An Architecture for Logging and Searching Chat Messages

By

Rajan Vijayaraghavan

Bachelor of Engineering

Electrical and Electronics Engineering

University of Madras, India, 1999

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

_____

Professor in charge

_____

_____

Committee Members

_____

Date thesis accepted

Acknowledgements

First of all, I would like to thank my academic advisor and committee chair, Dr. Susan Gauch, for guiding me through the Thesis work. I thank Dr. Arvin Agah and Dr. Joseph Evans for being in my Thesis committee.

I would also like to thank my friends and especially Solomon, Subhash for their help. I extend thanks to my other friends for making this study of mine quite an interesting experience.

Abstract

Instant Messaging has become an important means of communication between people around the globe, providing an alternative to telephone and email conversations. The number of users using Instant Messenger products has been increasing over recent years. Messenger services provide a perfect environment for private and personal chat. People of all ages log into messenger service or chat rooms to spend time chatting with known and unknown persons. As a form of entertainment for children, there may be danger lurking from adults chatting with teens, camouflaged as teens. This has created a desire to monitor chat activity on a system.

The goals of this thesis are to provide an archival system for a chat client, an incremental indexing system and a retrieval system that supports a variety of queries. The archival system is designed to be independent of the messaging software so it can be adopted more widely. The indexing and retrieval systems are portable to both Windows and Linux with minimal changes and the system is to be scalable to continuous operation.

Chat sessions can be archived on either the server side (chat rooms) or client side (Instant Messengers). This thesis has focused on client applications, but the development has been coordinated with a related server side project. One of the popular chat clients, Microsoft MSN Messenger, has been chosen to provide the text streams for archiving. The text is indexed and a search capability is incorporated. The retrieval system can support queries based on keyword, username, date, and other criteria.

Contents

List of Figures

1. Introduction

1.1. Motivation

One of the developments in Web technology is instant messaging, the process of instant communication between people. More and more people, of all ages, have started using instant messaging software and the numbers are continuing to grow. Prominent Web sites have chat rooms that host groups that exchange messages and billions of messages are being sent through the chat rooms every month [1].

Chat rooms provide server-based instant messaging in which people log in to one of the chat rooms to spend time. Yahoo and Microsoft provide two prominent web sites that have many chat rooms each grouping users by culture, location, etc. Many people log into the chat rooms and converse for hours, staying in a single chat room or migrating from one room to another or chatting simultaneously in many chat rooms. One of the advantages of chat rooms is that it is possible to chat with many people at the same time. This capability can also be viewed as a disadvantage because the conversation is open to all the people in that chat room. Messages with personal content are visible to all the persons involved in a chat room. To overcome this disadvantage, client-based instant messaging applications have arisen. People who want privacy in a conversation, or who want to talk with known users only, use instant messengers.

There are a number of instant messaging (IM) products in the market. Some of the prominent ones are Microsoft® MSN Messenger, Yahoo's Yahoo Messenger and America Online's AOL Instant Messenger. Although designed to provide one-on-one chat, a messaging conference between multiple users is also possible. MSN Messenger [2] allows up to 10 individual conversations simultaneously and also supports

conferences with a maximum of 4 people in the same conversation window.

Use of instant messaging has extended its reach into areas other than recreation. Corporate environments are turning to instant messaging to communicate co-workers more quickly and important internal office details or technical plans are discussed using chat sessions. Slowly, the corporate world has started reaping the benefit of using instant messaging software.

It is projected that by the year 2008, there will be an approximately US $8 billion market for instant messaging products. Currently, the number of users in corporate environment has surpassed 40 million [3]. In excess of 250 million users are predicted to use instant messaging by the year 2005. By 2006, instant messaging is expected to be the preferred medium of communication, overtaking email.

Chat rooms are also used to disseminate and share knowledge. At the University of Florida, one section of a course has the students use chat rooms for their weekly discussions, replacing the traditional physical discussion section. At the Mozilla development center, the programmers use IRC clients to keep others informed of code sharing, coordinating the checking in and out the source code in the source code control software. US Marines [4] have started using instant messaging software to keep in touch with their colleagues over around the world.

In general, chat rooms have become a place where nearly all-possible topics are discussed [5]. People share news and views of events and it is also used a mechanism for disseminating important information to a small group or the public at large. However, the usability and versatility of the instant messengers has lead to several important issues, as described below.

1.2. Parental Controls

Most of the instant messengers except Yahoo Messenger Enterprise Edition are free of charge. This has allowed people of different ages to converse in messengers and chat rooms, and the popularity of messenger services is growing. However, there are concerns whenever young people chat with unknown people. They might not know anything about the individual with whom they are chatting. There can be danger lurking from unscrupulous people chatting with teens, camouflaging as a teen, and extracting personal information from the teen. There had been situations when young people had been lured to reveal personal information about them and get into situations that are dangerous [6][7][8][9].

These possibilities have led to constant worry for parents whose children frequent the web to chat with people. Parents need way to monitor their children's activity when they are logged on to the computer. Currently, parents are able to list the sites visited by their children and to log all the conversations involving the child without their notice. These methods are referred to as chat monitoring or website monitoring. In some cases, products provide message filtering to prevent the children getting exposed to foul language.

On the server side, sites enable centralized text filtering. In MSN Chat rooms, for example, a user is removed from a room if he violates the rules that regulate conversations in the room. However, if a user is dropped from a room, they are allowed back immediately by merely logging into the same chat room again. So, safe chat features are not really effective in MSN chat rooms.

Yahoo chat rooms, in contrast, do not provide any safe chat procedures. In Yahoo

Messenger there is an option to filter text based on the words uttered by the chatting people. There are three options for word filtering in Yahoo: none; weak; and strong. Yahoo maintains a file (C:\Program Files\Yahoo!\Messenger\filter1.txt), which contains approximately 60 comma separated words that are removed when encountered in the conversation text.

1.3. Need for message archiving

With the projected increase in the number of chat users and the variety of uses for instant messaging, there arises a need for archives accessible to end users, a capability not provided by existing software. The conversation is lost after the conversation window is closed either deliberately or accidentally. Any important discussion or issue resolved during the chat session vanishes without a trace.

Some of the messenger services have a rudimentary chat-logging feature, but many do not. Microsoft® MSN Messenger, for example, has no message archiving ability. Yahoo Messenger, however, has message archiving ability, but it is not readable. The messages are stored in format that only Yahoo software can read. To provide parents a way to review their children's online chats or to allow messages to be reviewed to allow corporations to review important decisions, we plan to provide an archiving system for instant messages.

1.4. Search for keywords

The goals of my research are to provide a functional archiving and search feature for Microsoft MSN Messenger. The system must also be designed to be independent of the messaging software so it could be adopted more widely.

The existing problem with messenger services is that the conversation text is lost when the window is closed. So, any important information that was conveyed or received over the conversation is lost. In the corporate environment, many messages pass through the messengers. Some messages may be courtesy messages, some of them may be important decisions, and some may be important financial details. The messages logged are not searchable in most of the systems. This is due to the fact the messages are not archived and lost for good when the window is closed. So, the need for archiving arises to keep track of all conversations involving a particular user.

Searching un-indexed text is very slow task. All the files have to be opened to find out whether or not it contains the words of interest. If the user is a frequent user of messenger service, his/her task of locating particular messages will be in no way different from searching for a needle in the haystack. This brings into scene the need for providing search functionality along with the archiving ability.

1.5. Scalability of the system

Since, we store each message as a unique file, message archiving, causes many files to be created. If a user chats 5 hours a day and is sending 20 distinct messages per minute while receiving the same number of messages per minute, the archive will create 12,000 files per day on the user machine. If a user chats with more than one person at a time, the number of files does not increase since there is a natural limit to how much text one individual can see or produce. If the outdated files are cleaned up periodically, the disk space is not an issue. The files tend to be very short, so even 12,000 files at approximately 100 bytes each would only require 1,200,000 bytes or 1.2 MB.

2. Related Work and Existing Systems

2.1. Commercial Systems:

There are a quite number of commercial systems that monitor the sites visited and also store the chat sessions on a personal computer. They provide a variety of functionalities such as grabbing text from messenger service(s) and getting window titles of all web pages open on a system. Most of the systems provide a similar set of capabilities. Some commercial products are discussed below.

2.1.1. Iambigbrother

Iambigbrother [10] is a popular archiving system that is one of the most comprehensive monitoring products on the market. It logs the titles of all web pages visited on a personal computer system. It also captures all the text typed on the system and keeps track of the times at which conversations happen. The software has the ability to play back chat conversations based on user name 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.

2.1.2. Net Nanny

Net Nanny [11] is a less complete monitoring system, that does provide any search feature at all. It contains a database of web pages that are categorized by the ages of users for whom the content is appropriate. This database is used to filter the context that a user can view based on their age. The company has a research team that continually visits websites and updates the database. Net Nanny can log the URLs of

web pages visited and chat transcripts. The software is able to playback the text of the logged conversations. It displays the user name and time of the conversation as the heading.

2.1.3. Cyber-Snoop

Cyber-Snoop [12] is a product that can capture text from various messenger systems, e.g., Microsoft® MSN messenger, Yahoo Messenger and AOL Messenger. It can display the text captured, but lacks any ability to search the archive. It merely captures all window context, ignoring the content portion of email and logging only the header.

2.1.4. Desktop Snooper

Desktop Snooper [13] is very similar to Cyber-Snoop in that it too logs chat room activity, but provides no search feature. It will also store web page addresses visited.

2.1.5. I-Spy Now [14]

This is a chat conversations logging product that does the monitoring of chat conversations. It logs both sides of all chat conversations for AOL/ICQ/MSN/AIM Instant Messengers, and views them in real time, as they are happening.

2.1.6. Pearl Echo Internet Monitoring Software

Pearl Echo Internet Monitoring Software [15] differs from the previously discussed systems in that the logged files are stored on the server rather than on the individual client machines. Thus, this product is suitable for corporate purposes that require a centralized archive for reliability and/or to monitor a wide variety of users or

chat rooms. The system logs all chat conversations, the websites visited, and also keeps track of the system usage by personnel in an office environment.

2.1.7. Yahoo Messenger

Yahoo Messenger is one of the most prominent instant messaging programs that provides instant messaging. It has a built in message archiving ability based on the user sign-in name and the conference. It stores the transcripts for the number of days specified by the user. The logged chat transcripts are stored on the users local machine. Since the messages are encoded, the text logged is only readable by Yahoo Messenger itself. Although Yahoo creates an archive of chat conversation text, these are not searchable. They can only be replayed. The files say about who talked with whom, time of the chat.

Yahoo Messenger can provide safe chat procedures by maintaining variants of inappropriate language in a file (C:\Program Files\Yahoo!\Messenger\filter1.txt). Yahoo messenger provides three levels for filtering the text: 1) low, 2) medium and 3) high. Low means no filtering of language and high means strict language filtering.

2.1.8. Spector Pro

Spector Pro [16] records AOL chat rooms, AOL Instant Messenger, MSN Messenger, and Yahoo Messenger. It has a chronological listing of the archived files. Printable transcripts can be made with this software.

2.1.9. SpyBuddy 1.9 [17]

Spy Buddy is one of the spy software products that is very similar to I-Spy now. It can capture conversations from AOL/MSN/IRC/AIM messengers. It stores screen

shots of the desktop at preset intervals. There is no search feature available with the software. It also captures websites activity.

2.1.10. Keyboard Monitor Keylogger 3.0 [18]

This is another software product that captures text and encrypts it. Only the native software can decrypt the recorded files. The program provides the ability to capture all keyboard activity and has no search feature at all.

System	Archival	Search By keyword	Search By user name	Search by date	Keyword + user name (who said)	Keyword (who said)
IamBigBrother	Yes	Yes	No	No	No	No
Yahoo Messenger	Yes	No	No	No	No	No
Net Nanny	Yes	No	No	No	No	No
Cyber-Snoop	Yes	No	No	No	No	No
Desktop-Snooper	Yes	No	No	No	No	No
I-Spy Now	Yes	No	No	No	No	No
Pearl Echo Internet Monitoring Software	Yes	No	No	No	No	No
Spector Pro	Yes	No	No	No	No	No
SpyBuddy 1.9	Yes	No	No	No	No	No
Keyboard Monitor Keylogger 3.0	Yes	No	No	No	No	No
Mozilla	Yes (S)	No	No	No	No	No
University of Florida	Yes (S)	No	No	No	No	No

Table 1: Comparison chart for different existing products.

(S) – Server side archival.

2.2. Information Retrieval

A typical traditional Information Retrieval (IR) [19][20] system consists of a collection of information items, indexing algorithm, and a retrieval function. There are

three models of traditional IR systems: Boolean model, probabilistic model and vector space model. In any IR system, the collection of information items are indexed and weights are stored in an inverted file.

In the Boolean model, queries are represented by terms that are joined together by logical connectives and search is based on exact match. That is, the documents that contain the terms that satisfy the logical function defined by the query will be retrieved.

The probabilistic model is based on the idea that given a document and a query, it should be possible to calculate that the document is relevant to the query.

In the vector space model, both queries and documents are represented by vectors. The relevance between the query and a document is determined by the similarity measure between them, such as cosine of the angle of the two vectors. It is based on the formulae

$$wt_{ij} = tf_{ij} * idf_i$$

where

wt_{ij} is the weight of term i in document j.

tf_{ij} is the term frequency of term i in document j.

idf_i is calculated as $log_2(N/n_i)$, where N is the total number of documents in the collection and n_i is the number of documents containing the term *i*. The retieval model used in our search system is based on the vector space model.

2.3. ChatTrack project

Chat Track is an ITTC KU project undertaken by Dr. Susan Gauch to provide security features to chat systems. The goals of the system are: 1) monitor the chat activity when a user chats in a chat room, 2) index and provide a detailed search

capability, e.g., search by keyword, search by speaker id, date based filtering of results, 3) implement safe chat procedures like tagging users who violate rules set forth by the chat system etc. The system is being produced for IRC chat systems. The security system that is being developed for a chat room system is extended to IM software. In this thesis, the IM software under investigation is Microsoft MSN Messenger.

