, 204, 212 were clustered incorrectly)

BAGGLO produced a precision of 99.5413%, a recall of 99.5455%, and FScore of 99.5413% with 1 error. The erroneous classification was that of a baseball session utterance "out" which was classified as cricket! (utterance # 126)

Please note that we did not evaluate the GRAPH algorithm for clustering. This was because, if the graph contains more than one connected component, then *vcluster* returns a *(k + m)*-way clustering solution where *m* is the number of connected components in the graph. Thus, it is not possible to always obtain the specified number of clusters.

4.2.1.2. Experiment 1.2

In order to verify if the sequence and number of utterances on a topic made any difference, we created the second test input with the same data from the first test input. The only difference was that instead of interleaving 10 utterances from a session, we interleaved 5 utterances each from a session. The result of the clustering process was the same as for the first test case.

4.2.1.3. Experiment 1.3

To conclude the test cases, we created a third test input with the only difference being that we interleaved single messages each from the baseball and cricket chat sessions. The result of the clustering process was the same as for the first test case. From this, we concluded that the number of the chat utterances in a sequence does not have any effect on the clustering process (in terms of precision, recall and FScore measurements). Furthermore, RB, RBR, and DIRECT seem to be the best clustering algorithms for clustering chat data utterances. It has been found by Zhao and Karypis [11], [123], [31], [32] that, in terms of quality, for reasonably small values of *k* (usually less than 10–20), the DIRECT approach leads to better clusters than those obtained via repeated bisections, however it is also slower. For large values of *k*

however, the RB approach tends to be better than the DIRECT approach. Therefore, I chose the RB method, since for my testing purposes, I use a fairly large amount of chat data where the value of *k* may be large.

4.3. Finding the number of Clusters

In this phase, we try to find an initial clustering solution, wherein most similar chat utterances will be grouped together. This procedure consists of two parts:

1. Selection of the best clustering criterion function
2. Selection of an algorithm to yield the optimal number of initial clusters.

We used test data consisting of several utterances (~200) from each of the following four chat sessions:- cricket, baseball, politics, and computers.

4.3.1. Experiment 2: Selecting a clustering criterion function

The following is a list of the various clustering criterion functions which are supported by all the clustering algorithms provided in CLUTO.

i1 - This criterion function maximizes the sum of the average pairwise similarities between the documents assigned to each cluster, weighted according to the size of each cluster.

i2 - In this algorithm, each cluster is represented by its centroid vector and the goal is to find the clustering solution that maximizes the similarity between each document and the centroid of the cluster to which it is assigned.

e1 - This external criterion function was motivated by multiple discriminant analysis and is similar to minimizing the trace of the between-cluster scatter matrix. It tries to

separate the documents of each cluster from the entire collection, as opposed trying to separate the documents among the different clusters.

g1 - Selects the graph-based G1 criterion function.

g1p - Selects the graph-based G1' criterion function.

h1 - This is a hybrid criterion function obtained by combining I1 and E1.

h2 - This is a hybrid criterion function obtained by combining I2 and E1. [31]

George Karypis and team [11] report that the various criterion functions can sometimes lead to significantly different clustering solutions. In general, the *I*2 and *H*2 criterion functions lead to very good clustering solutions, whereas the *E*1 and *G1'* criterion functions lead to solutions that contain clusters that are of comparable size.

In order to find out the best clustering criterion function, we ran the several experiments for each of these functions on test data sets consisting of 3 different topics ranging from the easiest to the difficult – labeled as easy, moderately easy, and real data. The following are our findings:

4.3.1.1. Experiment 2.1

We used a very easy test data set for this experiment. The utterances were on 3 different topics, with 10 utterances from each topic. All utterances from a particular topic were identical.

4.3.1.2. Experiment 2.2

We used a moderately easy test data set for this experiment. The utterances were on 3 different topics, with 10 utterances from each topic. Not all utterances were identical, but some of them differed in the words used, but still were on topic.

4.3.1.3. Experiment 2.3

For this experiment, we intended to use real data as an indication of the complexity involved in the kind of data which need to be analyzed for practical purposes. As before, the utterances were on 3 different topics, with 10 utterances from each topic. Most of the utterances were non-identical, and some did not even contribute any information as regards the topic they were from.

4.3.1.4. Summary

From the above experiments, we obtain the following values:

Criterion Function	FScore		
	Easy data	**Moderately easy data**	**Real data**
I1	1.0	0.966	0.753
I2	1.0	1.0	0.822
E1	1.0	1.0	0.79
H1	1.0	1.0	0.79
H2	**1.0**	**1.0**	**0.861**
G1	1.0	1.0	0.822
G1P	1.0	1.0	0.648

Table 2 – Summary of results for clustering criterion functions

From the above table, we find that all algorithms performed very well for the easy data set giving a maximum FScore of 1.0. For the moderately easy data set, all the algorithms with the exception of I1, gave a 100% clustering result, and as before the maximum FScore was 1.0. For the real data set, though, the **H2** algorithm gives us the best FScore of 0.861. We therefore choose **H2** as the clustering criterion function for analyzing our data, since from the above experimental results, and past experience, we have found this algorithm to yield the best results most consistently.

4.3.2. Measurement Criteria

In general, we will not know *a priori*, how many threads a chat session contains. Thus, we need to develop and evaluate a variety of metrics that will determine the number of threads for us automatically. Thus, we need to come up with a measure of clustering quality and select those metrics that produce their highest values for the correct number of threads.

4.3.2.1. Clustering Quality Metric (CQM)

The clustering quality metric for each cluster is a function of its internal similarity and/or external similarity. The internal similarity of an object is its similarity with other objects within its own cluster. The external similarity of an object is its similarity with objects in other clusters. Objects that have large values of internal similarity and small values of the external similarity will tend to form the core of their clusters.

For each metric, the CQM was calculated over a variety of cluster numbers and the highest value was used to predict the best number of clusters. The best number predicted by a CQM was compared to truth to help determine the most accurate CQM for predicting the number of clusters.

4.3.2.2. Metrics

4.3.2.2.1. Metric 1

$$\text{Similarity Score} = \sum_{i=0}^{n/2} \text{iSim}$$

Hypothesis

A higher value of the internal similarity denotes a highly cohesive group (i.e. a cluster containing all the relevant documents). Therefore, the number of clusters for which the sum of the internal similarities of all the clusters is a maximum shall be considered as the best initial solution.

4.3.2.2.2. Metric 2

$$\text{Similarity Score} = \sum_{i=0}^{n/2} 1/\text{eSim}$$

Hypothesis

A lower value of the external similarity between objects of different clusters indicates the formation of highly cohesive groups. Therefore, the number of clusters for which the sum of the inverses of the external similarities of all the clusters is a maximum shall be considered as the best initial solution.

4.3.2.2.3. Metric 3

$$\text{Similarity Score} = \sum_{i=0}^{n/2} \text{iSim/eSim}$$

Hypothesis

A lower value of the external similarity between objects of different clusters, and a higher value of internal similarity between objects of the same cluster indicates the formation of highly cohesive groups. Therefore, the higher the ratio of internal to external similarity values the better is the quality of the clusters formed. The number of clusters for which the sum of the ratio is a maximum shall be considered as the best initial solution.

4.3.2.2.4. Metric 4

$$\text{Similarity Score} = \sum_{i=0}^{n/2} \text{\# of documents in the cluster} * \text{iSim}$$

Hypothesis

We add a bonus of the number of documents so that the system is not biased towards extremely small clusters of documents. Therefore, the number of clusters for which the sum of the products is a maximum, shall be considered as the best initial solution.

4.3.2.2.5. Metric 5

$$\text{Similarity Score} = \sum_{i=0}^{n/2} \text{\# of documents in the cluster} * \sqrt{(\text{iSim} / \text{eSim})}$$

Hypothesis

We add a bonus of the number of documents so that the system is not biased towards extremely small clusters of documents. The ratio of internal similarity to the external similarity is scaled to be equal to the square-root of their actual values. This is modeled after the SQRT scaling scheme in CLUTO [11] that is used primarily to smooth out large values.

Therefore, the number of clusters for which the sum of the products is a maximum shall be considered as the best initial solution.

4.3.2.2.6. Metric 6

$$\text{Similarity Score} = \sum_{i=0}^{n/2} \frac{\text{\# of documents in the cluster} * \log(\text{iSim} / \text{eSim})}{\text{Total \# of documents in the corpus}}$$

Hypothesis

We add a bonus of the number of documents so that the system is not biased towards extremely small clusters of documents. The ratio of internal similarity to the external similarity is scaled to be equal to the logarithm (base 10) of their actual values. This is modeled after the LOG scaling scheme in CLUTO [11] that is used primarily to smooth out large values.

Therefore, the number of clusters for which the sum of the products is a maximum shall be considered as the best initial solution.

4.3.3. Test Results

As in the previous set of experiments to identify the best clustering criterion function, we are using test data sets on three topics and of varying degrees of complexity: easy, moderately easy, and real data. Further, we are testing this data for clustering over a range of clusters from one through four. The idea is to test the following metrics to see which predicts the best number of clusters closest to the actual number of topics within the chat session. Furthermore, in real cases wherein we lack the *a priori* knowledge of exact number of topics, we want these metrics to identify a good number of clusters before further processing. The following are our findings:

4.3.3.1. Experiment 3.1: Evaluating Clustering Quality Metrics on easy data

4.3.3.1.1. For input cluster number - 1

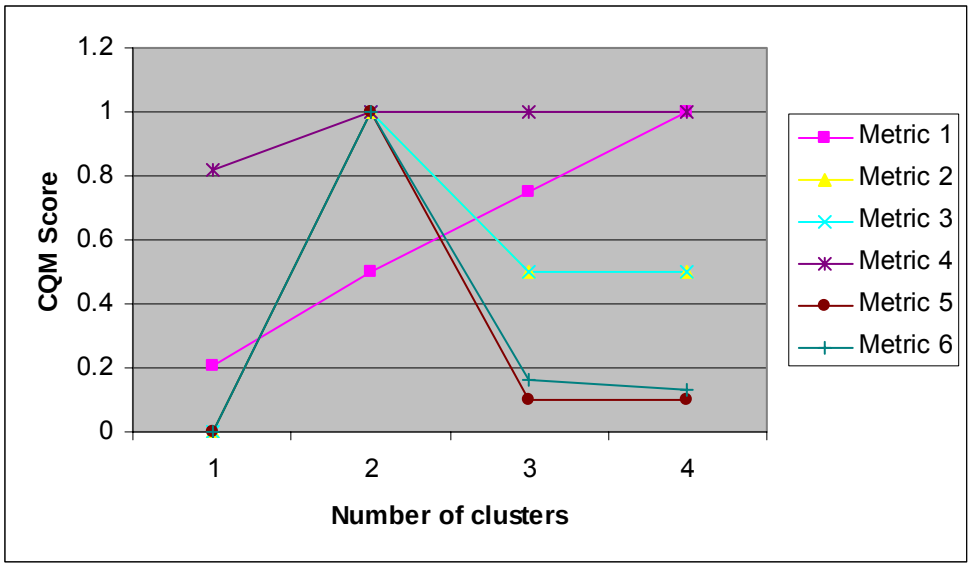

Figure 8 - Plot of CQM Score vs Number of Clusters for Easy data. Input clusters = 1

From the graph, we can see that as we vary the input parameter of the number of clusters the CQM score also varies. With the exception of Metric 1, all the metrics have a maximum CQM score when the input number of clusters was 2. This suggests that the performance of the metrics is suspect when the input contains data from a single topic. For metric 1, the CQM score increased for increasing number of clusters. This is obviously so because it takes into consideration only the internal similarities of the various clusters formed – which will be a maximum when we have a large number of highly cohesive clusters. Under ideal conditions, the internal similarity score will be a maximum when there is just one document per cluster. However, for an actual chat session, this would be an unrealistic scenario.

4.3.3.1.2. For input cluster number - 2

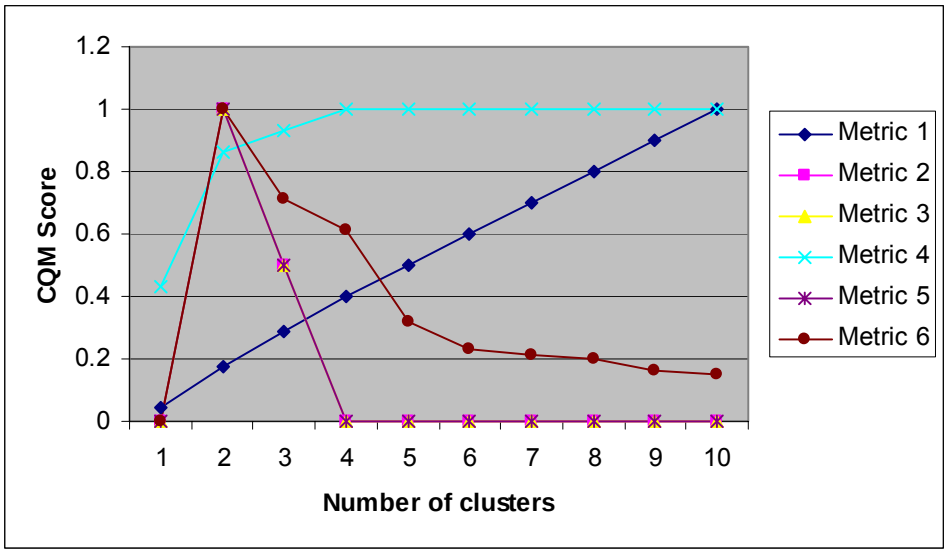

Figure 9 - Plot of CQM Score vs Number of Clusters for Easy data. Input clusters = 2

In this experiment, we note that metrics 2, 3, 5, and 6 yield a perfect similarity score when the input number of clusters is 2, the desired solution. Metric 1 gives an increasing CQM score with increase in number of clusters due to the reasons mentioned in Section 4.3.3.1.1. Metric 4 gives an increasing CQM score as the number of clusters increases and gives a perfect value for cluster numbers 4 and beyond.

4.3.3.1.3. For input cluster number - 3

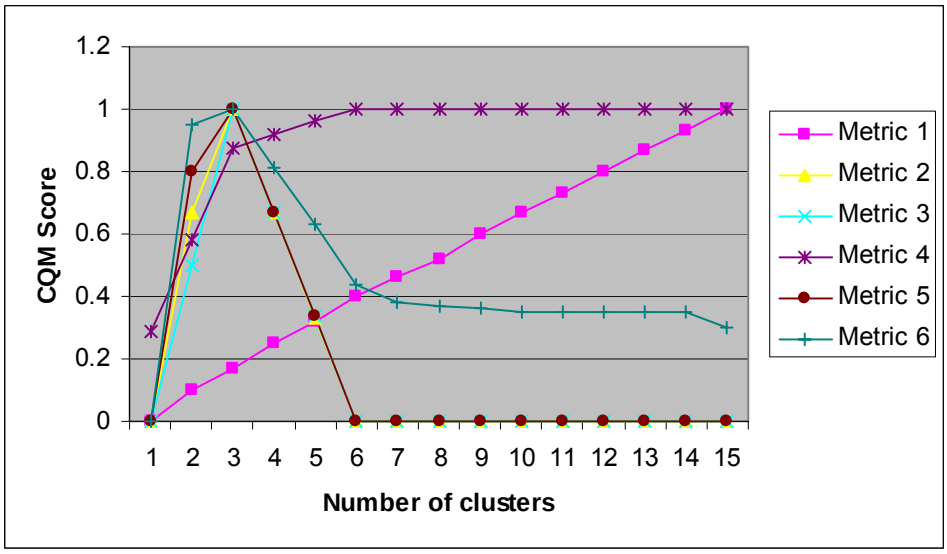

Figure 10 - Plot of CQM Score vs Number of Clusters for Easy data. Input clusters = 3

The results for this case when there are 3 topics present in the chat session is consistent with the results of the previous experiment. A perfect CQM score is obtained for 3 clusters, by metrics 2, 3, 5, and 6.

4.3.3.1.4. For input cluster number - 4

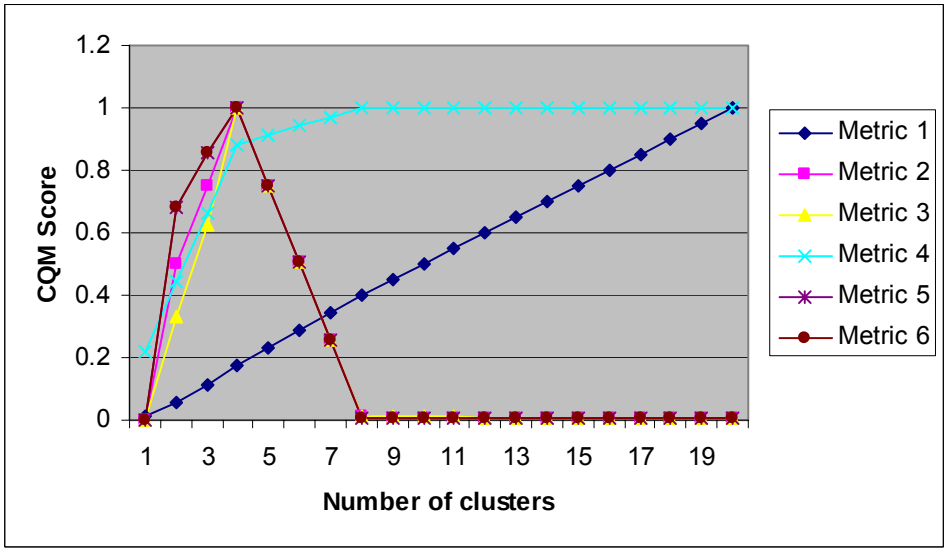

Figure 11 - Plot of CQM Score vs Number of Clusters for Easy data. Input clusters = 4

The results for this case when there are 4 topics present in the chat session is consistent with the results of the previous experiments. A perfect CQM score is obtained for 4 clusters, by metrics 2, 3, 5, and 6.

4.3.3.1.5. Summary

From the above experiments, we can conclude that metrics 2, 3, 5, and 6 perform well for clustering easy data sets.

4.3.3.2. Experiment 3.2: Evaluating the Clustering Quality Metrics on moderately easy data

4.3.3.2.1. For input cluster number - 1

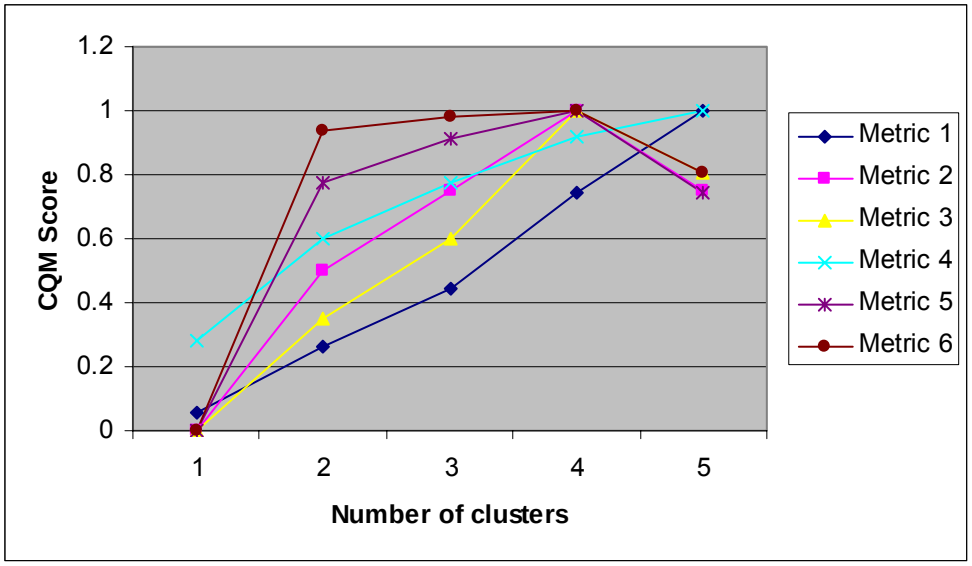

Figure 12 - Plot of CQM Score vs Number of Clusters for Moderately Easy data. Input clusters = 1

As before, all the metrics yield inaccurate results when the input cluster is 1. Metrics 2, 3, 5, and 6 give a perfect result for 4 clusters, whereas metrics 1 and 4 yield a perfect result when the cluster number is maximum.