3. System Architecture

This chapter discusses the system architecture. We begin by discussing the chat client, Microsoft® MSN Messenger, then describe the chat archiving client we have developed, focusing on the archival, indexing and retrieval process.

Microsoft® MSN Messenger is one of the popular instant messengers available in the market. Because there is no cost involved in using it and the user interface is easy to learn, more than 40 million users [2] around the globe use it. One of the reasons for its wide adoption is that it is incorporated with integrated with the Microsoft® Internet Explorer browser software, Microsoft Outlook® Express 5, and MSN services such as MSN Hotmail and MSN Mobile.

Microsoft® MSN messenger supports one-to-one chat or multi-person chat within a single window. There can be up to 10 instant messaging windows open at any time and conferences can a allow maximum of 4 unique users chatting in the same conversation window at the same time. Each message must be less than 400 characters long.

The important components of our system are the: the archival system, the indexing system, and the retrieval system. We will now discuss the three main modules of our ChatTrack project in the following sections.

3.1. Archival system

Archiving, or logging, is the process of the storing the conversation text into files. The system consists of a program that watches over the user's shoulder, constantly monitoring for any messenger activity on the system. Some of the features that the system supports are: 1) identifying when a new conversation window opens up; 2)

logging both incoming and outgoing messages; 3) checking for administrative messages generated by MSN messenger; 4) keeping track of users joining in a conference; 5) taking appropriate actions to active users and to detect when a user leaves the conversation or any of the users leaving a conference window; and 6) detecting window closing during conference sessions or one-on-one conversations. Microsoft requires that people users of its MSN messenger service have msn.com or hotmail.com email addresses.

The archival program checks for all currently running programs on the system. The presence of any MSN messenger conversation activity is detected if there is any window with title "<username/nickname>– Instant Message" in MSN messenger versions before 5.0 and "<username/nickname>– Conversation" in MSN Messenger 5.0. Here "<username/nickname>" is the user sign in name or the nickname which the user has chosen for the chat session.

If a messenger window is present, the program gets the window handle of the conversation window. The program then uses the window handle to generate a session id by converting it to an unsigned long integer. Since window handles are unique to each currently open window in the Windows operating system, this created unique identifiers for each session. If a window closes, the same handle may be used later. However, no two currently open windows will have the same window handle.

Using the conversation window handle, the program gets the window handle for the associated text area. In MSN Messenger, the handle for the text area can be obtained by requesting the window handle for "RichEdit2.0" class and then calling Windows API GetWindowText. Due to the flexibility offered by MSN messenger regarding nicknames, the program can also access the email id of the user in conversation with the local user.

This information allows us to identify the person.

The same APIs does not work for Yahoo Messenger because Yahoo Messenger has some built-in security features that prevent other applications grabbing messages using the above APIs.

New message(s) are gathered from, from the conversation window, in an incremental process. At any time, grabbing the text results is getting the entire text in the text area. Each time a message is grabbed, its length, including the current message, is compared with the length of the message grabbed previously. If there is a change in the message length it means there is a new message in the conversation window.

The initial message length is set to 94 characters. This value is the string length of the welcome message that MSN Messenger displays in the conversation window. If the current message length exceeds the previous message length, the next text is captured and broken into messages. The system is able to grab text but is not able to grab emotion icons, called emoticons, or any graphical images. Emoticons and/or images are treated as white spaces. Then, each message is parsed to separate the user name from the actual text. The archival program also captures the email sign in names. The window class "Edit2.0" gives access to user email id or nickname. Since the users may use either their email name or a nickname, we create a mapping between the user sign in name and any nicknames they use. When writing the messages into our archive, we write the corresponding user sign in name rather than the nickname so that all conversation from a particular user will be associated with each other. If the user has multiple email addresses, we will not be able to the user.

Messages captured from the conversation window may contain both user sent

messages and administrative messages. The administrative messages can be easily identified because they will not contain any user name. Administrative messages sent by MSN Messenger are also grabbed and those of interest are stored as though said by user "None".

The messages from each session are stored in a separate directory. The directory is names using a combination of the window handle and the date. For example, a chat conversation with sign in name say "subhashiv@hotmail.com" with the window handle as 912423, would be stored in directory "session000000912423/yy/mm/dd".

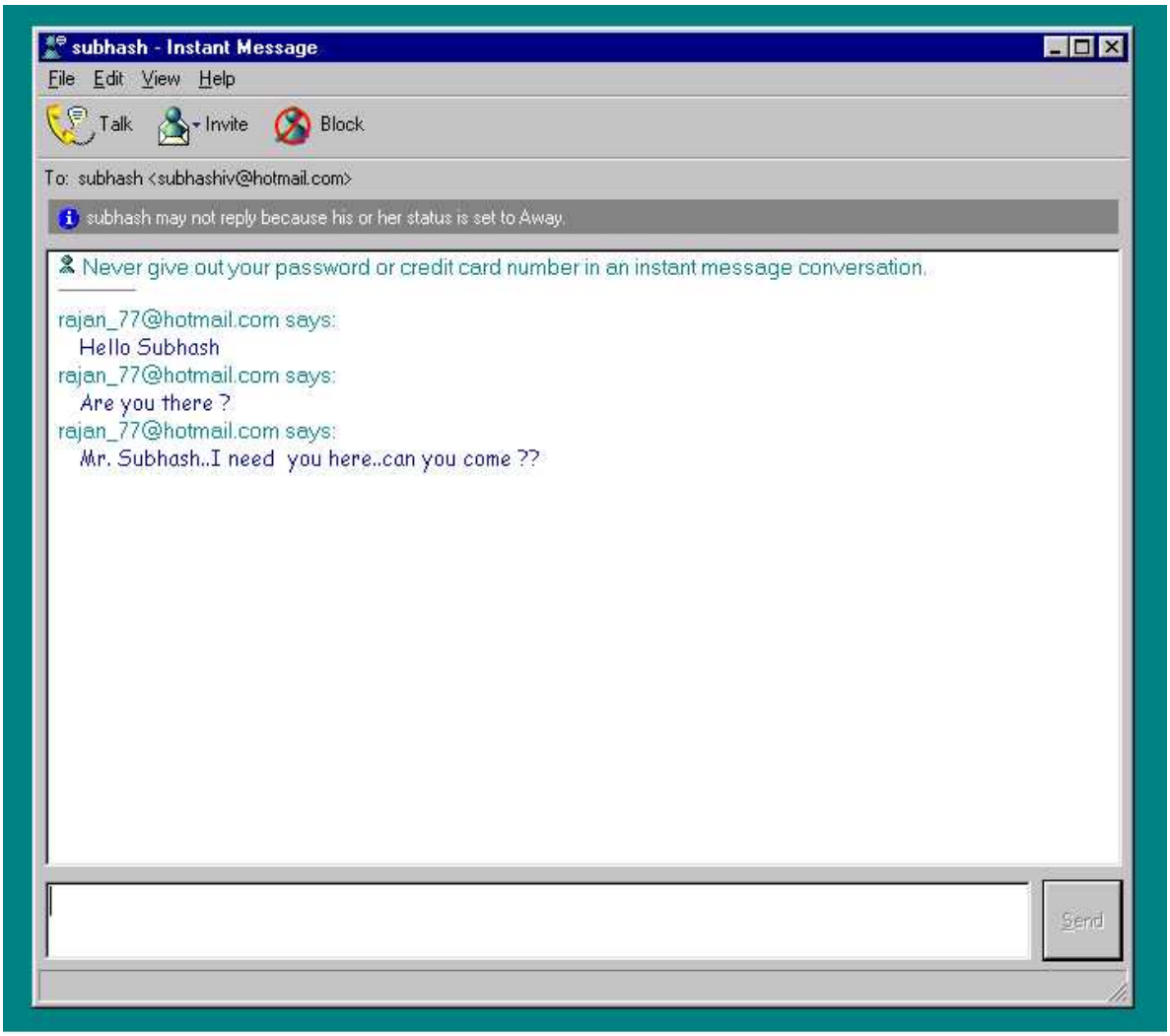


Figure 3.1: MSN messenger conversation window screen shot

where “yyyy” is the current year in four digit representation,

“mm” is the current month in 2 digit representation, and

“dd” is the current date in 2 digit representation.

Each message grabbed results in three files being created: 1) a message file; 2) a user information (speaker id) file that stores the email id of the user who said the message; and 3) a receiver file that contains user email ids of the users who would have heard the message.

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. We append “.uid” to the file that stores the user sign in name and “.mesg” for the text as said by the user. Finally, the file “hhmmssmsmsms.receiver” is created to store information about all the users who actually heard the message.

The archival program also creates a file called “FilesfromMessenger” and that lists the new files that were archived by the archiver. This file is used to tell the indexer which new files have been captured and thus to be indexed. The fields in this file are, in order, sessionid, year, month, date, and timestamp. The archival program periodically renames the file “FilesfromMessenger” to “toIndex” and calls the indexing program.

Currently, the archival program can monitor up to 100 different MSN Messenger windows at one time. Since a user cannot actually chat with more than 10 people at the same time with our current chat client, this limit is larger than is practically necessary.

The number of files created by the logging is an issue to concern. Assuming a user chats 5 hours a day, sending 20 unique conversation texts and receiving 20 conversation texts per window in a minute, then for a day the number of files created would be 72,000 on the local file system. In a year, the number of files would be 26,280,000. This may be a problem because of the number of files may exceed the total number of files to be created on a local system. The advantage with Windows is that it allows existence of large number of files. If the local file system is NTFS, Windows allows up to 4,294,967,295 files. Cleaning up the file every few months periodically will help in keeping the number of files with in limits. If the user converses with more than

one person at a time, the increase in number of files will not be linear.

There are two approaches in storing the captured files. One is to store all the files in the local system. Another approach would be to store the messages on a remote machine. The second idea is nearly identical to a server side chat logging. The main advantage of storing the files in the local machine is that the privacy and security of the messages is retained. In the remote machine archiving concept, if the remote machine is hacked into, then the all the messages are out in the open. This is potentially possible because intelligent hackers always trying to hack into some machine all the time. In the corporate environment, if the remote machine is hacked into then there is a chance that the corporate plans may go out and eventually the company may end up having a business risk.

In the logging program, messages include the administrative messages generated by the Messenger service. The administrative messages sometime contain the user(s) joining or leaving the conversation. In that case, the session id, user sign in name, the event (join or leave), date, and timestamp are logged. There are other messages that appear quite frequently on a conversation window. Some of them are messages that indicate file transfer, or file transfer started, file transfer completed, message could not reach the recipient, etc. Some of the common administrative messages are in the appendix section. The sample appearance of file organization is shown in the figure 3.2.

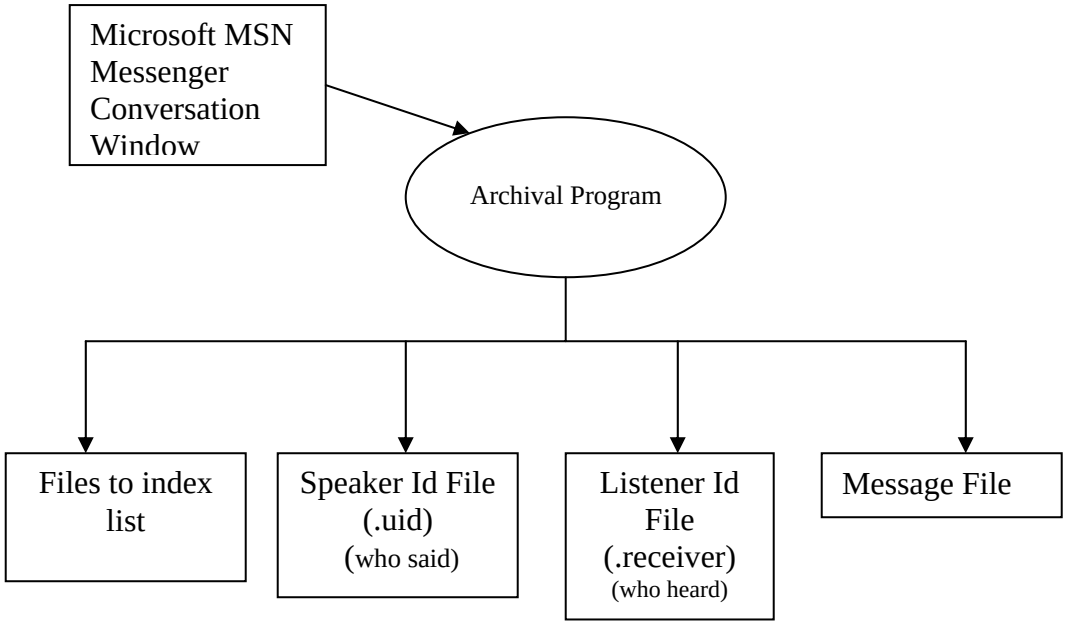


Figure 3.2: Archival System

3.2. Indexing System

Indexing is the process of creating an inverted file or index from the raw text files. An inverted file is a word-oriented mechanism for indexing a text collection in order to speed up searching. An inverted file is usually a collection of a dictionary file and a postings file. A typical Dictionary file contains one entry for each unique word in the document collection. Each record consists of the word itself, the total number of documents in which the word is present, total frequency of the word in the whole collection, inverse document frequency of the word and a pointer to the postings record for the word. The postings file contains information about a word in each of the documents in which it is present. Information in a posting record is: 1) frequency of the word in a document, 2) weight of the word in the document, 3) the document id for the

document, 4) a pointer to the next posting for the word.

For example, if the word ‘heart’ is present in 100 documents of the a collection of 200. It will have one record in the dictionary file. The inverse document frequency for the word is calculated as

Inverse document frequency (*idf*) = $log \left(\frac{Total\ Number\ Of\ Documents\ in\ the\ collection}{Number\ of\ documents\ containing\ the\ word} \right)$

Each of the posting records contains the frequency of the word ‘heart’ in that document and term weight of word in that document. The term weight of the word in a document j is calculated by

Term weight in document j $(w_j) = idf_{ij} * frequency\ of\ the\ word\ in\ document\ j$;

While retrieving for the word ‘heart’, the word is searched in the dictionary file first. If it is present, the postings list is accessed and the corresponding document ids are returned in decreasing order, sorted by weight.