4.3.3.2.2. For input cluster number - 2

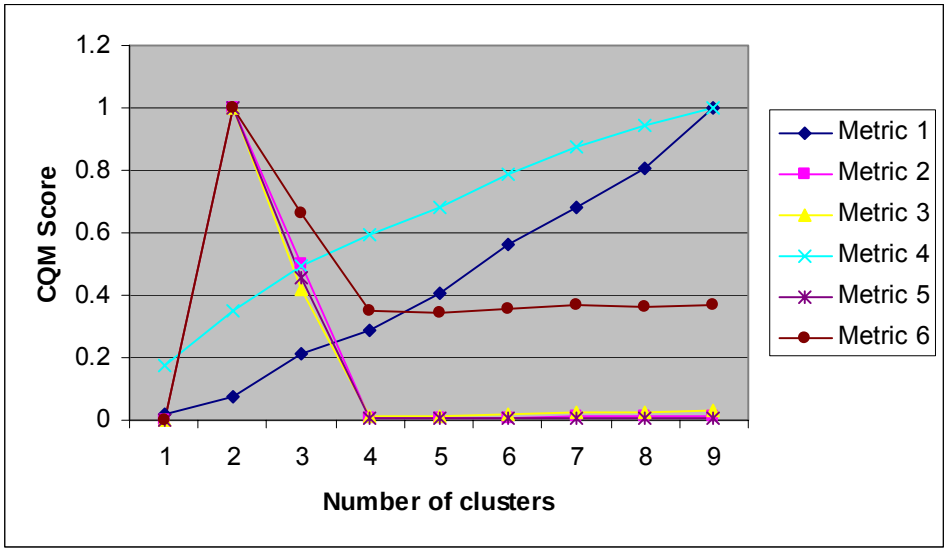

Figure 13 - Plot of CQM Score vs Number of Clusters for Moderately Easy data. Input clusters = 2

The results for this case, when there are 2 topics present in the chat session, is consistent with the results of the corresponding experiment on the easy data set. A perfect CQM score is obtained for 2 clusters, by metrics 2, 3, 5, and 6.

4.3.3.2.3. For input cluster number - 3

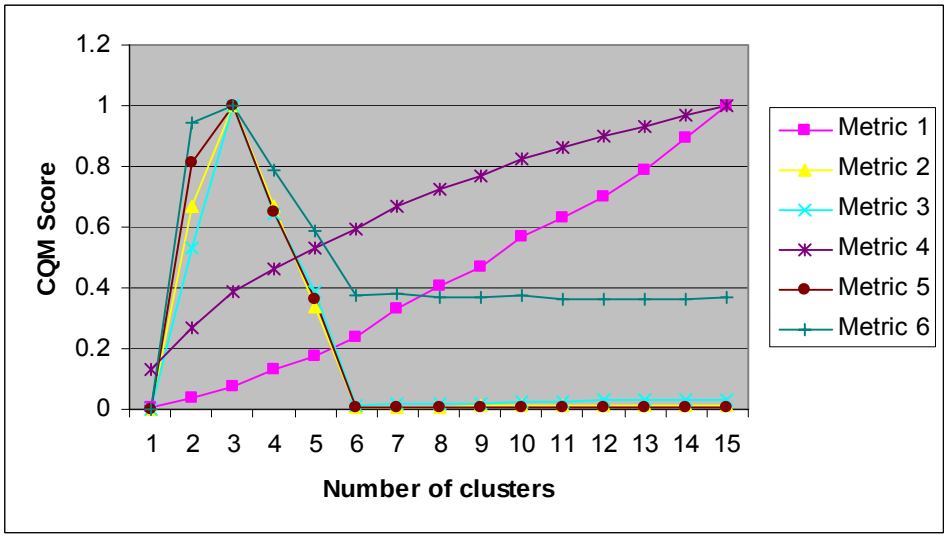

Figure 14 - Plot of CQM Score vs Number of Clusters for Moderately Easy data. Input clusters = 3

The results for this case, when there are 3 topics present in the chat session, is consistent with the results of the corresponding experiment on easy data. A perfect CQM score is obtained for 3 clusters, by metrics 2, 3, 5, and 6.

4.3.3.2.4. For input cluster number – 4

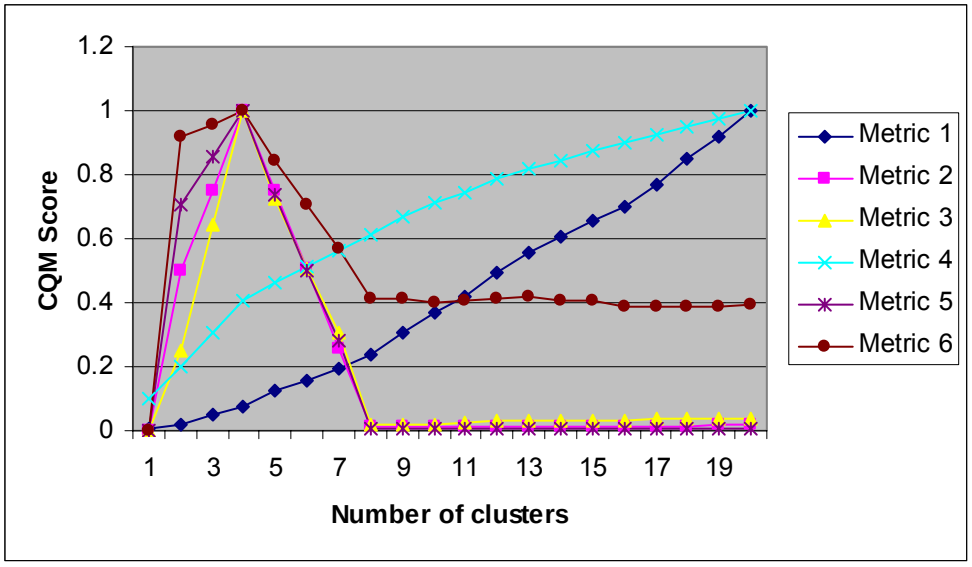

Figure 15 - Plot of CQM Score vs Number of Clusters for Moderately Easy data. Input clusters = 4

The results for this case when there are 4 topics present in the chat session, is consistent with the results of the previous experiments. An optimal CQM score is obtained for 4 clusters, by metrics 2, 3, 5, and 6.

4.3.3.2.5. Summary

From the above experiments, we can conclude that metrics 2, 3, 5, and 6 perform well for clustering moderately easy data sets.

4.3.3.3. Experiment 3.3: Evaluating the Clustering Quality Metrics on real data

4.3.3.3.1. For input cluster number – 1

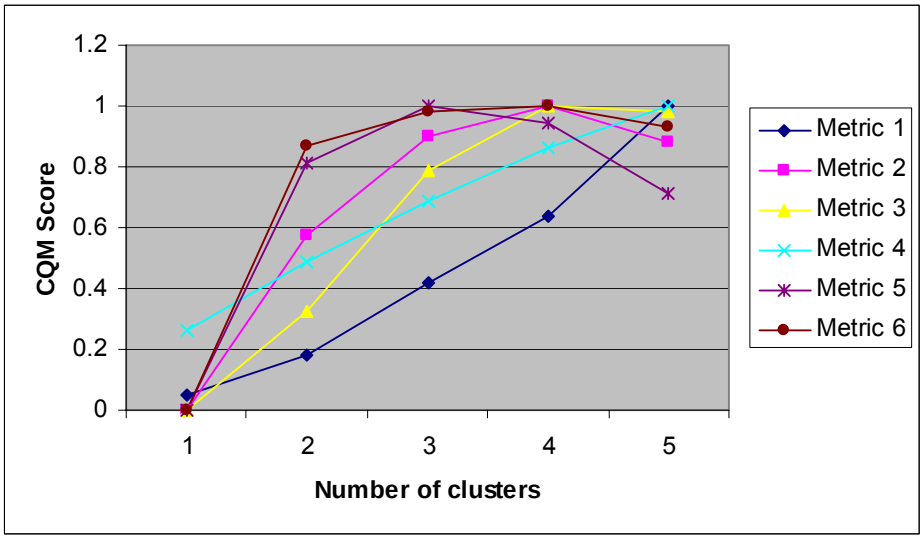

Figure 16 - Plot of CQM Score vs Number of Clusters for Real data. Input clusters = 1

As before, all the metrics yield inaccurate results when the input cluster is 1. Metrics 2, 3, and 6 give a perfect result for 4 clusters, whereas Metric 5 yields a perfect result for 3 clusters. Metrics 1 and 4 yield a perfect result when the cluster number is maximum.

4.3.3.3.2. For input cluster number – 2

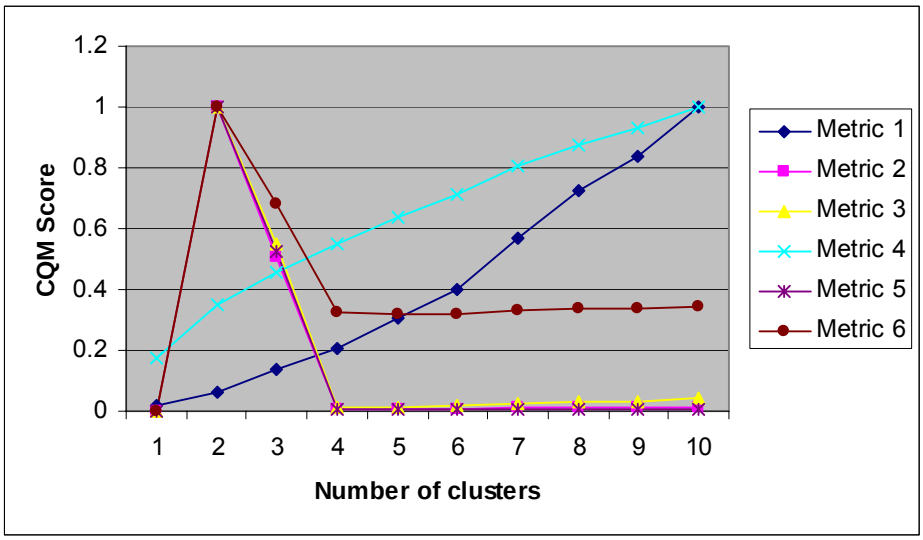

Figure 17 - Plot of CQM Score vs Number of Clusters for Real data. Input clusters = 2

The results for this case, when there are 2 topics present in the chat session, is consistent with the results of the corresponding experiment on the previous data sets. A perfect similarity score is obtained for 2 clusters, by metrics 2, 3, 5, and 6.

4.3.3.3.3. For input cluster number – 3

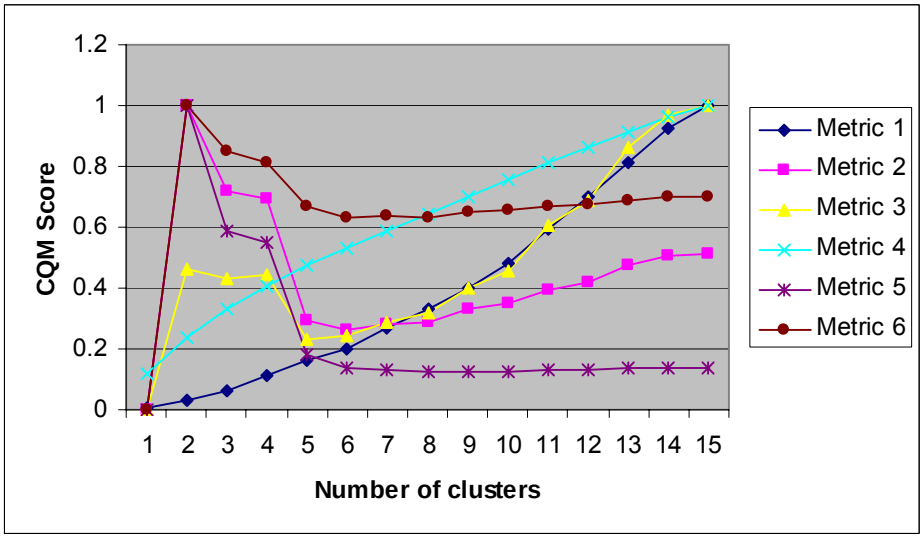

Figure 18 - Plot of CQM Score vs Number of Clusters for Real data. Input clusters = 3

The results for this experiment, when there are 3 topics present in the chat session, is different from the results of the corresponding experiment on the previous data sets.

A perfect similarity score is obtained for 2 clusters, by metrics 2, 5, and 6. Metrics 1, 3 and 4 achieve a maximum CQM score for 15 clusters. Although the detection of the best cluster is not exactly correct, it is however much closer to the actual value of 3.

4.3.3.3.4. For input cluster number – 4

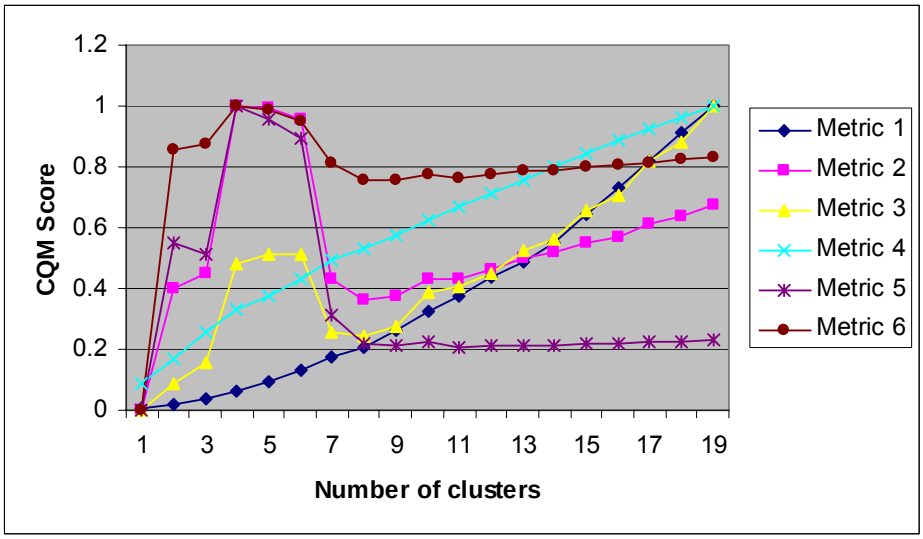

Figure 19 - Plot of CQM Score vs Number of Clusters for Real data. Input clusters = 4

Although there is a much higher degree of variance in the similarity scores for this data set, we see from the plot that metrics 2, 5, and 6 perform very well and yield a maximum value of CQM Score for 4 clusters. Metrics 1, 3 and 4 achieve the maximum value of CQM score for 15 clusters.

4.3.3.3.5. Summary

From the above experiments for the 3 different data sets, we conclude that metrics 2, 5, and 6 perform well for clustering all data sets. Of these, we choose metric 6 for our initial clustering solution, since from experimental and empirical results we have seen that this metric yields intuitive results under most circumstances. Furthermore, due to

the logarithmic scaling, the range of values obtained for the CQM score are very comprehensible.

4.4. Post Processing

Continuing from the previous section, we form the initial clusters using the "h2" clustering criterion. With these clusters in place, we then proceed to perform the post-processing on the results. The clustering is done only on the stopword processed input, not the original chat transcript. Thus, we must prepare an intermediate result file that adds back the input utterances that were removed prior to clustering. These utterances are given an invalid cluster ID of '-1'. As before, we use the FScore as a metric to evaluate the accuracy of the results obtained. Then, we perform post-processing to assign these messages to a cluster.

The following tables show the results before and after the post-processing. There were originally messages from four topics in the input: baseball, computer, cricket, and politics.

	Baseball	**Computers**	**Cricket**	**Politics**
# of Messages	223	216	218	220
# of Unlabelled Messages	22	40	2	18
# of Clusters identified	2	2	2	1
# of Maximum messages clustered	200	175	215	202
# of Clusters with single document	1	1	1	0

Table 3 - Statistics before post-processing

FScore = 0.949

For the input data set, the unprocessed output had an FScore of 0.949. We then applied three post-processing algorithms to this output, and the results of the operations are as shown below.

Algorithm 1 uses a varying window size of valid neighboring cluster IDs to evaluate the missing cluster ID. That is, for a window size of 6, we consider 3 valid cluster IDs that occurred immediately before the invalid cluster ID, and 3 that occurred immediately after. Since it is highly unusual for people to be discussing an unrelated topic for the duration of just one utterance in the middle of a conversation, we assign the cluster ID that occurs most often in the neighborhood around the unassigned message.

We measured the performance of Algorithm 1 for several window sizes ranging from 1 through 79 (this is approximately 1/10 of the size of the input utterances). The FScore was OK when the window size was varied from 1-2. Beyond this point, any increase in the window size affects the FScore adversely. The maximum value of FScore is 0.998, achieved for window sizes 1 and 2. Table 4 shows details of the clustering statistics:

	Baseball	**Computers**	**Cricket**	**Politics**
# of Messages	223	216	218	220
# of Unlabelled Messages	0	0	0	0
# of clusters identified	2	2	2	1
# of correctly clustered Messages	222	215	217	220
# of clusters with single document	1	1	1	0

Table 4 - Statistics after post-processing using varying window size = 1

FScore = 0.998

Algorithm 2 uses the valid cluster ID that precedes the invalid cluster ID (or cluster IDs). The idea here is that, if an utterance could not be clustered for reasons of

stopword removal or otherwise, then it may be construed as a continuation of the existing conversation.

Similar to Algorithm 2, Algorithm 3 uses the valid cluster ID that succeeds the invalid cluster ID (or cluster IDs). The idea in this case is that, if an utterance could not be clustered, then it may be construed as the beginning of a conversation on a different topic of which the current conversation is a part of.

For Algorithms 2 and 3, the FScore is a constant, since we are not varying the window size but only considering either the immediately preceding valid cluster ID, or the one immediately succeeding the invalid ID. The FScore values are, 0.998 for Algorithm 2, and 0.998 for Algorithm 3 respectively.