3.2.1. Document Preprocessing

The inverted file is created from the words in the raw text files. Not all the tokens from the text files are added to the inverted file. Words that appear in almost all documents, or the most commonly used words (called stop list words), are omitted. Since they do not form a good query term, they are removed before a document is indexed. Single character tokens and numeric tokens are also not added to the inverted file. In our case, the raw files are tokenized into tokens and printed in a file. The file containing the tokens, devoid of the stop list words and other types of words mentioned above, is added to the inverted file.

3.2.2. Indexing process

The base indexing code for this thesis was written by Peter Whiting and later modified by Milan Gada. The code allows both incremental additions of documents to the inverted file and batch indexing (indexing all the documents in a single run). Most systems allow only batch indexing. Because the chat messages arrive all the time when a user chats, this thesis mainly uses the incremental indexing feature.

The indexing code produces a slightly modified inverted file. It produces a dictionary file, a posting file, and a documents file. The dictionary file consists of the word, number of documents that contains the word (number of postings record for this words), the inverse document frequency for the word, and a pointer to the postings file. The postings file consists of the document id for the document containing the word and two lists, a word list and a document list. The word list is the list of all postings to the same word and the document list is the list of all postings for the same document. One more file, called the documents file is created by this indexing system. This file consists of the normalization factor for a document and link to the first posting record for the document. All the above files use the concept of memory mapping the files. Memory mapping makes a file appear as a long array of records, so accessing the records is very fast

To provide more search functionality, the code has been modified to produce two sets of inverted files, one for keywords and the other one for the speaker email names.

3.2.3. Incremental Indexing

Incremental indexing is the process of building the inverted file in incremental process. It comes into picture only when there is an active conversation window. The

archival system, along with logging all the conversation, makes a list of files that have arrived newly in the Files-To-Index file. The list gives the files that need to be indexed. The indexing program takes this file and updates the inverted index. The indexer, after indexing, deletes the Files-To-Index file. The document-id file is also updated with the last document id allotted to the last file indexed. The creation/updating of the inverted index done upon demand as text arrives.

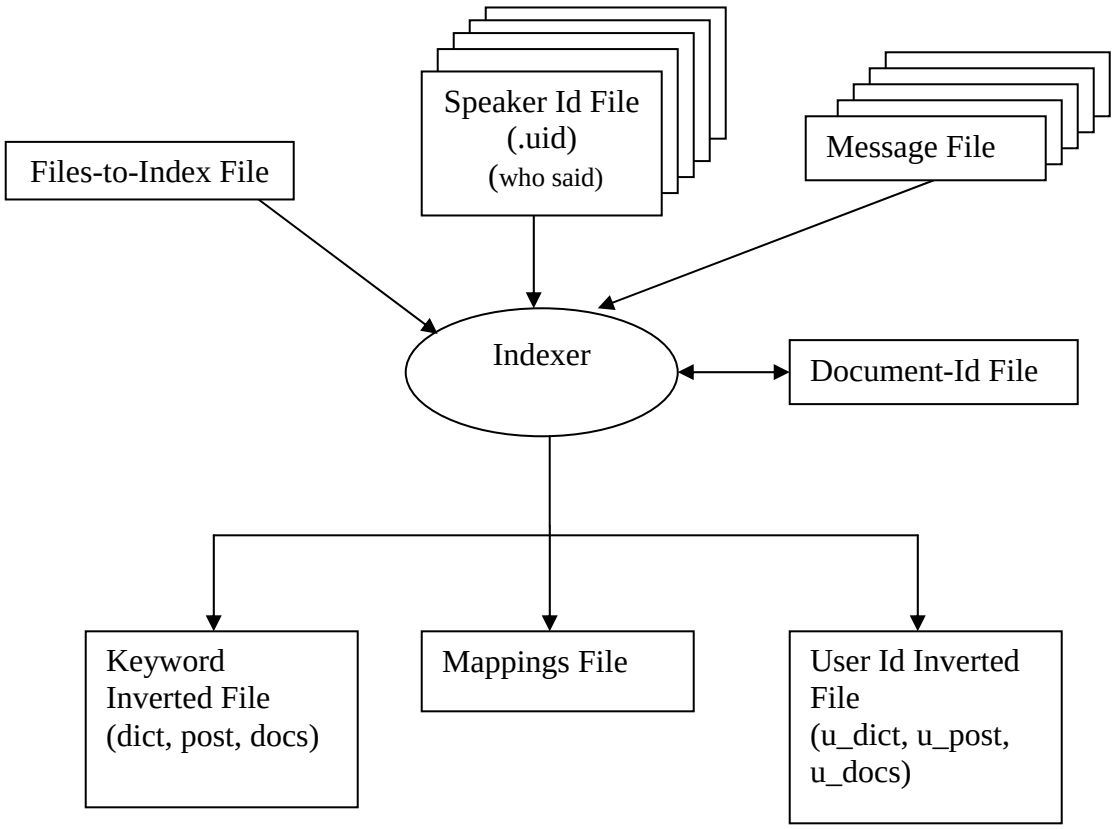


Figure 3.3: Incremental Indexing System

The above figure shows the incremental indexing program logic. The Files-To-Index list contains all the files that need to be added to the system. The Speaker-Id file

contains the user email name of the speaker of the text. The message file contains the conversation text. The information from the speaker id file go to speaker (User) Id Inverted File and the message file go to the keyword Inverted File. The mappings file contains information about the file location and the document id associated with a file. The mappings file is used by the retrieval program to display the text. It is created/updated by the indexing program as and when new files are indexed.

The Mappings file contains the document id for a file, session id, year, month, date (these are directory names) and timestamp (this is the filename). The retrieve program retrieves the document ids of the documents and this is used to get the information about the location of the file in coordination with the data from the Mappings File. The Document-Id is the unique identifier for each document. The Document-Id is used to solve this problem. At the start of indexing, it contains the value "0". The indexing program takes this as the initial document id. It allots document ids sequentially as new files arrive for indexing. The Mappings and the Document id files are updated from time to time due to the incremental indexing.

Currently, indexing has been set to occur approximately every 30 seconds. This enables searching the archive as early as 1 minute from the time the text was sent to the MSN Messenger window. The call to indexing program depends upon the availability of new files for indexing. The archival program calls the indexing program only when there are new files to be indexed.

The archival program is a background process with no user interaction as is the indexing program. The speed of indexing is very fast due to the use of memory-mapped files. Since the system is being developed along with the ChatTrack project, the indexing

and retrieve systems are shared by both the Windows (client) and Linux (server) platforms.

3.2.4. Batch Indexing

Batch indexing is the process of building the inverted index by indexing all the files at the same time. This type of indexing is used in this system to test the working of the system.

Batch indexing is a two-step process in this case. In the first step, it runs through the directories and gathers the file names and the locations. The names are added to files to index list file. The second step is the actual indexing of the gathered files. Before indexing, the Document Id file is set to zero.

3.3. Retrieval System

The retrieval engine is a dialog based Windows application. A screen shot of the retrieve dialog application is shown in the Figure 3.4. The system is able to search based on keyword, keyword and user name, keyword with date constraints, and keyword with date and user name.

The architecture for the retrieval system is shown in the following figure.

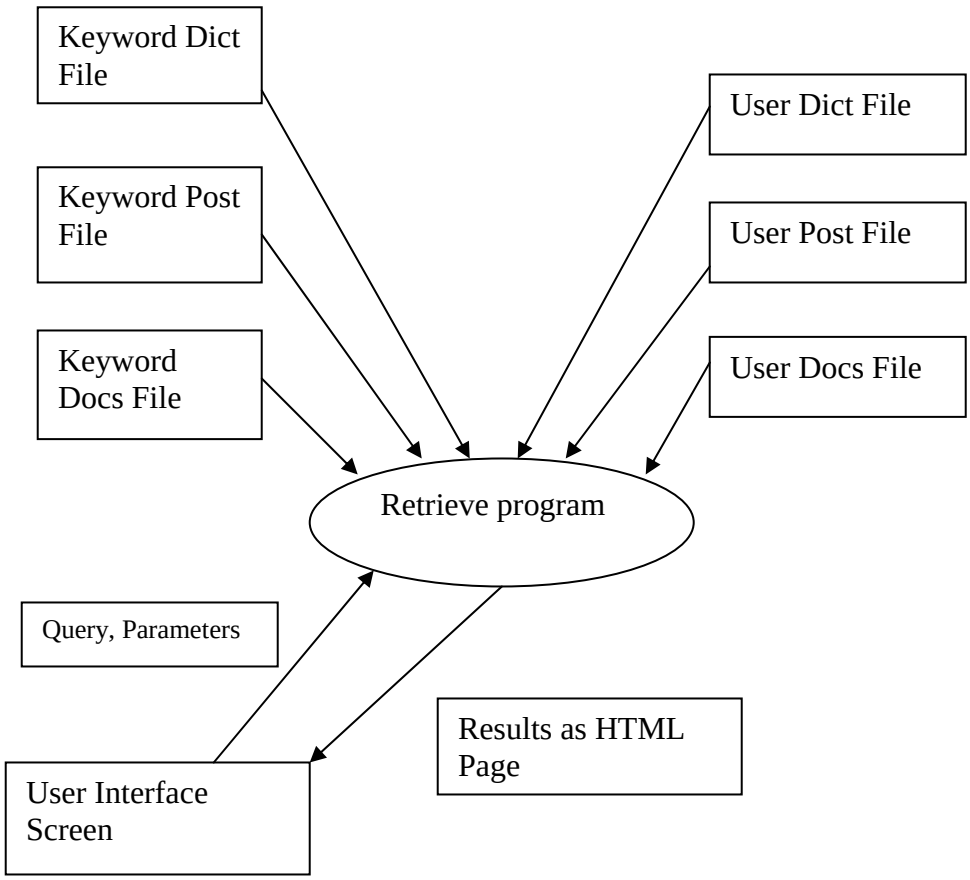


Figure 3.4: Search Flowchart

The figure does not show the various types of queries the system supports. The screen shots in the later pages show the different methods of querying and displaying results based on user preferences.

3.3.1. Queries Supported

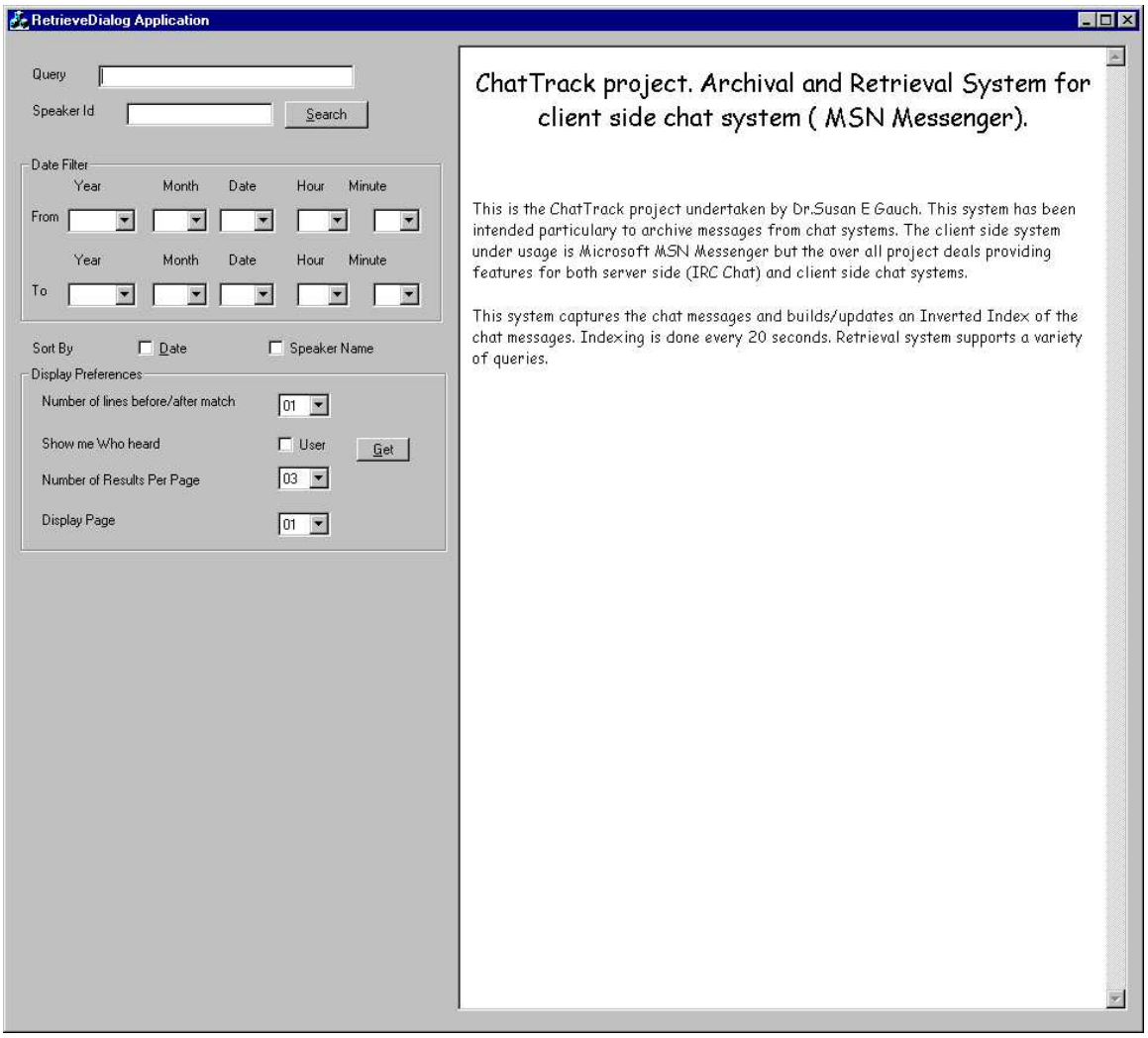


Figure 3.5: Screen Shot of the Retrieve User Interface.

Figure 3.5 shows a screen shot of the query window. Results appear in the right side of the screen.