	Baseball	**Computers**	**Cricket**	**Politics**
# of Messages	223	216	218	220
# of Unlabelled Messages	0	0	0	0
# of clusters identified	2	2	1	1
# of correctly clustered Messages	222	215	218	220
# of clusters with single document	1	1	0	0

Table 5 - Statistics after post-processing using preceding Cluster ID

FScore = 0.998

	Baseball	**Computers**	**Cricket**	**Politics**
# of Messages	223	216	218	220
# of Unlabelled Messages	0	0	0	0
# of clusters identified	2	2	2	1
# of correctly clustered Messages	222	215	216	220
# of clusters with single document	1	1	0	1

Table 6 - Statistics after post-processing using succeeding cluster ID

FScore = 0.998

The following graph indicates the FScores of the post processed output, as against the unprocessed output.

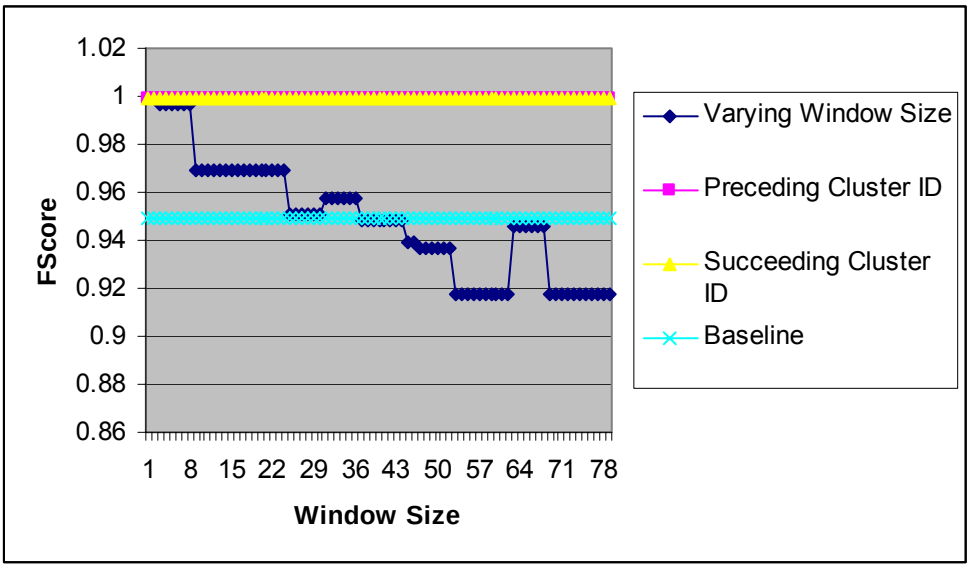

Figure 20 Post processed output – A comparison of the performance of the 3 algorithms

This experiment shows that the results from the 3 different algorithms are comparable, and each improves the quality of thread detection.

4.5. Detailed Example

In this section, we will take you through the system and demonstrate how it works. For this demonstration, we are using the input data that was used in the preceding experiments. We start with the creation of input file from the ChatTrack format into the required Input file format that can then be used for further processing.

Figure 21 shows the chat utterances ChatTrack format. We have used utterances from four different topics for testing our idea. Each topic is contained in a separate directory and is depicted as a box in the diagram. All utterances are stored in

individual files, and for the sake of representation they have all been included in the box as separate entries.

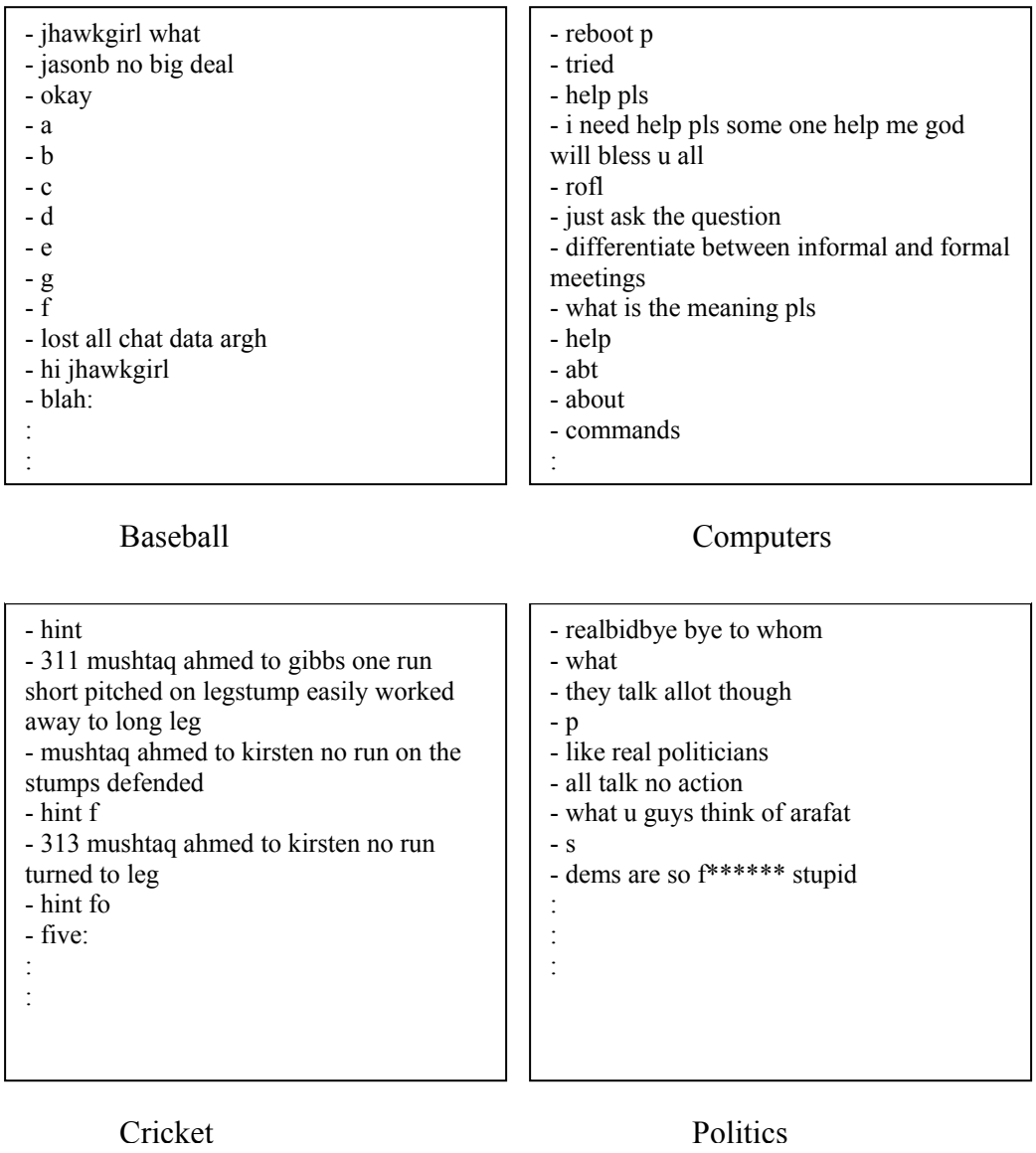

Figure 21 Sample Utterances in ChatTrack Format

The utterances are then converted from the ChatTrack format to the one as shown in Figure 22, the format that is understood and used by our thread detection system. A detailed description is given in Section 5, System Architecture.

```
baseball/.190.txt   jhawkgirl what
baseball/.191.txt   jasonb no big deal
baseball/.192.txt   okay
baseball/.193.txt   a
baseball/.194.txt   b
baseball/.195.txt   c
baseball/.196.txt   d
baseball/.197.txt   e
baseball/.198.txt   g
baseball/.199.txt   f
baseball/.2.txt     lost all chat data argh
computers/.090022000.mesg       reboot p
computers/.090248000.mesg       tried
computers/.091106000.mesg       help pls
computers/.091117000.mesg       i need help pls some one help me god will bless u all
computers/.091334000.mesg       rofl
computers/.091340000.mesg       just ask the question
computers/.091423000.mesg       differentiate between informal and formal meetings
computers/.091428000.mesg       what is the meaning pls
computers/.095125000.mesg       help
computers/.095143000.mesg       abt
computers/.095146000.mesg       about
computers/.095159000.mesg       commands
cricket/.93.txt             hint
cricket/.94.txt             311 mushtaq ahmed to gibbs one run short pitched on legstump easily
worked away to long leg
cricket/.95.txt              312 mushtaq ahmed to kirsten no run on the stumps defended
cricket/.96.txt             hint f
cricket/.97.txt             313 mushtaq ahmed to kirsten no run turned to leg
cricket/.98.txt             hint fo
cricket/.99.txt             five
politics/.180.txt   realbidbye bye to whom
politics/.181.txt   what
politics/.182.txt   they talk allot though
politics/.183.txt   p
politics/.184.txt   like real politicians
politics/.185.txt   all talk no action
politics/.186.txt   what u guys think of arafat
politics/.187.txt   s
politics/.188.txt   dems are so f****** stupid
```

Figure 22 - Preprocessed Input file

This file must then be processed to remove the stopwords. Figure 23 shows the results of this procedure on the above given input. Please note that several utterances from

Figure 22 are missing in the following figure. This is a consequence of stopword removal on those utterances that contain only stopwords

```
baseball/.190.txt   jhawkgirl
baseball/.191.txt   jasonb deal
baseball/.192.txt   okay
baseball/.2.txt             lost chat data
computers/.090022000.mesg       reboot
computers/.090248000.mesg       tried
computers/.091106000.mesg       pls
computers/.091117000.mesg       pls god bless
computers/.091334000.mesg       rofl
computers/.091340000.mesg       question
computers/.091423000.mesg       differentiate informal formal meetings
computers/.091428000.mesg       meaning pls
computers/.095143000.mesg       abt
computers/.095159000.mesg       commands
cricket/.93.txt             hint
cricket/.94.txt             311 mushtaq ahmed gibbs run short pitched legstump easily leg
cricket/.95.txt             312 mushtaq ahmed kirsten run stumps defended
cricket/.96.txt             hint
cricket/.97.txt             313 mushtaq ahmed kirsten run leg
cricket/.98.txt             hint fo
politics/.180.txt   realbidbye
politics/.182.txt   talk allot
politics/.184.txt   real politicians
politics/.185.txt   talk action
politics/.186.txt   guys arafat
politics/.188.txt   dems stupid
.
```

Figure 23 - Input file after Stopword removal

The following figure shows the results of clustering. The resulting cluster IDs are stored in a separate “Results” file. For convenience however, in this figure, they have been shown alongside each utterance, so that it is easier for us to identify the group of utterances that were identified as part of one cluster.

baseball/.190.txt	jhawkgirl	1
baseball/.191.txt	jasonb deal	1
baseball/.192.txt	okay	1
baseball/.2.txt	lost chat data	1
computers/.090022000.mesg	reboot	0
computers/.090248000.mesg	tried	0
computers/.091106000.mesg	pls	0
computers/.091117000.mesg	pls god bless	0
computers/.091334000.mesg	rofl	0
computers/.091340000.mesg	question	0
computers/.091423000.mesg	differentiate informal formal meetings	0
computers/.091428000.mesg	meaning pls	0
computers/.095143000.mesg	abt	0
computers/.095159000.mesg	commands	0
cricket/.93.txt	hint	2
cricket/.94.txt	311 mushtaq ahmed gibbs run short pitched legstump easily leg	2
cricket/.95.txt	312 mushtaq ahmed kirsten run stumps defended	2
cricket/.96.txt	hint	
cricket/.97.txt	313 mushtaq ahmed kirsten run leg	2
cricket/.98.txt	hint fo	2
politics/.180.txt	realbidbye	
politics/.182.txt	talk allot	3
politics/.184.txt	real politicians	3
politics/.185.txt	talk action	3
politics/.186.txt	guys arafat	3
politics/.188.txt	dems stupid	3

Figure 24 - Clustered results

The utterances which had been removed from clustering due to stopword removal, must now be reintroduced in the final analysis. Since they were not clustered, we must assign an invalid cluster ID of -1 to them. This is as shown in Figure 25:

baseball/.190.txt	jhawkgirl what	1
baseball/.191.txt	jasonb no big deal	1
baseball/.192.txt	okay	1
baseball/.193.txt	a	-1
baseball/.194.txt	b	-1
baseball/.195.txt	c	-1
baseball/.196.txt	d	-1
baseball/.197.txt	e	-1
baseball/.198.txt	g	-1
baseball/.199.txt	f	-1
baseball/.2.txt	lost all chat data argh	1
computers/.090022000.mesg	reboot p	0
computers/.090248000.mesg	tried	0
computers/.091106000.mesg	help pls	0
computers/.091117000.mesg	i need help pls some one help me god will bless u all	0
computers/.091334000.mesg	rofl	0
computers/.091340000.mesg	just ask the question	0
computers/.091423000.mesg	differentiate between informal and formal meetings	0
computers/.091428000.mesg	what is the meaning pls	0
computers/.095125000.mesg	help	0
computers/.095143000.mesg	abt	0
computers/.095146000.mesg	about	-1
computers/.095159000.mesg	commands	0
cricket/.93.txt	hint	2
cricket/.94.txt	311 mushtaq ahmed to gibbs one run short pitched on legstump easily worked away to long leg	2
cricket/.95.txt	312 mushtaq ahmed to kirsten no run on the stumps defended	2
cricket/.96.txt	hint f	
cricket/.97.txt	313 mushtaq ahmed to kirsten no run turned to leg	2
cricket/.98.txt	hint fo	2
cricket/.99.txt	five	2
politics/.180.txt	realbidbye bye to whom	3
politics/.181.txt	what	-1
politics/.182.txt	they talk allot though	3
politics/.183.txt	p	-1
politics/.184.txt	like real politicians	3
politics/.185.txt	all talk no action	3
politics/.186.txt	what u guys think of arafat	3
politics/.187.txt	s	-1
politics/.188.txt	dems are so f****** stupid	3

Figure 25 - Clustered results after the reintroduction of all Input utterances

We must now apply postprocessing to assign a valid cluster ID to the invalid ones based on the three algorithms we discussed above. For this example, we have used Algorithm 1 because we have found it to be more robust in terms of performance.

baseball/.190.txt	jhawkgirl what	1
baseball/.191.txt	jasonb no big deal	1
baseball/.192.txt	okay	1
baseball/.193.txt	a	1
baseball/.194.txt	b	1
baseball/.195.txt	c	1
baseball/.196.txt	d	1
baseball/.197.txt	e	1
baseball/.198.txt	g	1
baseball/.199.txt	f	1
baseball/.2.txt	lost all chat data argh	1
computers/.090022000.mesg	reboot p	0
computers/.090248000.mesg	tried	0
computers/.091106000.mesg	help pls	0
computers/.091117000.mesg	i need help pls some one help me god will bless u all	0
computers/.091334000.mesg	rofl	0
computers/.091340000.mesg	just ask the question	0
computers/.091423000.mesg	differentiate between informal and formal meetings	0
computers/.091428000.mesg	what is the meaning pls	0
computers/.095125000.mesg	help	0
computers/.095143000.mesg	abt	0
computers/.095146000.mesg	about	0
computers/.095159000.mesg	commands	0
cricket/.93.txt	hint	2
cricket/.94.txt	311 mushtaq ahmed to gibbs one run short pitched on legstump easily worked away to long leg	2
cricket/.95.txt	312 mushtaq ahmed to kirsten no run on the stumps defended	2
cricket/.96.txt	hint f	2
cricket/.97.txt	313 mushtaq ahmed to kirsten no run turned to leg	2
cricket/.98.txt	hint fo	2
cricket/.99.txt	five	2
politics/.180.txt	realbidbye bye to whom	3
politics/.181.txt	what	3
politics/.182.txt	they talk allot though	3
politics/.183.txt	p	3
politics/.184.txt	like real politicians	3
politics/.185.txt	all talk no action	3
politics/.186.txt	what u guys think of arafat	3
politics/.187.txt	s	3
politics/.188.txt	dems are so f****** stupid	3

Figure 26 - Clustered Results after Postprocessing

5. VALIDATION

In order to verify that the system is capable of detecting threads in a real chat session for which the number of topics is not already known, we will repeat the experiments on a test data set that has not been used to train the system. We will evaluate this both formally, by giving the metric scores for the performance of the system, and informally, by showing the extracted threads. We will first show the formal validation.

5.1. Formal Validation

The test data set consist of about 200 utterances each from a real chat session on 4 different topics. In this section we will present the metrics used and the values obtained for experiments performed on this test data set. This will help us in improving our understanding of the performance of the system.

The statistics for the clusters formed before and after postprocessing phase are as shown:

The statistics before postprocessing are as shown in Figure 40.

	Computers	**Lakers**	**Angeleyez**	**Pyroshells**
# of Messages	217	262	268	246
# of Unlabelled Messages	42	74	51	79
# of clusters identified	2	2	2	1
# of correctly clustered Messages	174	187	216	167
# of clusters with single document	1	1	1	0

Figure 27 - Statistics before postprocessing

FScore = 0.86

	Computers	Lakers	Angeleyez	Pyroshells
# of Messages	217	262	268	246
# of Unlabelled Messages	0	0	0	0
# of clusters identified	2	2	2	1
# of correctly clustered Messages	216	261	267	246
# of clusters with single document	1	1	1	0

Figure 28 - Statistics after postprocessing

FScore = 1

Thus we see that for the formal validation on a new test data set, the results are consistent with the ones obtained in the evaluation section. The FScore in both cases is very high with 0.998 for the evaluation experiments, and 1 for the formal validation experiment.

5.2. Informal Validation

As before, we must convert the input chat utterances from the ChatTrack format to the CLUTO format of document ID and message as shown below:

```
083509000.mesg  Is it me or are there others that like to leave work for lunch?
083519000.mesg  been pretty quiet
083542000.mesg  Its always quiet during the day in here
083630000.mesg  your tellin me
083653000.mesg  Im on lunch. so I have to leave in hmmmm 15 mins or so
:
:
093158000.mesg  i like my arteries the way they are
093212000.mesg  which is why i dont have mcdonals breakasts :
093258000.mesg  lmao i guess i dont like my arteries
093309000.mesg  since i eat there 3 times a week lmao
093316000.mesg  only breakfast tho
:
:
093619000.mesg  :P
093633000.mesg  you should think about giving up smoking
093636000.mesg  0_o
093640000.mesg  o_O
093644000.mesg  um duh
093645000.mesg  0_0
:
:
094442000.mesg  I'll give you some of mine too
094445000.mesg  lol
094452000.mesg  seriously
094456000.mesg  so am I
094459000.mesg  I'm only 120 lbs
:
:
100250000.mesg  LaZaRuS: excuse me?
100307000.mesg  o_O
100309000.mesg  shyel: must not have been bad
100312000.mesg  excuse?
:
:
113329001.mesg  methodX: I forgot snook
113334000.mesg  lol
113335000.mesg  bot snook is a yeast infected ****
113335001.mesg  OK, methodX.
113339000.mesg  bot snook
```