As shown in figure 3.5, the user is able to query the archive on a variety of criteria and they are also able to control certain aspects of the results presentation. The user is able to query on keywords contained in messages. They are able to have these results

filtered by date and/or by the speakers MSN email id.

The types of date filtering available are

1. Show results after some date (context: the user mentioned “from” date parameter).

2. Show results after before date (context: the user mentioned “to” date parameter).

3. Show results between “from” and “to” dates (context: the user mentioned both “from” and “to” date parameters).

In the third case, the system checks that the user gives the ‘from’ date is earlier than the ‘to’ date. If they are reversed, the system re-orders them again and filters the results approximately.

Some of the presentation features incorporated to the system are: displaying results sorted by MSN email id (ascending order), displaying results sorted by time (most recent message on the top). The user can change the number of results he/she wishes to view on a single page of results, the number of conversation text he/she wants to see around the exact match. The system also highlights the message containing exact match so that it is easily identifiable and the system can also show the email ids of all users who listened to the conversation text. The last option is redundant for one-on-one conversations, but or multi user conferences, it is useful to know who all listened to the conversation.

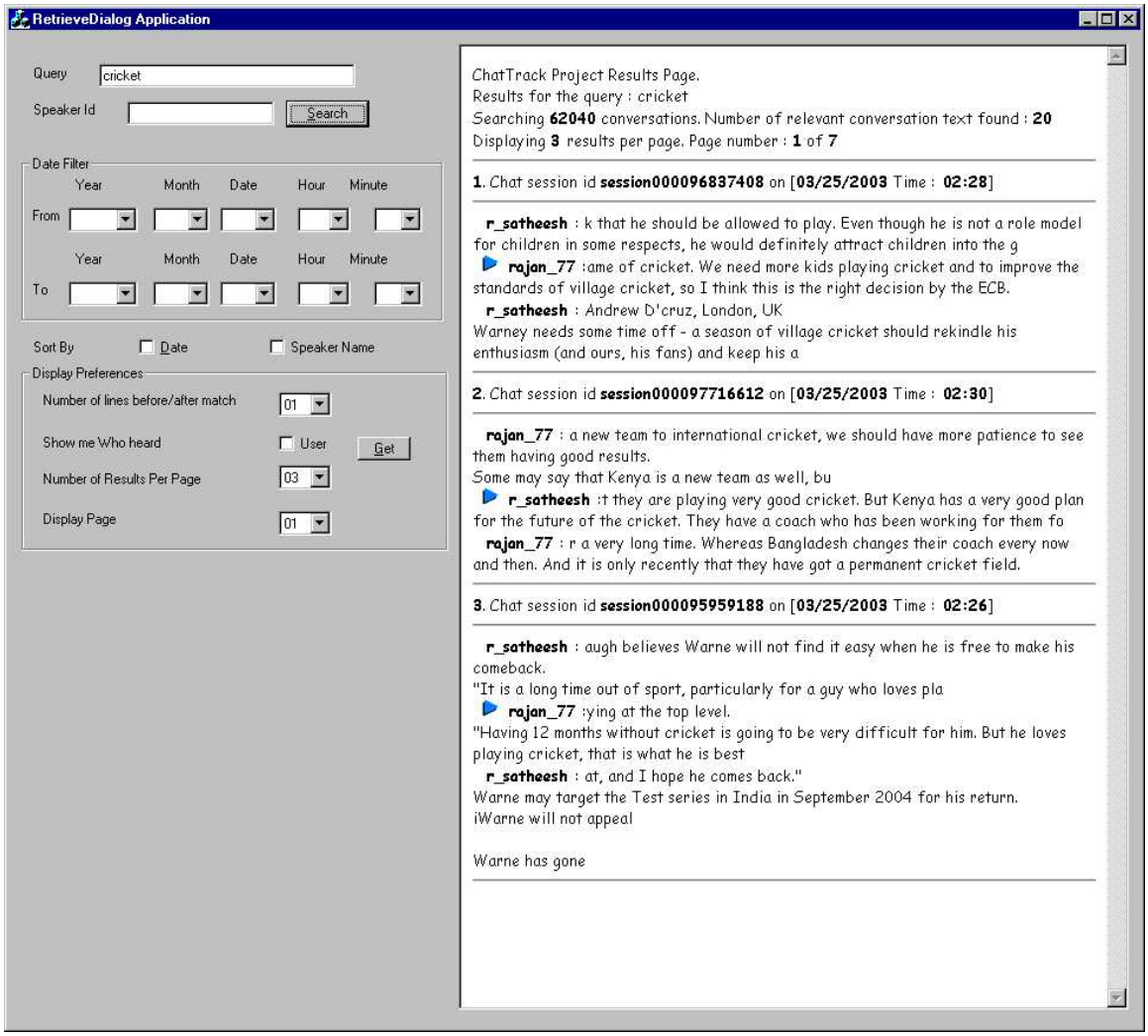


Figure 3.6: Screen Shot of sample results specifying only keyword as query parameter.

Figure 3.6 shows a screen shot of the results page for query “heart” on a collection of 62,040 messages collected. The results page shows some of the conversation text. It shows the keyword searched (‘cricket’), the number of conversation messages searched (62,040 in this case), the number of relevant conversation texts retrieved (20 for this keyword), the number of results being displayed (3 results per page) and the number of the page displayed (page 1of 7). Arrows indicate the location of the exact match.

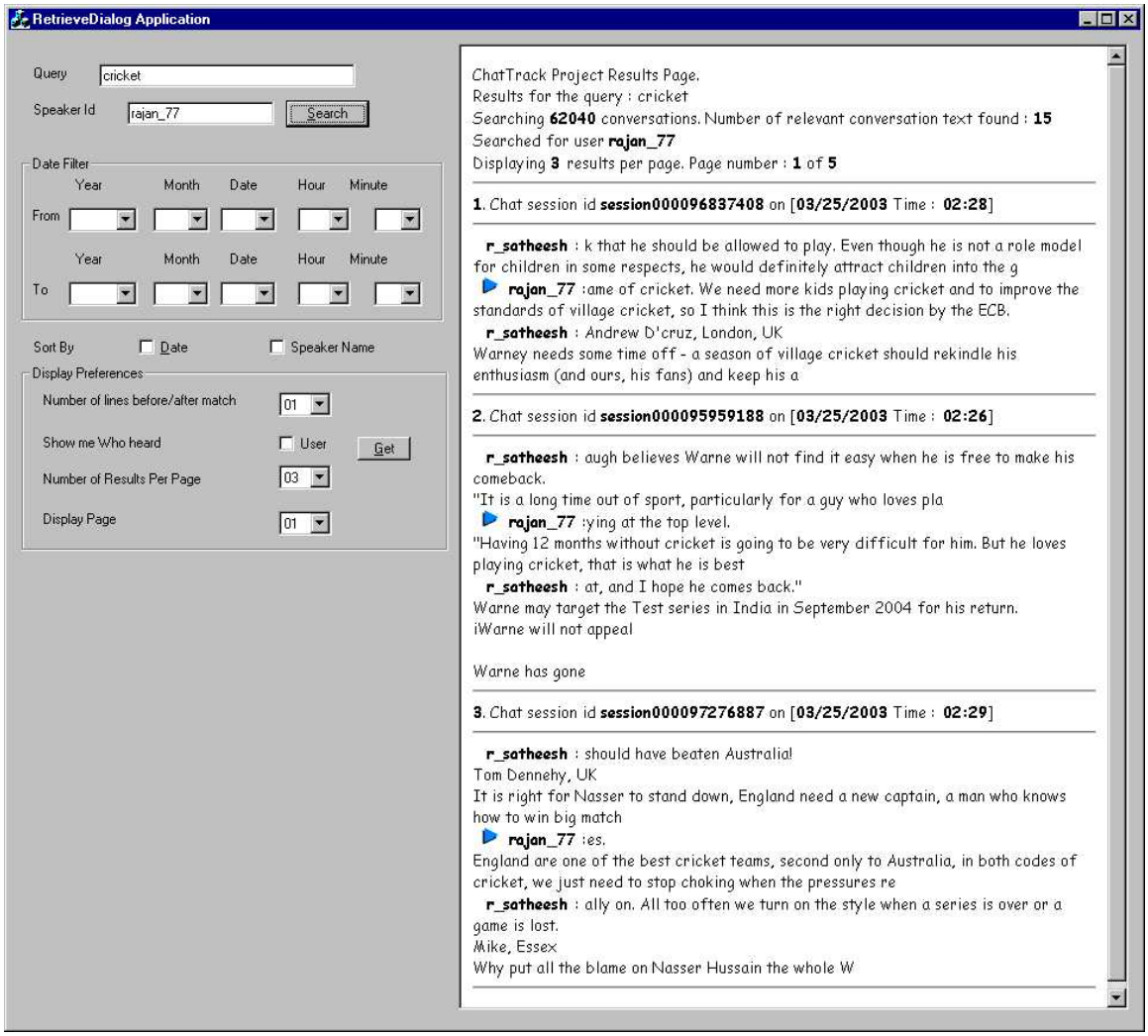


Figure 3.7: Screen Shot of the results screen with keyword and speaker email id as query parameters.

The next screen shot, Figure 3.7, shows the search capability by keyword and user name. The query requests for results containing the keyword “cricket” with email of the speaker “rajan_77”. The results page shows results that contain only text containing the keyword “cricket” said by speaker “rajan_77”, the top three of the 15 messages meeting this criteria are displayed.

Users can specify date options as one of the query parameters.

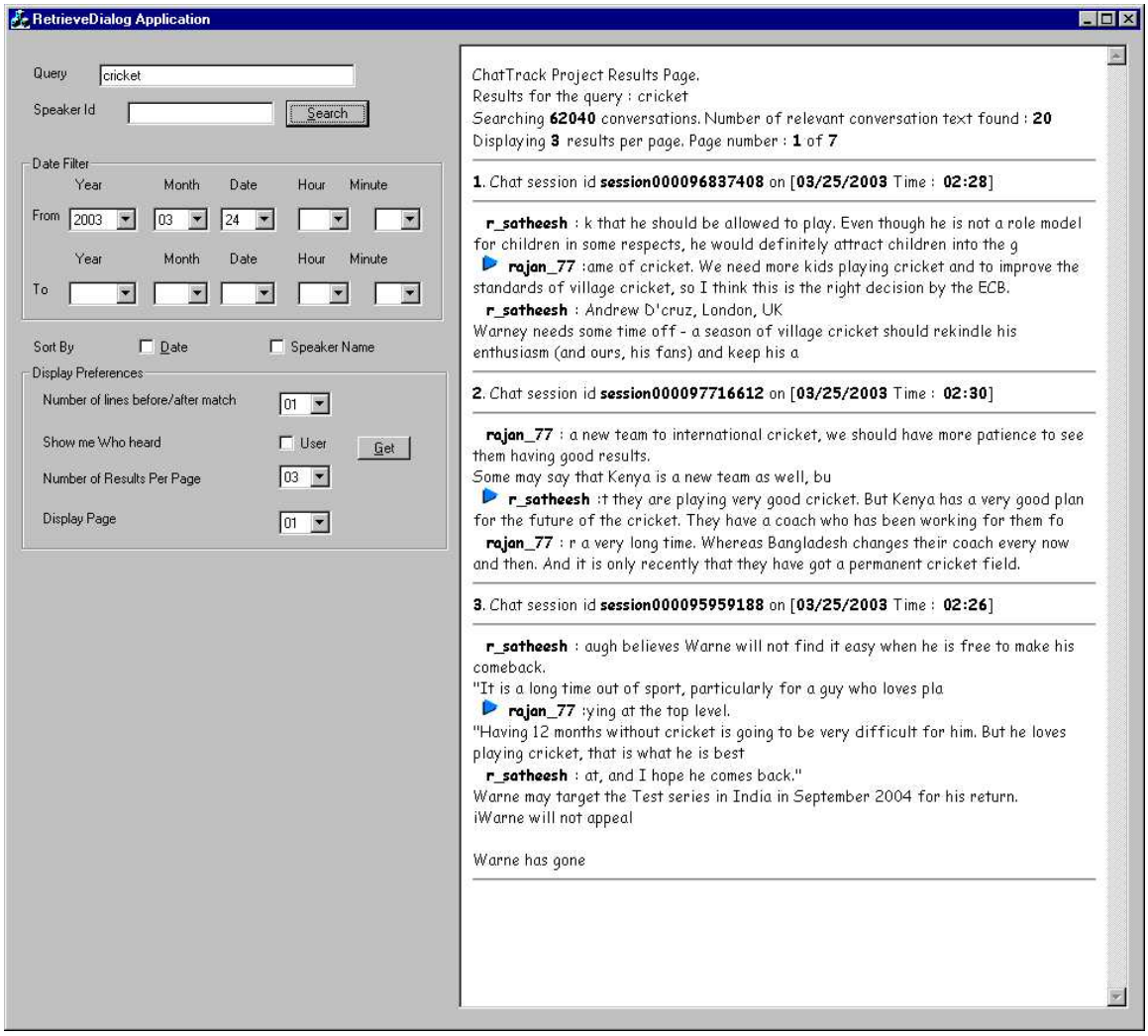


Figure 3.8: Screen Shot of the results screen with keyword uttered after some date ('from' date option)

If the user specifies only 'from' date with the keyword, the system displays results that appear after the specified date, see Figure 3.8. Results that are after March 24, 2003 are displayed. The number of lines before and after text is kept to one.