Figure 29 - Preprocessed Input

Stopword removal on the above input data yields the input as shown below in Figure 28:

```
083509000.mesg  leave lunch
083519000.mesg  pretty
083542000.mesg  day
083630000.mesg  telling
083653000.mesg  lunch leave hmmmm 15 mins
:
:
093158000.mesg  arteries
093212000.mesg  mcdonals breakasts
093258000.mesg  guess arteries
093309000.mesg  eat 3 times
093316000.mesg  breakfast tho:
:
:
093633000.mesg  giving smoking
093636000.mesg  0o
093640000.mesg  oo
093644000.mesg  um duh
093645000.mesg  00:
:
:
094442000.mesg  mine
094452000.mesg  seriously
094459000.mesg  120 lbs
:
:
100250000.mesg  lazarus excuse
100307000.mesg  oo
100309000.mesg  shyel
100312000.mesg  excuse
:
:
113329001.mesg  methodx forgot snook
113335000.mesg  bot snook yeast infected ****
113335001.mesg  methodx
113339000.mesg  bot snook
```

Figure 30 - Stopword removed Input file

The input is then clustered to yield the results as shown in Figure 29:

083509000.mesg leave lunch	121
083519000.mesg pretty	99
083542000.mesg day	108
083630000.mesg telling	-1
083653000.mesg lunch leave hmmmm 15 mins	121
:	:
:	:
093158000.mesg arteries	114
093212000.mesg mcdonals breakasts	-1
093258000.mesg guess arteries	114
093309000.mesg eat 3 times	51
093316000.mesg breakfast tho:	68
:	:
:	:
093633000.mesg giving smoking	69
093636000.mesg 0o	-1
093640000.mesg oo	-1
093644000.mesg um duh	-1
093645000.mesg 00:	-1
:	:
:	:
094442000.mesg mine	5
094452000.mesg seriously	-1
094459000.mesg 120 lbs	105
:	:
:	:
100250000.mesg lazarus excuse	26
100307000.mesg oo	-1
100309000.mesg shyel	73
100312000.mesg excuse	57
:	:
:	:
113329001.mesg methodx forgot snook	50
113335000.mesg bot snook yeast infected ****	25
113335001.mesg methodx	55
113339000.mesg bot snook	64

Figure 31 - Clustering Results

The input utterances that were rendered empty due to stopword removal during the input processing phase are now reinserted into this sequence. All documents that were not clustered or were reinserted are assigned an invalid cluster ID of -1. The resulting input and results file look as shown below in Figure 30:

083509000.mesg	Is it me or are there others that like to leave work for lunch?	121
083519000.mesg	been pretty quiet	99
083542000.mesg	Its always quiet during the day in here	108
083630000.mesg	your tellin me	-1
083653000.mesg	Im on lunch. so I have to leave in hmmmm 15 mins or so	121
:		:
:		:
093158000.mesg	i like my arteries the way they are	114
093212000.mesg	which is why i dont have mcdonals breakasts :	-1
093258000.mesg	lmao i guess i dont like my arteries	114
093309000.mesg	since i eat there 3 times a week lmao	51
093316000.mesg	only breakfast tho	68
:		:
:		:
093619000.mesg	:P	-1
093633000.mesg	you should think about giving up smoking	69
093636000.mesg	0_o	-1
093640000.mesg	o_O	-1
093644000.mesg	um duh	-1
093645000.mesg	0_0	-1
:		:
:		:
094442000.mesg	I'll give you some of mine too	5
094445000.mesg	lol	-1
094452000.mesg	seriously	-1
094456000.mesg	so am I	-1
094459000.mesg	I'm only 120 lbs	105
:		:
:		:
100250000.mesg	LaZaRuS: excuse me?	26
100307000.mesg	o_O	-1
100309000.mesg	shyel: must not have been bad	73
100312000.mesg	excuse?	57
:		:
:		:
113329001.mesg	methodX: I forgot snook	50
113334000.mesg	lol	-1
113335000.mesg	bot snook is a yeast infected ****	25
113335001.mesg	OK, methodX.	55
113339000.mesg	bot snook	64

Figure 32 - Clustered Results before Postprocessing

This modified set of Input and Result files are then fed into the Postprocessing phase, to assign valid cluster IDs. The resulting cluster IDs are as shown in Figure 31.

083509000.mesg	Is it me or are there others that like to leave work for lunch?	121
083519000.mesg	been pretty quiet	99
083542000.mesg	Its always quiet during the day in here	108
083630000.mesg	your tellin me	99
083653000.mesg	Im on lunch. so I have to leave in hmmmm 15 mins or so	121
:		:
:		:
093158000.mesg	i like my arteries the way they are	114
093212000.mesg	which is why i dont have mcdonals breakasts :	114
093258000.mesg	lmao i guess i dont like my arteries	114
093309000.mesg	since i eat there 3 times a week lmao	51
093316000.mesg	only breakfast tho	68
:		:
:		:
093619000.mesg	:P	36
093633000.mesg	you should think about giving up smoking	69
093636000.mesg	0_o	36
093640000.mesg	o_O	36
093644000.mesg	um duh	36
093645000.mesg	0_0	36
:		:
:		:
094442000.mesg	I'll give you some of mine too	5
094445000.mesg	lol	5
094452000.mesg	seriously	5
094456000.mesg	so am I	5
094459000.mesg	I'm only 120 lbs	105
:		:
:		:
100250000.mesg	LaZaRuS: excuse me?	26
100307000.mesg	o_O	26
100309000.mesg	shyel: must not have been bad	73
100312000.mesg	excuse?	57
:		:
:		:
113329001.mesg	methodX: I forgot snook	50
113334000.mesg	lol	25
113335000.mesg	bot snook is a yeast infected ****	25
113335001.mesg	OK, methodX.	55
113339000.mesg	bot snook	64

Figure 33 - Clustered Results after Postprocessing

From the identified clusters, we can form the following discussion threads. A total of 122 clusters were identified numbered from 0-121. We will first take a look at the Clustering Quality Metric Score obtained for this data set. The experiment was run on this data for the number of clusters ranging from 2 to 303. Of these, the CQM

Score of 1.70143 was highest for 122 input clusters. We thus choose 122 as the probably number of discussion threads in the given chat session

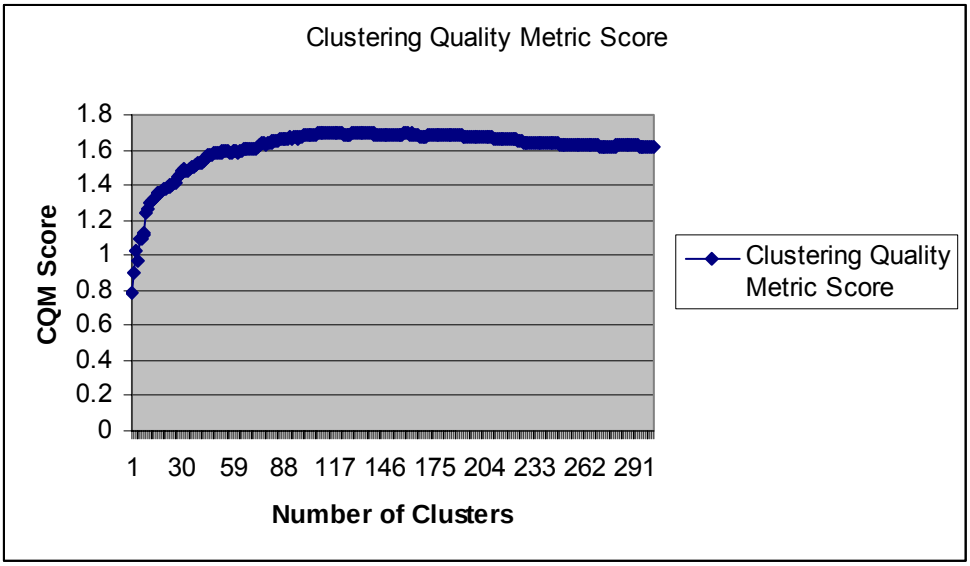

Figure 34 - CQM Score of Input chat session

We will provide a sample of the identified threads as shown in the following figures:

wb PUP
wb fatboy
What do Cicadas eat?
sightime for work again
bye guys
I may very well eat my arm
I'm suddenly starving
dammit
there's nothing I want

Figure 35 - Thread 51

The utterances in Thread 51 seem to follow a discussion on hunger and food.

lmao
heh
lol
bot methodX
well, methodX is a bad*** ********!
OK, methodX.

Figure 36 - Thread 55

Thread 55 discusses about methodX.

Strawberry cheesecake that's what!
my slab
lmao
mmmm
that would be soo good
steaak
i had choclate cake with strawberries

Figure 37 - Thread 58

Thread 58 seems to be on strawberry cakes.

I'd say you owe me a beer, but I'm allergic to that too):(
im not
allergic to beer!
ill take it
yes
allergic to beer

Figure 38 - Thread 88

Thread 88 seems to be following a discussion on allergies and beer !

now i'm on roids
grouchy, lmao
hehe
so am I
last roid was taken today
I'm also weepy and hyper
alternately

Figure 39 - Thread 112

Thread 112 seems to be on “roids”.

[Whitewolf] Hey Whitey, wheres your hat?
i know.
hey Darkness
Darkness is Whitewolf
I know you know ;)
who is she?
it has been said that she is not cute.
me
wb?
rumour has it wb is thanks

Figure 40 - Thread 5

Thread 118 seems to be about Whitewolf, and Darkness.

i think Maester_BlueAecid is rose BA 's Master, and her true love.
awwww
awwwwwwwwwwwwwww
lol
1 year old in like 2 weeks
rose BA
Maester_BlueAecid: rose BA is the most beautiful girl ever set upon this world, and the girl that wears Maester BlueAecid's collar and Maester BlueAecid's ring. *purrs*

Figure 41 - Thread 115

Thread 115 seems to be discussing “Maester_BlueAcid”.

We see that the results in the informal validation are not nearly as impressive as those for the formal validation. Let us take a closer look at the statistics for both the formal and informal validation, and try to deduce the reason for this dramatic decrease in performance.

Parameter	Formal Validation	Informal Validation
# of utterances	993	800
Average length of messages in input (# of words / message)	5.21	4.62
# of utterances after stopword removal	747	607
# of unique words in stopword removed input	1152	715
Average length of messages in stopword removed input (# of words / message)	2.28	2.54
Average # of occurrences of a word (after stopword removal)	1.48	2.16
Total number of utterances not clustered by CLUTO	0	144
Number of clusters identified by CQM	4 (out of 2…496)	123 (out of 2…303)

Table 7 - Statistical comparison of formal and informal validation

From the above values we can see that although most of the numbers are similar and comparable for both formal and informal validation, the average message length in the formal validation dropped from 5.21 to 2.28 due to stopword removal, that is, a reduction by almost 3 words. Comparing this with the informal validation, we find that the decrease is only about 2 words, from 4.62 to 2.54. This indicates that stopword removal has not been as effective for informal validation as it had been for formal validation, with the result being that most of the noise words were passed to the clustering tool as information. This in turn led to a total of 144 utterances ending up unclustered in the informal validation experiment, as opposed to 0 for the formal validation. Thus by employing a more rigorous stopword removal procedure we could provide a cleaner input to the clustering phase, thereby increasing the accuracy of the results obtained.

6. DISCUSSION AND CONCLUSIONS

Traditional document clustering techniques have been applied to group similar documents and identify topics where the input consists of newspaper stories, or streaming audio or video content. However, no such methods have been applied to detect and extract threads in textual chat data. Furthermore, a prominent problem faced in clustering chat data is that the utterances themselves are very small; hence the information available for clustering is also extremely limited. As such, reliance on the clustering algorithms alone to identify threads is unrealistic. In order to overcome these problems, we devised several methods to process both the input and output so that the threads are meaningful.

Our first concern is to ensure that we provide a meaningful set of data for the clustering algorithms to work on. This was ensured, by applying stopword processing to the input data, eliminating words that do not convey any particular information. This input data is then operated upon by various clustering algorithms, and we evaluated a variety of clustering algorithms and criterion functions to determine which were better suited to processing chat data than others. We ended up selecting "RB", and "H2" as our combination of clustering algorithm and clustering criterion function. We also evaluate several clustering quality metrics to see which one was best suited for giving the closest approximation to the actual number of topics in a chat session for which the exact number of topics is not known *a priori*. Based on the results of the experiments conducted, we selected Metric 6, Section 4.3.2.2.6.

The result of this combination is then used as a starting point for our post processing. This last phase is very important to improve the accuracy of the thread

extraction. In our experiments, post-processing increased the accuracy from 0.86 to 1, an increase of 14%.

During the course of this project, we have shown a method of using document clustering as a means to identify and group together chat utterances that are similar in content. We also showed that the temporal information about the chat utterances is vital, and incorporating that information greatly improves the threads detected. Therefore, the content of a chat utterance as well as the time at which it occurred during the course of the chat session, are both important.

7. Future Work

While my project aims to provide a method which could be used to group together chat utterances that are similar, it is by no means comprehensive. There are several enhancements that could be made in order to make this approach work better. The stopword removal could be improved to yield much cleaner input. Furthermore, given the extremely limited amount of information available in each utterance, we could employ query expansion in order to enhance the quality of the input available to the clustering algorithms. This could be one of two forms: either use a dictionary to expand each term individually, or group together two or more chat utterances to form one single utterance. For the clustering algorithms itself, we could experiment with all the permutations and combinations available from the CLUTO package, or perhaps even another similar package, to find out which combination works best for clustering chat data.

Several improvements could be made to the post-processing phase as well, in order to improve the overall thread quality. This could be such as running a final check on the overall results (in batches of 10 utterances or another such suitable number), and obtain the cluster ID which occurs most often in that particular batch. Now, if there are some clustering IDs that occur just once in that batch, then that is an indication that they have been clustered incorrectly, and as a best effort to assign the correct cluster ID to them, we might assign them the ID that has occurred most frequently in that batch.

Also, we could use the usernames. We chose not to for our synthetic threads because they are too strong a clue when the data comes from different rooms. However, for real chat, this may help.

8. References

[1] BioNet Systems, 2003. http://www.bionetsystems.com/press/bionet-09SEP2003-press.pdf, "Introducing Net Nanny's New, Revolutoinary Chat Monitoring Program", September 9, 2003 Bellevue, Washington.

[2] Bjorner Larsen and Chinatsu Aone, "Fast and Effective Text Mining Using Linear-time Document Clustering", *KDD-99, San Diego, California, 1999.*

[3] C. J. van Rijsbergen, Information Retrieval, Butterworths, London, 2nd ed., 1979.

[4] Christopher D. Manning and Hinrich Schütze, 2001. "Foundations of Statistical Natural Language Processing", 2001

[5] CLUTO, 2003. http://www-users.cs.umn.edu/~karypis/cluto/index.html, "CLUTO version 2.1.1, Software Package for Clustering High-Dimensional Datasets", November 2003.

[6] D. R. Cutting, D. R. Karger, J. O. Pedersen and J. W. Tukey, 1992. "Scatter/Gather: a cluster-based approach to browsing large document collections", *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318-29, 1992.

[7] Daniel Barbará, Julia Couto, Yi Li, 2002. "COOLCAT: An entropy-based algorithm for categorical clustering", *CIKM '02, November 4-9, 2002, McLean, VA, USA.*

[8] David A. Hull, "The TREC-7 Filtering Track: Description and Analysis"

[9] E. M. Voorhees. 1986. "Implementing agglomerative hierarchical clustering algorithms for use in document retrieval", *Information Processing and Management*, 22:465-76, 1986

[10] E. Rasmussen. "Clustering Algorithms". In W. B. Frakes and R. Baeza-Yates (eds.), *Information Retrieval*, pages 419-42. Prentice Hall, Eaglewood Cliffs, N. J., 1992.

[11] George Karypis 2003. "CLUTO* A Clustering Toolkit", University of Minnesota, Department of Computer Science, Technical Report: #02-017, November 28, 2003

[12] H. Schütze and C. Silverstein, 1997. "Projections for efficient document clustering." In *Proceedings of the 20th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74-81, 1997.

[13] IamBigBrother.com, 2004. http://www.iambigbrother.com., February 2004.

[14] Instant Messengers, 2001. http://www.instant-messengers.com/site/news/im_more_popular_than_ever.htm, "Instant Messaging More Popular Than Ever at Work", 2001.

[15] InternetSafetySoftware.com, 2004. http://www.internetsafetysoftware.com/spyformsnmessenger/, February 2004.

[16] J. J. Rocchio, 1966. "Document retrieval systems - optimization and evaluation". Ph.D. Thesis, Harvard University, 1966.

[17] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang, 1998. "Topic Detection and Tracking Pilot Study Final Report", *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, February, 1998.

[18] James Allen, Ron Papka and Victor Lavrenko, 1998. "On-line New Event Detection and Tracking", *Proceedings of the 21st ACM-SIGIR International Conference on Research and Development in Information Retrieval, Melbourne, Australia, August 1998.*

[19] James Allen, Victor Lavrenko and Ron Papka, 1998. "Event Tracking", University of Massachusetts, Computer Science Department, CIIR Technical report IR-128, January 1998

[20] Jason Bengel, Susan Gauch, Rajan Vijayaraghavan, Solomon Nagelli, 2003. "Archiving and Indexing Chat Utterances", Department of Electrical Engineering and Computer Science and Information Technology and Telecommunications Center University of Kansas, 2003.

[21] Juha Makkonen and Helena Ahonen–Myka, 2003. "Utilizing Temporal Information in Topic Detection and Tracking", 08-19-2003

[22] Leuski, A. 2001. "Evaluating Document Clustering for Interactive Information Retrieval", *CIKM'01, November 5-10, 2001, Atlanta, Georgia, USA.*

[23] Oren Zamir and Oren Etzioni, 1998. "Web Document Clustering: A Feasibility Demonstration", *SIGIR'98, Melbourne, Australia*

[24] *Proceedings of the TDT Workshop*, University of Maryland, College Park, MD, October 1997.

[25] Rajan Vijayaraghavan, 2003. "An Architecture for Logging and Searching Chat Messages", Master Thesis Report, Department of Electrical Engineering and Computer Science, University of Kansas, April 28, 2003

[26] Steinbach, M., Karypis, G. and Kumar V. 2000. "A Comparison of Document Clustering Techniques", *TextMining Workshop, KDD, 2000*

[27] Tech Talk. 2000. http://www.ces.ncsu.edu/depts/it/itaids/news/00-06/6.shtml., "Instant Messaging", June 2000

[28] Vasileios Hatzivassiloglou, Luis Gravano and Ankineedu Maganti, 2000, "An Investigation of Linguistic Features and Clustering Algorithms for Topical Document Clustering", *SIGIR 2000 7/00 Athens, Greece*

[29] Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Stephen Thomas, 2002. "Relevance Models for Topic Detection and Tracking", 2002.

[30] Ying Zhao and George Karypis, 2002. "Criterion Functions for Document Clustering: Experiments and Analysis" University of Minnesota, Department of Computer Science / Army HPC Research, TR# 01-40, February 21, 2002

[31] Ying Zhao and George Karypis, 2002. "Evaluation of Hierarchical Clustering Algorithms for Document Datasets", *CIKM 2002, November 4–9, 2002, McLean, Virginia, USA.*

[32] Ying Zhao and George Karypis, 2003. "Hierarchical Clustering Algorithms for Document Datasets", Department of Computer Science, University of Minnesota, Minneapolis, Technical Report #03-027 (extended version of the CIKM 2002 paper)

[33] Yitong Wang and Masaru Kitsuregawa, 2002. "Evaluating Contents-Link Coupled Web Page Clustering for Web Search Results",*CIKM'02, November 4-9, 2002, McLean, Virginia, USA.*