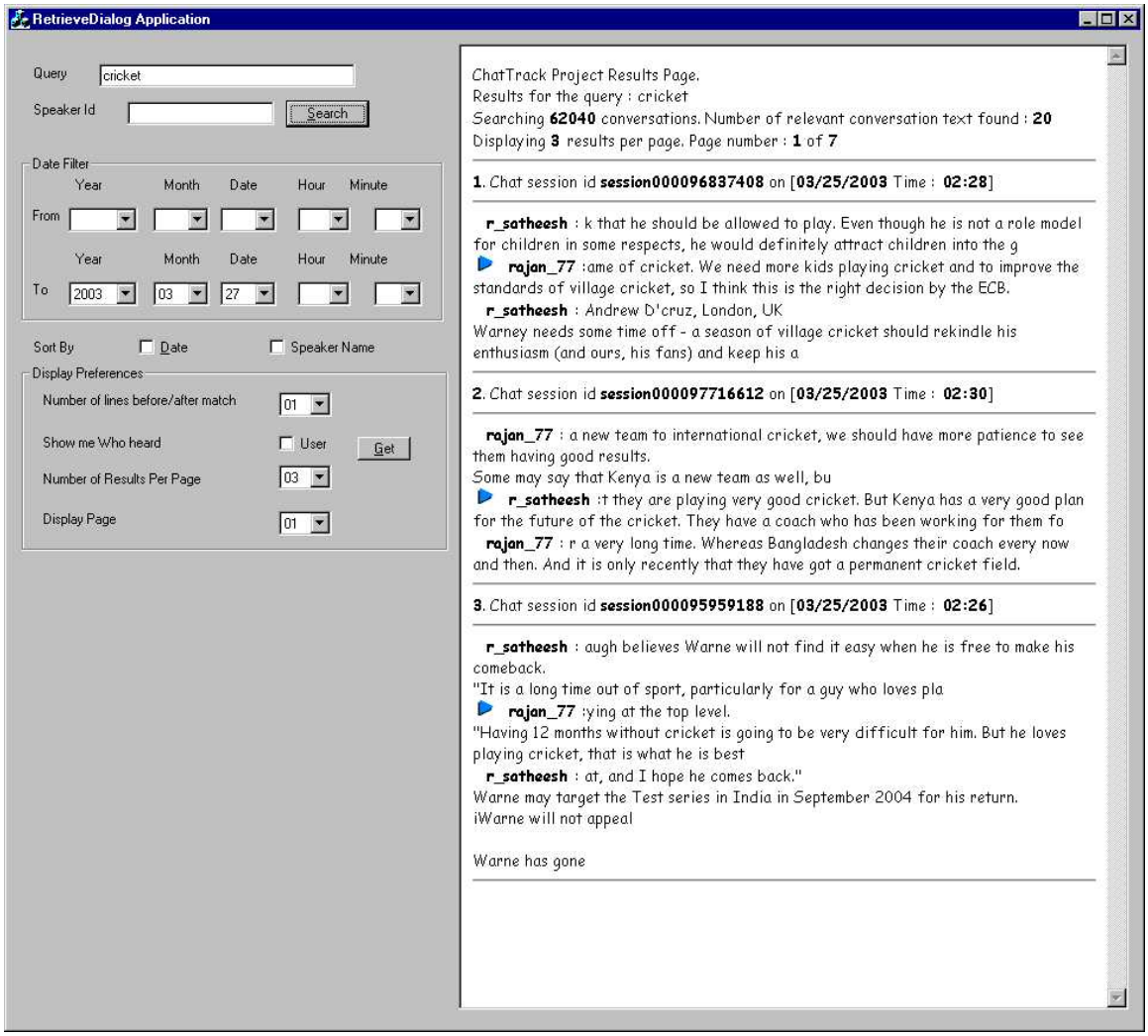


Figure 3.9: Screen Shot of the results screen with keyword uttered after some date ('to' date option).

If the user specifies only the 'to' date option with the keyword, the system displays results that appear before the specified date. From the results screen, see Figure 3.9, results that are before March 27, 2003 are displayed. The number of lines before and after text is kept to one.

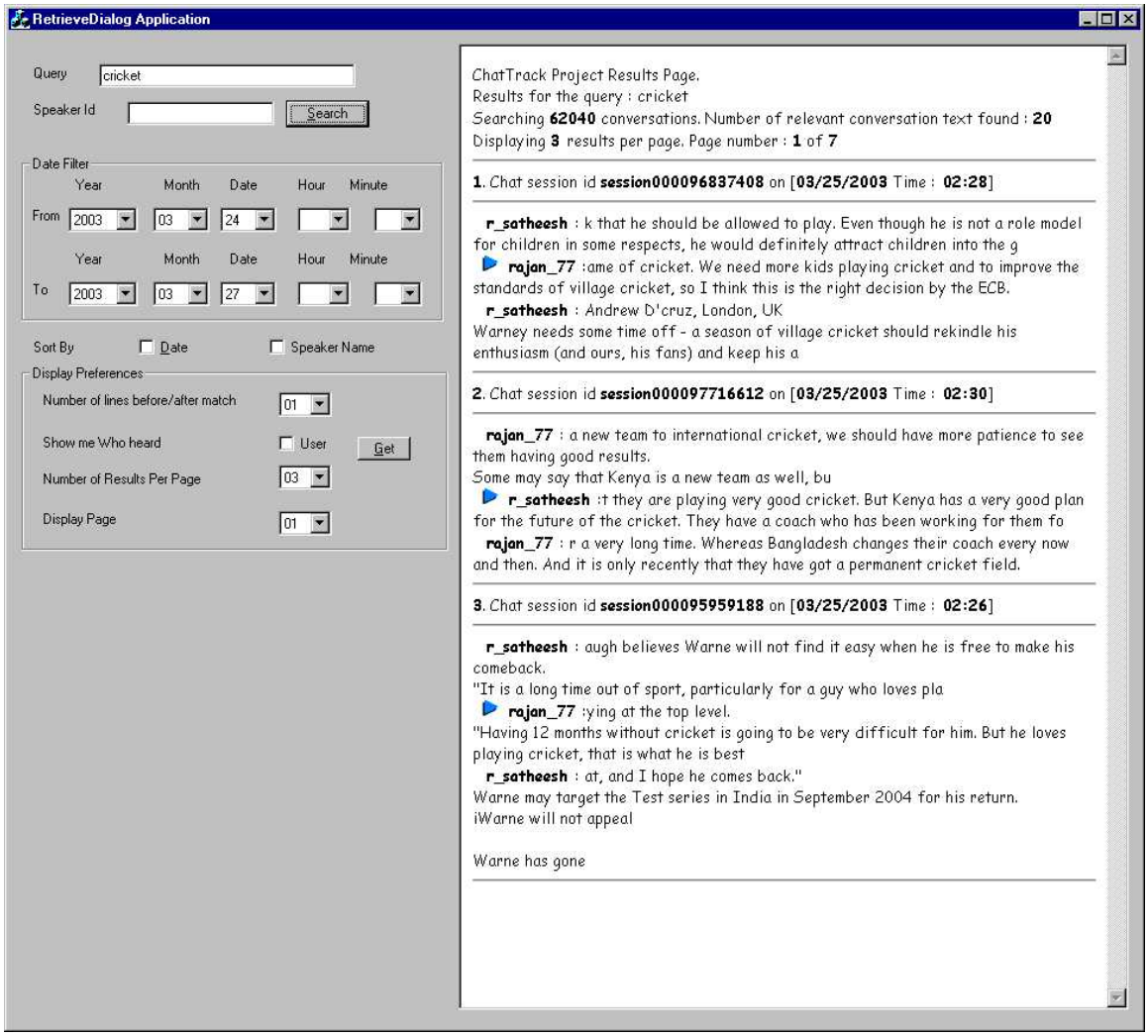


Figure 3.10: Screen Shot of the results screen with keyword said between some dates.

Finally by specifying the keyword and both the 'from' and 'to' date options, the system displays results that come between those date options. Results that are between March 24, 2003 and March 27, 2003 are displayed. The resolution can be extended to minute-based cut offs.

Figure 3.11 shows a screen shot with results sorted by date. The latest results, by date, are at the top of the results page.

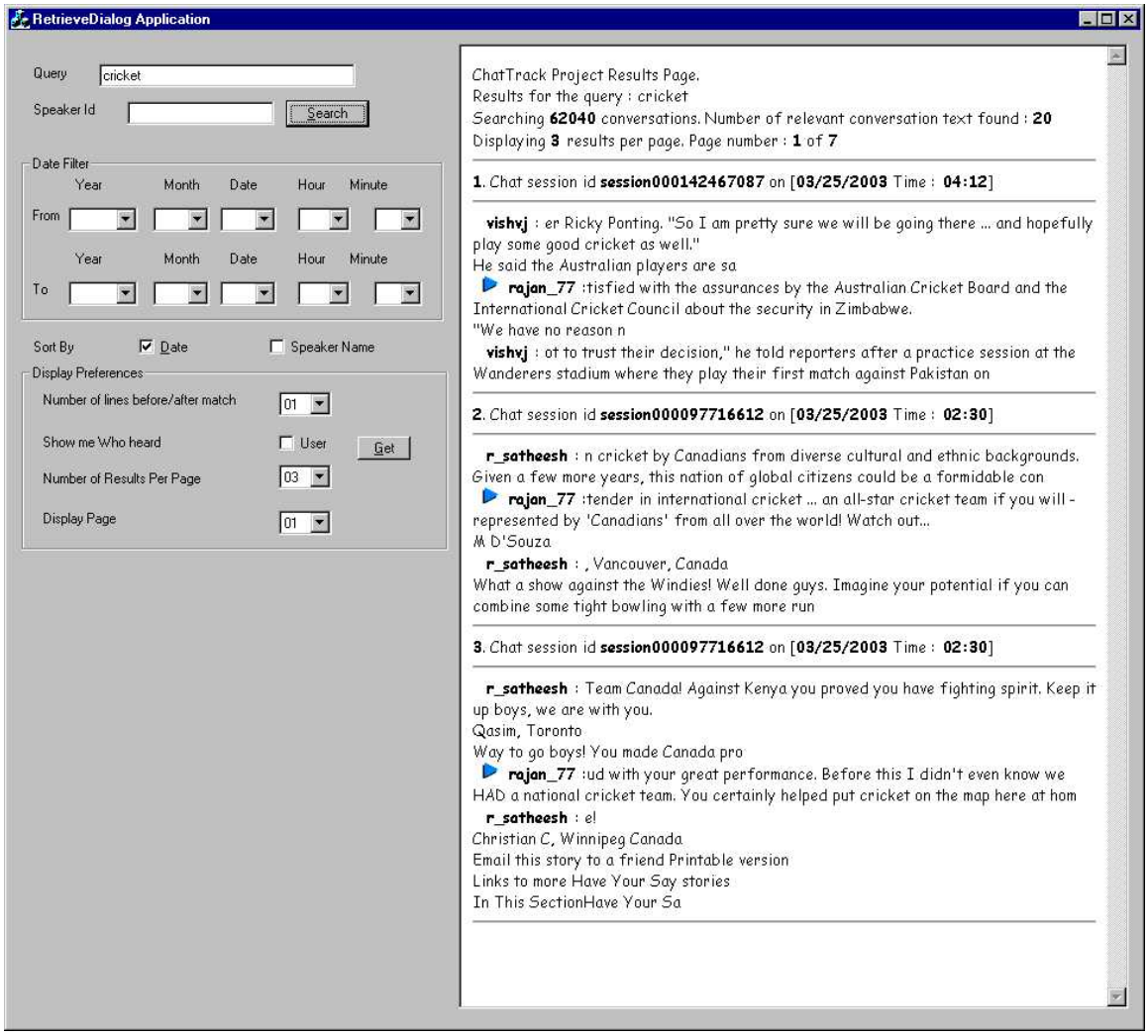


Figure 3.11: Screen Shot of the results screen with keyword and sorted by time.

Figure 3.12 shows results sorted by user email id. The results are displayed in ascending order of user email ids.

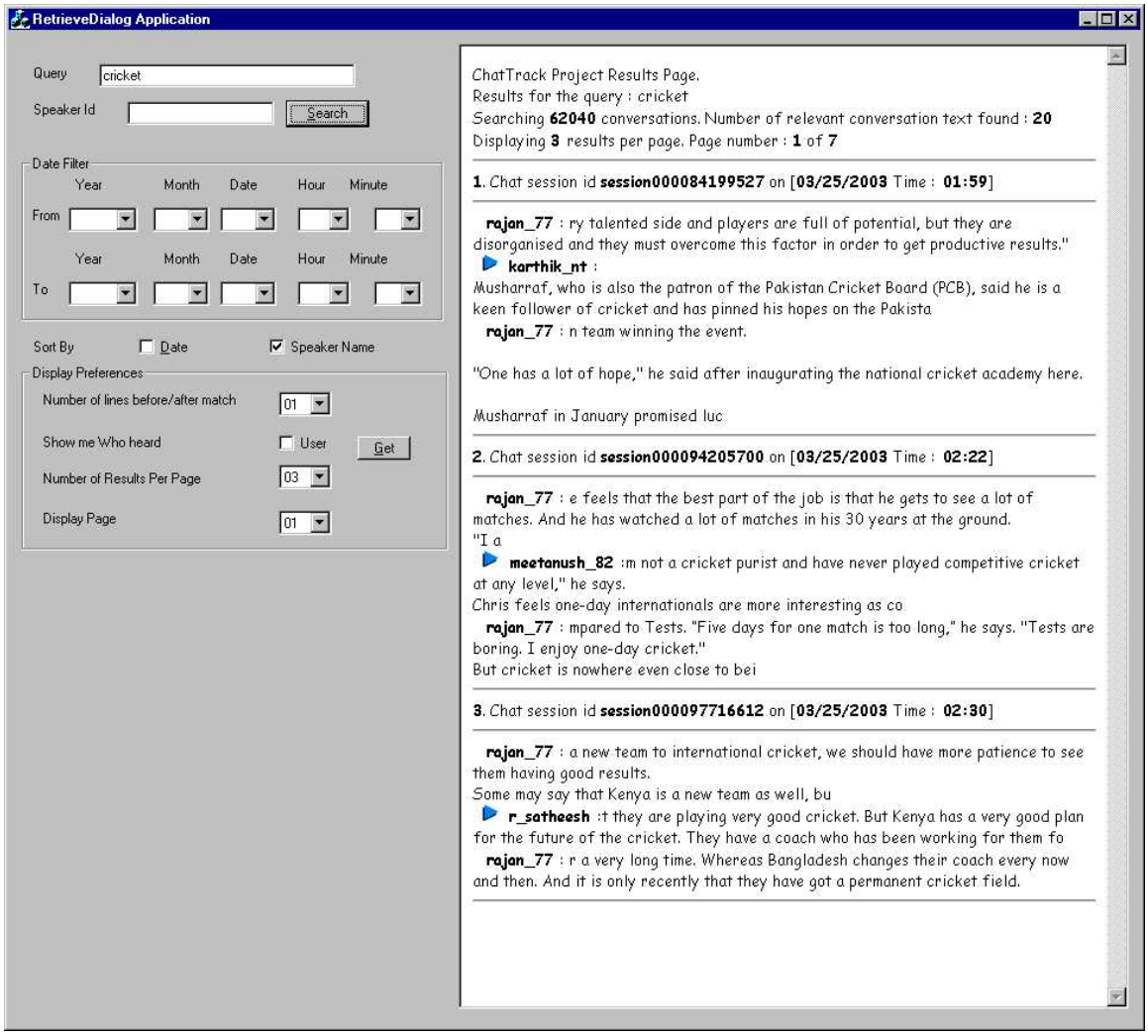


Figure 3.12: Screen Shot of the results screen with keyword and sorted by speaker email id.

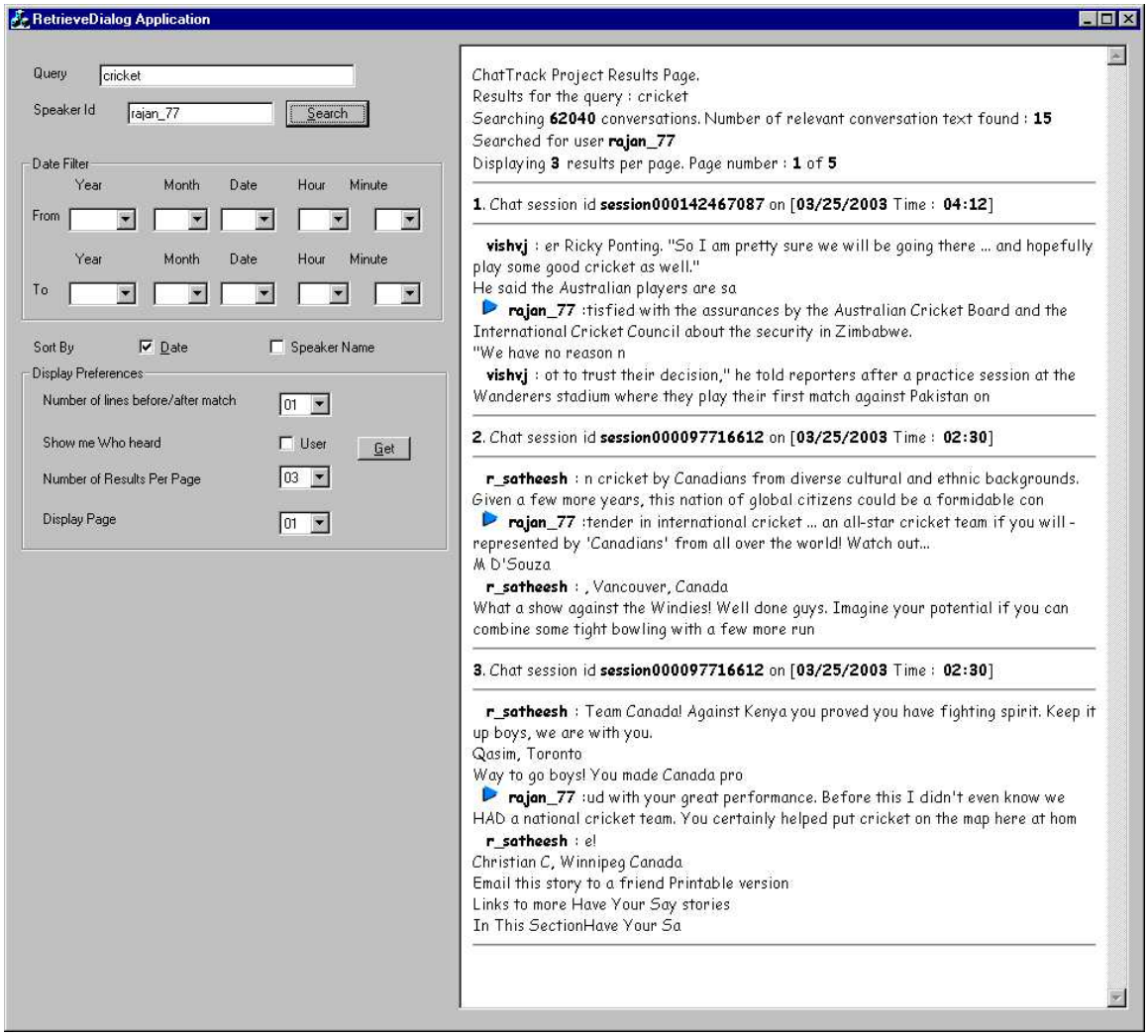


Figure 3.13: Screen Shot of the results screen with keyword and speaker email id and sorted by time.

Figure 3.13 shows results for the query ‘cricket’ uttered by the speaker with email id ‘rajan_77’ and the results are sorted by date. The most recent messages appear at the top of the results page.

Figure 3.14 contains a screen shot showing results filtered by date and sorted by user email id. The results displayed in ascending order of user email ids.

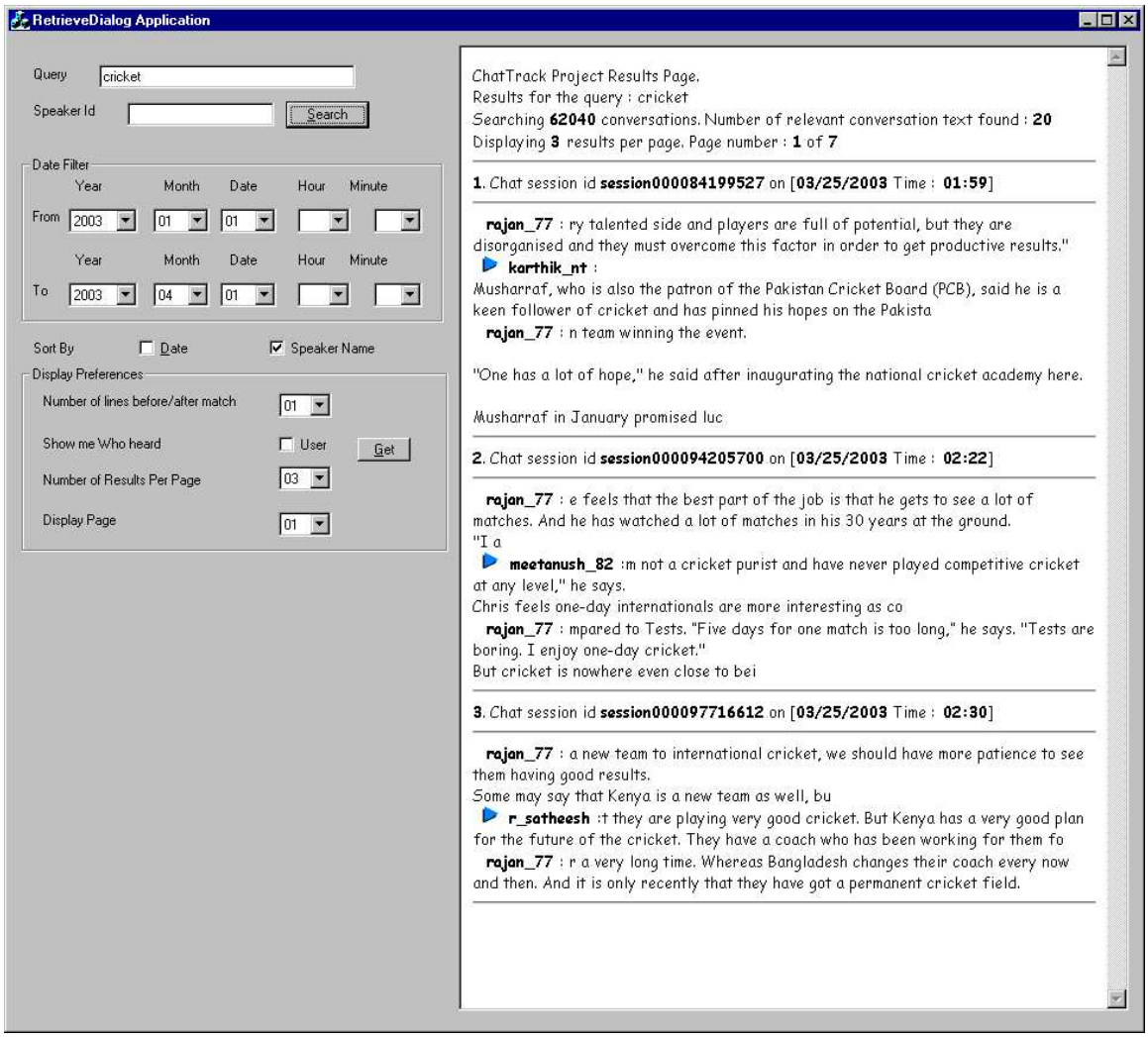


Figure 3.14: Screen Shot of the results screen with keyword and date option sorted by speaker email id.

Specifying the 'from' and 'to' date options and results sorted by user name is possible though the screen shots are not shown. The above result is for keyword 'cricket' between dates January 01, 2003 and April 01, 2003 sorted by speaker email ids.

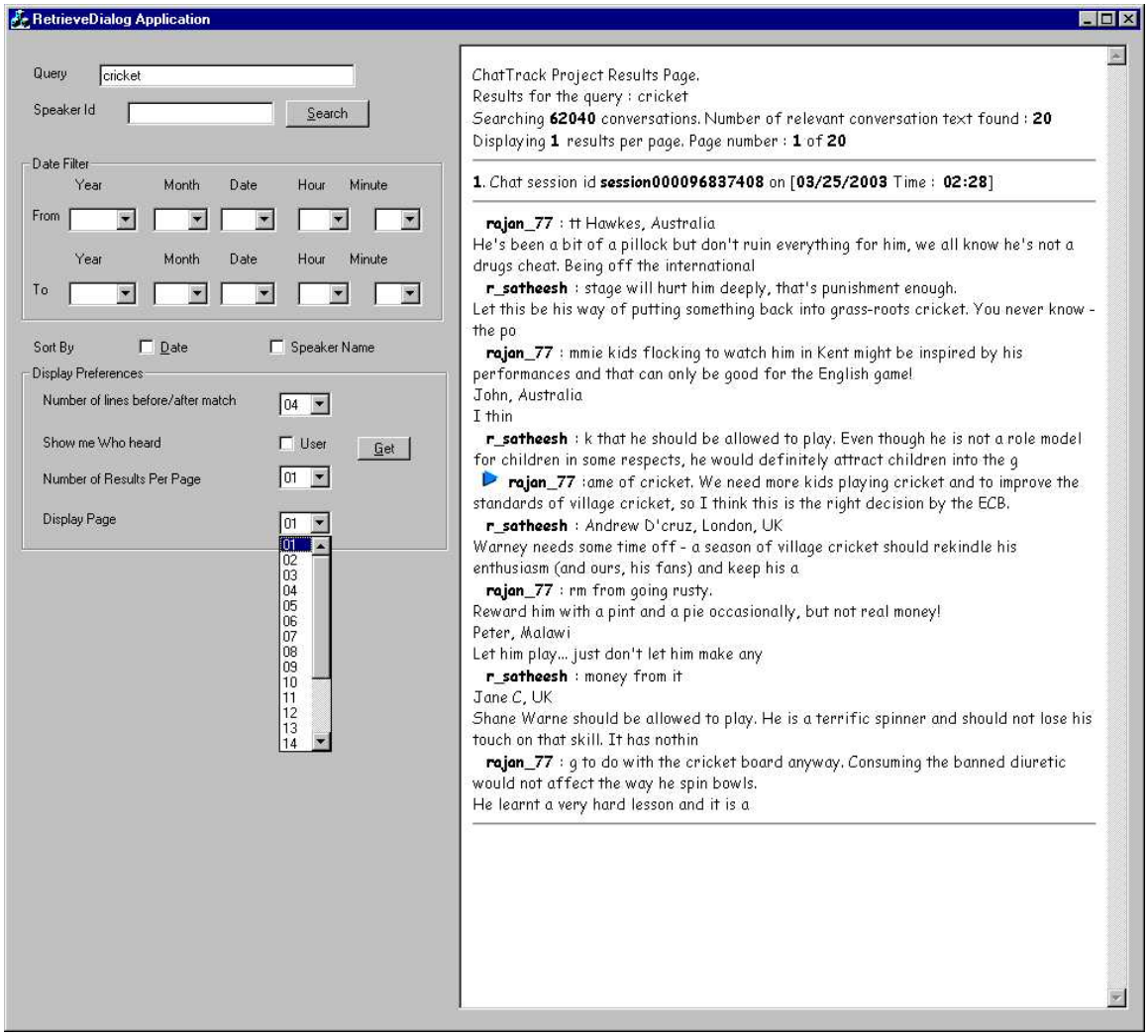


Figure 3.15: Screen Shot of the results screen with keyword and showing four lines before and after match.

The system allows the user to control the text displayed surrounding the exact keyword match. The number of lines of text before and after the exact match is provided as a query parameter. The above screen shot displays results with 4-messages around the exact match for the keyword ‘cricket’.

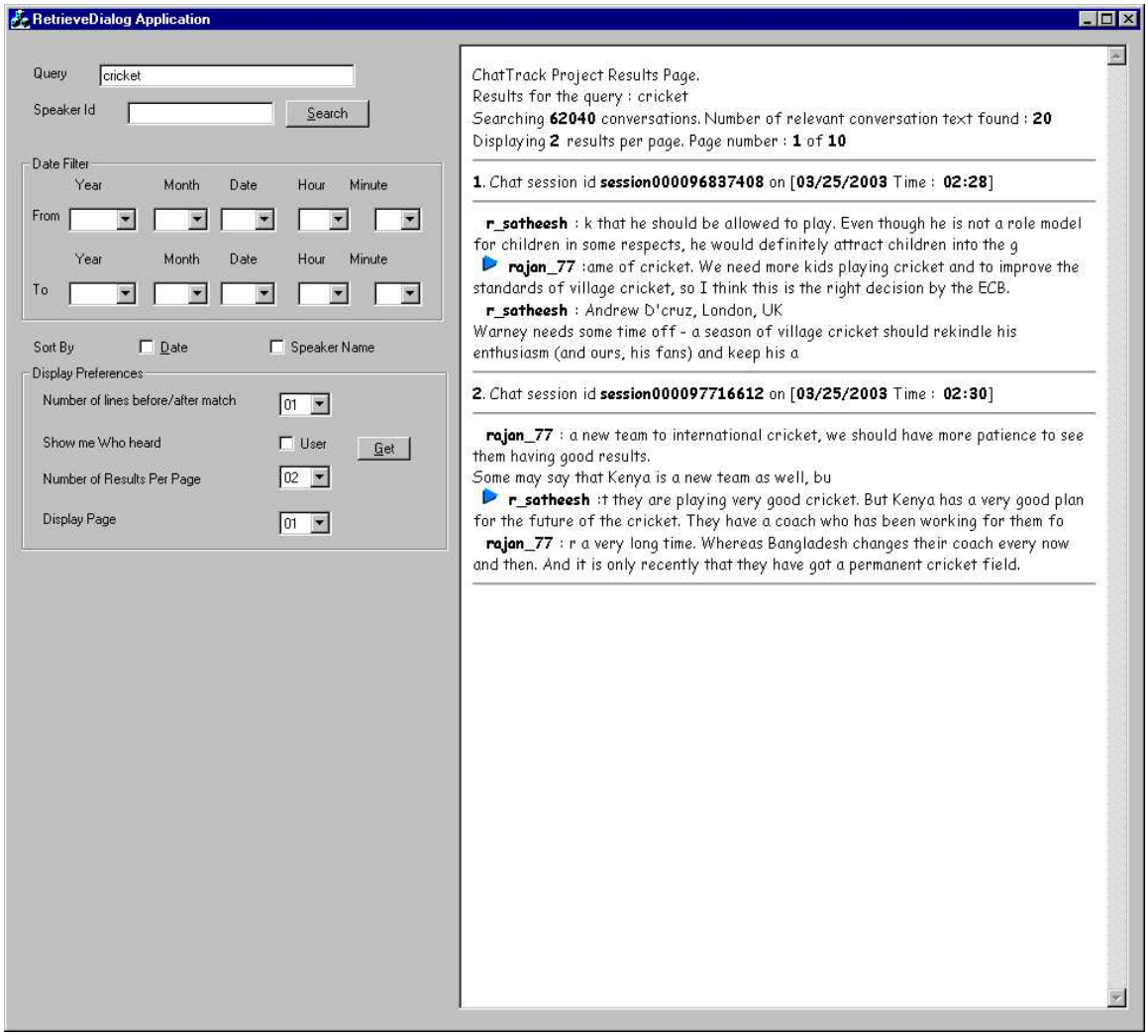


Figure 3.16: Screen Shot of the results screen with keyword and displaying two results per page.

Users can also specify the number of results that should be displayed in a single page. The results screen, Figure 3.16, displays two results per page for the keyword 'cricket'.

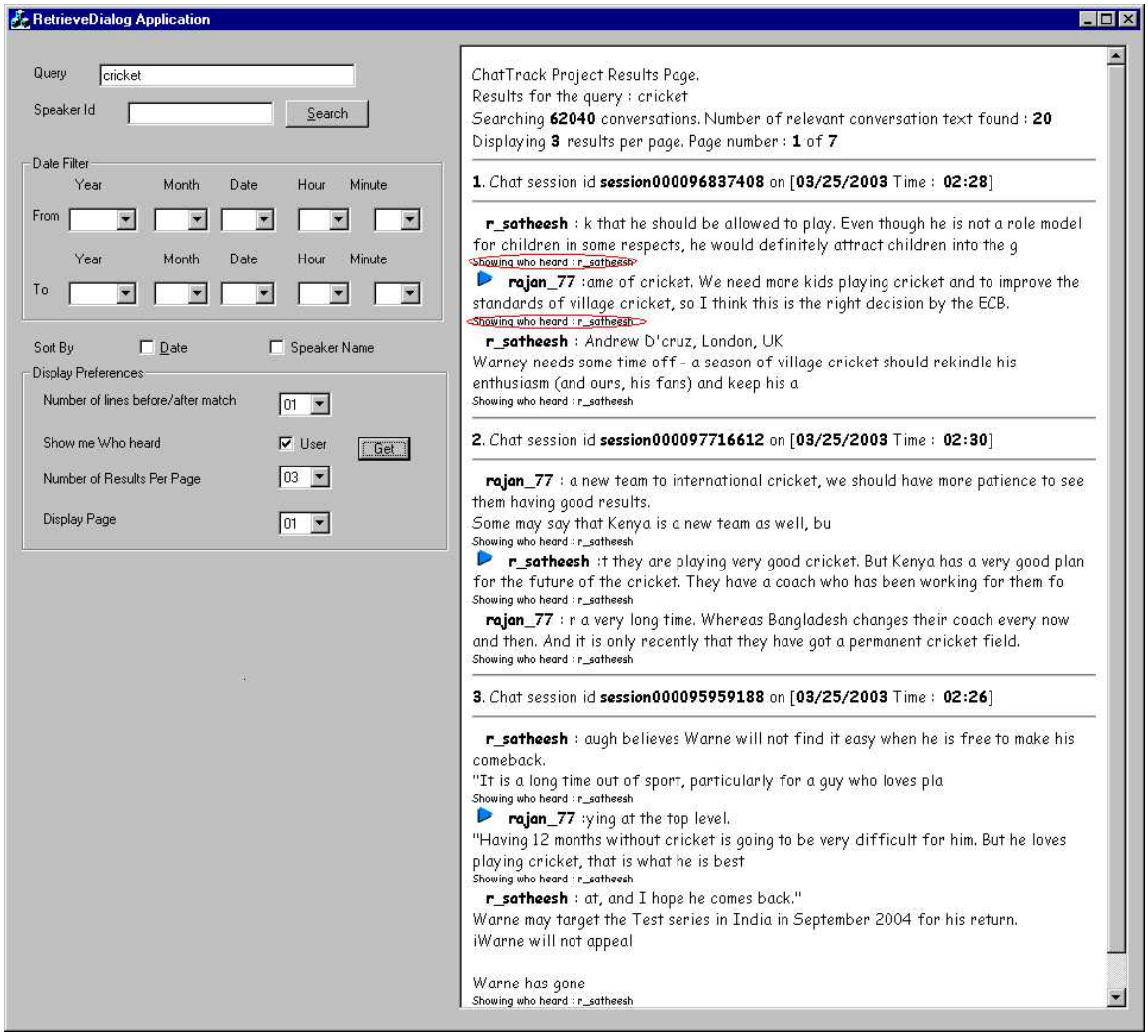


Figure 3.17: Screen Shot of the results screen with keyword and showing who heard the conversation.

The *show user* option can be used to view all who listened to a conversation text. As the logging system logs listener email ids, it is easy to view all who would have received the message. Although this option is unnecessary for one-on-one conversations, it is helpful in multi user conferences.

3.4. Summary

We have demonstrated the correct operation of our system on a variety of queries. The user interface provides the ability to search by keyword and to filter the results based on dates and speaker email ids. It tracks who heard each conversation and allows users control of how results are displayed. User may change the number of results per pages and change the number of messages that should appear before and after the exact match.

4. Scalability Issues

Since we are anticipating aggressive growth in the amount of chat data produced, we must discuss scalability issues. Some of the issues in this case are the growth of the size of the inverted index files, the number of files that get created by the archival system, and the indexing and retrieval times. In most cases, using MSN Messenger, the system is not going to be used for long durations on any day. This makes the growth of the files on the local system a slow process. Realistically, most users are not going to chat with more than 5 unique users at the same time and they are not going to chat more than 6 hours a day.

Since the system archives client side chat, we only need to archive the data received and sent by a single user. The growth of the word dictionary file is not going to be a steep one. Most of the informal conversation text will contain many stop list words and the stop list words are not going to be part of the word dictionary. We will need to extend a traditional stop list to contain words frequently used in informal chat, e.g., gtg, gg, etc. We do not expect the number of distinct tokens to grow quickly once a basic vocabulary of approximately 100,000 words is achieved.

To evaluate the scalability of our system, we generate sample chat messages by selecting words from a document. The message length for a conversation text is kept constant at 150 bytes although MSN Messenger allows messages up to 400 bytes long.

If we assume that the user is has 5 chat windows open, each generating 1 message every 10 seconds, that would equate to 5,400 files per hour, at 5 hours per day, 27,000 files per day would be created. A month would be 810,000 files and a year would be 295,650,000 files. These are high estimates and more typical would create 10% of this

data, or 2,956,500 files per year.

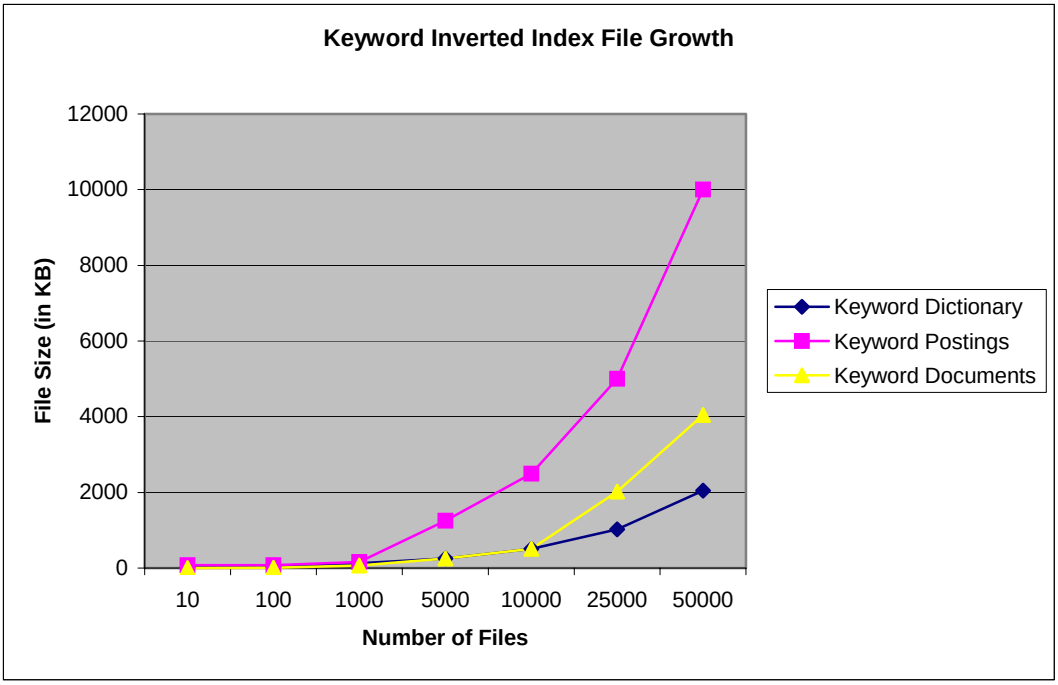
In practical terms, 90% of the times, no user has the patience to type more than 100 characters at a time. So, 150 characters is a pessimistic average length.

Based on real data, the system archived 504 message files in 128 minutes in one session and in another session captured 132 message files in 35 minutes. In both the cases, the number of files created per minute is approximately 15. This number is well below the estimated volume of 20 message files per minute. When we count the total number of files created, ewe estimated around 60 files per minute, yet the actual was just 15 files per minute.

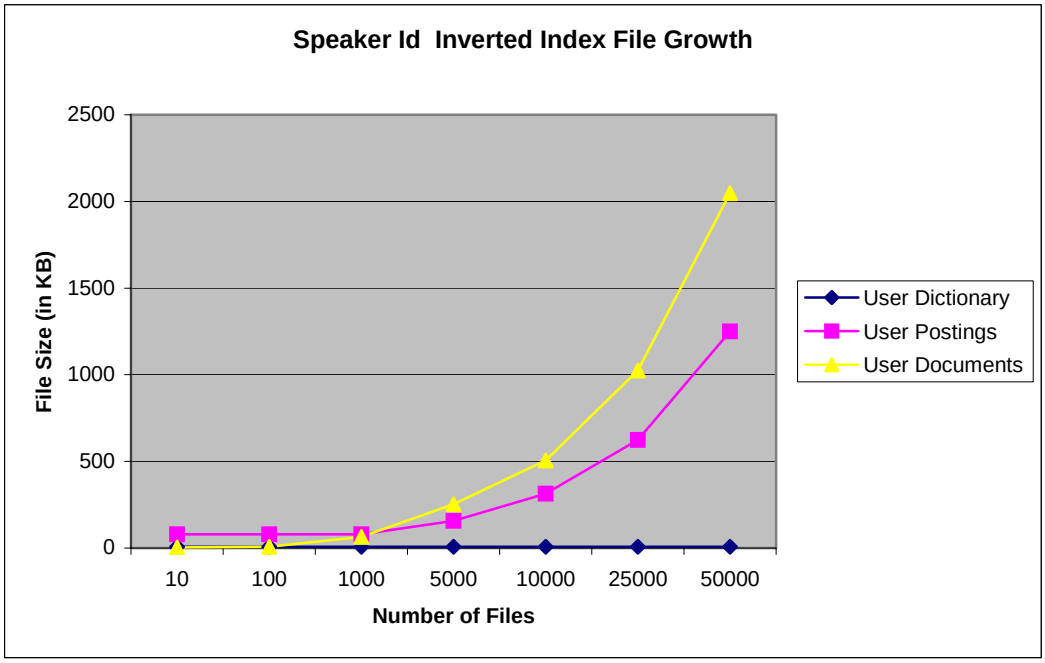
The following table shows the rate if growth of the inverted file as a function of the number of chat files. For indexing 50,000 files, the system uses 22MB. Thus, we need to periodically clean up the files.

Number of Files	Dict (Size in KB)	Post (Size in KB)	Docs (Size in KB)	User Dict (Size in KB)	User Post (Size in KB)	User Docs (Size in KB)
10	16	79	4	8	79	4
100	32	79	8	8	79	8
1000	127	157	64	8	79	64
5000	253	1250	253	8	157	253
10000	506	2500	506	8	313	506
25000	1024	5000	2024	8	625	2024
50000	2024	10000	4048	8	1250	4048

Table 2: Showing the files size growth with the change in number of files.



Graph 1: Shows the growth of keyword Inverted Index growth with respect to number of files.

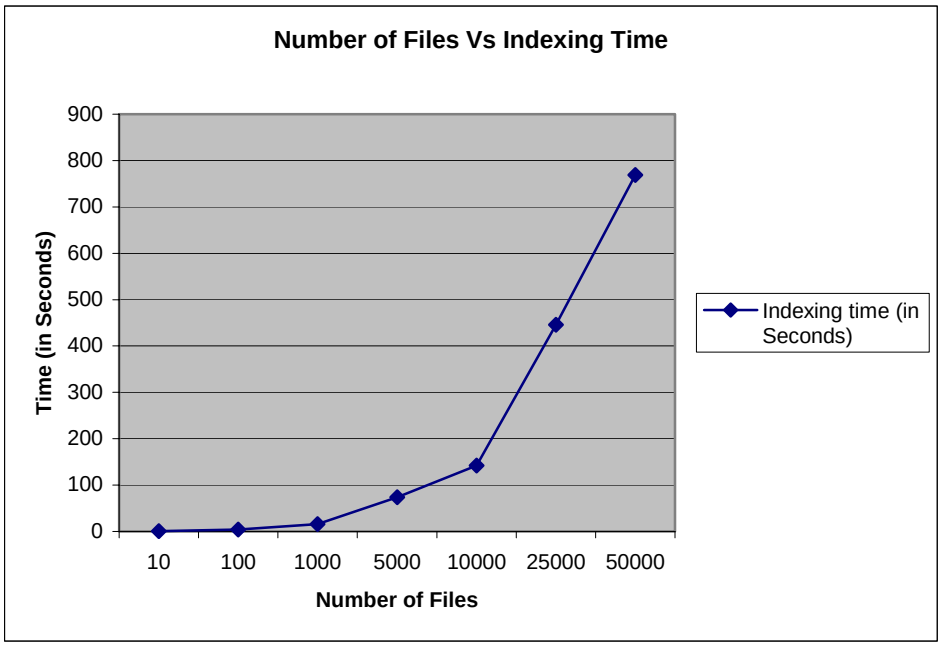


Graph 2: Shows the growth of speaker email id or User id Inverted Index growth with respect to number of files

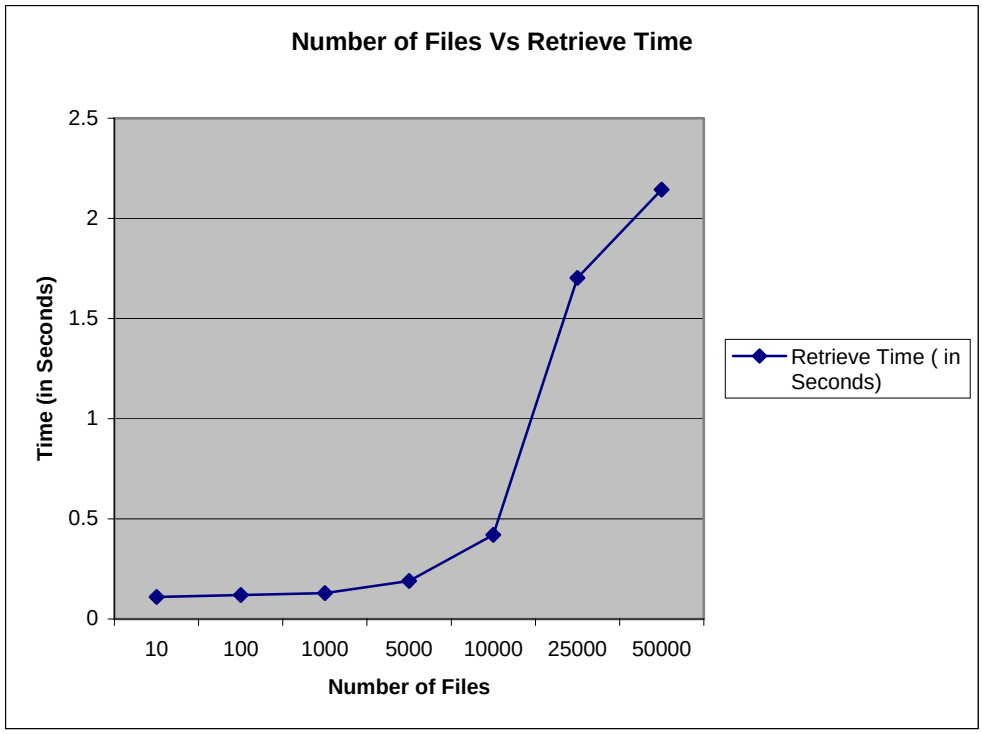
We also evaluate the effect of the archive size on the indexing and retrieval times. Table 3 shows the time taken to index a varying number of files. The retrieval time is shown on the third column.

Number of Files	Time for Indexing (in seconds)	Time for Retrieve (in seconds)
10	1	0.11
100	4	0.12
1000	16	0.13
5000	74	0.19
10000	142	0.42
25000	446	1.703
50000	769	2.143

Table 3: Indexing and Retrieval time with respect to number of files.



Graph 3: Shows the indexing time with respect to number of files.



Graph 4: Shows the retrieval time with respect to number of files.

Currently the archival system gives a call to indexing every approximately 30 seconds. From the data collected, in 30 seconds the indexing system can index 2000 files each having 150 bytes of data. The Messenger service can only produce maximum 30 text messages (one message per second) in 30 seconds. So, calling the indexing system every 30 seconds is really reasonable and it can guaranteed that the system will index the messages without falling behind. Also, since the indexing is incremental, adding 100 files to an existing index can be done in 4 seconds no matter how large the archive grows.

Based on the number of files created by real chat conversation, it is easily seen that this system can provide a scalable architecture. The system plays back the sequence

of chat conversation, as it happened, accurately. It provides a capability to search based on a number of query preferences. Currently, it supports queries based on keywords, search ability to filter results by date, speaker name which many of the existing products in the market does not provide. One of the key points of the system is its ability to display the email ids of the users who listened the conversation. This feature will be helpful when it is a multi user conference.

5. Conclusions

This system provides an effective tool for archiving and retrieving chat messages from MSN Messenger. The system was tested on Windows NT 4.0 and Windows 2000 Professional edition and with some minor code changes it can be made to work for other versions of Windows. The current system logs messages from MSN Messenger versions 5.0 and earlier.

6. Scope for future work

6.1. XML

XML can be used to store the files that are getting created by the archival system. By representing the logged message of a whole session in a single XML file, the system can drastically reduce the number of files. During indexing the XML file can be parsed to create individual files. The files can be deleted after indexing them because the data is already present in the XML file. Also, using XML allows the archiving system to work with other logging products that agree to produce output in XML format.

6.2. User Profiles

User profiles [33][34] based on the text messages received can be seen a useful extension to this work. Generation of user profile will help in giving a general idea of the nature of the person on the other side. Language, used by the users, can be an indication of the nature or character. If a person assuming an age of young person, language usage can help in identifying the age range of the other user. By indexing all messages by email ids, we can easily group together all messages from a single user for further analysis.

6.3. Thread Identification

Chat conversations involve discussions about many topics during a session. Segmenting the chat messages based on the topics conversed would be an interesting research problem. Text segmentation [22][24][26][27] is arranging the message text based on the subject of the conversation. The process of thread identification starts with detecting the topic of the conversation [32]. Messages that belong to the same topic, or

resemble a topic, can be grouped together. Research on Topic Detection and Tracking to newswire and broadcast news [21][23][25][30] is being conducted the same for chat conversation messages could be done. Then messages, on same subject could also be arranged or segmented as threads similar to message threads in newsgroups.

6.4. Text Summarization

Text Summarization [28][29][31] is the process of summarizing the conversation on a topic and displaying only important messages or displaying high points of a conversation. Summarization is a three-step process: content identification, conceptual organization and realization. Text summarization has been studied in many areas, but summarizing informal chat messages poses unique set of problems.

6.5 Server Side Chat

This thesis mainly dealt with client side archiving and providing features that are absent in the existing systems. The system has been coordinated with Chat Track project and current work is integrating the search capabilities described here with an IRC chat server. We are also developing a browser-based search client.

Bibliography.

[1] Yahoo Messenger Enterprise Edition. 2003. http://enterprise.yahoo.com/messenger/., "Yahoo! Messenger Enterprise Edition", April 2003.

[2] MSN Messenger Service. 2003. http://advantage.msn.co.in/flattened/EBC0BB22-A87B-4BC2-A321-FCACA47F54BB.asp., "MSN Messenger Service", April 2003

[3] Instant Messengers. 2001. http://www.instant-messengers.com/site/news/im_more_popular_than_ever.htm., "Instant Messaging More Popular Than Ever at Work", April 2003

[4] IM Means Business. IEEE Spectrum Communications November 2002. pp 28-32.

[5] University Daily Kansan 2003. Article: *"Bohl rumors abound"*, February 14,2003

[6] The Guardian Angel. 2001. http://www.theguardianangel.com/internet_safety_adults_chat.htm., "Internet Safety Awareness", March 2001.

[7] Milwaukee Journal Sentinel. 2000. http://www.jsonline.com/news/metro/sep00/net12091100a.asp., "Judge aims sentence at Internet chat rooms", September 2000.

[8] About.Com. 1999. http://crime.about.com/library/weekly/aa101299.htm., "The War Against Pedophiles", December 1999.

[9] Naples/Collier News. 1998. http://www.naplesnews.com/today/local/a130833f.htm., "Computer abuse: Naples woman a victim of 'virtual murder' ", September 1998.

[10] Iambigbrother. 2003. http://www.iambigbrother.com., "Who is your family chatting with? Find out", April 2003.

[11] NetNanny. 2003. http://www.netnanny.com/products/netnanny4/description.html., "Net Nanny Product Description", April 2003.

[12] Cyber-Snoop. 2003. http://www.cyber-snoop.com/., "Cyber Snoop Version 4.0 Internet Monitoring Software", April 2003.

[13] Desktopsnooper.2003. http://www.desktopsnooper.com/productinfo.html., "Product Information", April 2003.

[14] I-Spy. 2003. http://www.i-spy-software.com/., "What iSpyNOW can do for you", April 2003.

[15] Pearl Echo Internet Monitoring Software. 2003. http://www.pearlsw.com/., "Internet Monitoring Software Solutions for the Way You Work, Study and Play", April 2003.

[16] Spector Pro. 2003. http://www.spectorsoft.com/products/SpectorPro_Windows/index.html., "Home >>> Products >>> Spector Pro for Windows >>> Product Description", April 2003.

[17] Spy Buddy 1.9. 2003. http://www.spy-gadgets.com/spybuddy/., "Award-Winning Internet / Computer Monitoring and Surveillance Spy Software", April 2003.

[18] Keyboard Monitor Keylogger 3.0 2003. http://www.spy-software-directory.com/keyboard_monitor.asp., "Keyboard Monitor 3.0 Features", April 2003.

[19] Ricardo Baeza-Yates, Berthier Ribeiro-Neto. "Modern Information Retrieval", Addison-Wesley, 1999.

[20] Ian H. Witten, Alistair Moffat, Timothy C.Bell. "Managing Gigabytes. Compressing and Indexing Documents and Images". Morgan Kaufmann II edition 1999.

[21] Charles L Wayne. “Topic Detection and Tracking using idf-Weighted Cosine Coefficient”, *Proceedings of the DARPA Broadcast News Workshop*, San Francisco, CA, 1999, pp 189-192.

[22] Doug Beeferman, Adam Berger, John Lefferty. “Statistical Models for Text Segmentation” Machine Learning 34(1999), 1999, pp 177-210.

[23] Charles L Wayne. “Topic Detection and Tracking Overview and Perspective”, *DARPA Broadcast News Transcription and Understanding Workshop, Lansdowne, V.*, February, 1998.

[24] Jay M.Ponte, W.Bruce Croft. “Text Segmentation By Topic”. *Proceedings of the First European Conference on Research and Advanced Technology for Digitial Libraries*, 1997, pp 120-129.

[25] J.P. Yamron, L. Gillick, S. Knecht, S. Lowe, P. van Mulbregt “Statistical Models for Tracking and Detection”. *Working notes of the DARPA TDT-3 Workshop.* 2000.

[26] Doug Beeferman, Adam Berger, John Lafferty “Text Segmentation Using Exponential Models”. *In Proc. Empirical Methods in Natural Language Processing 2 (AAAI) '97*, Providence, RI, 1997.

[27] Hideki Kozima. “Text Segmentation Based On Similarity Between Words”. *In Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics* , Columbus, OH. 1993, pp 286-288.

[28] Regina Barzilay, Michael Elhadad. “Using Lexical Chains for Text Summarization”. *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, Madrid, Spain, July 1997, pp 10-17.

[29] Julian Kupiec, Jan Pedersen, Francine Chen. “A Trainable Document Summarizer”. *In Proceedings, 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval”,* Seattle, Washington, July 1995, pp 68-73.

[30] Jon Fiscus, George Doddington, John Garofolo, Alvin Martin “NIST’S 1998 Topic Detection And Tracking Evaluation (TDT2)”. *Proceedings of the DARPA Broadcast News Workshop, Virginia, USA.* 1998

[31] James Allan, Rahul Gupta, Vikas Khandelwal. “Temporal Summaries of News Topics”. *Proceedings of SIGIR 2001*, New Orleans, LA, September 2001, pp 10-18.

[32] Ron Papka, James Allan, Victor Lavrenko. “UMASS Approaches to Detection and Tracking at TDT2”. *In Proceedings of the 1999 DARPA Broadcast News Workshop*, Herndon, Virginia, February-March 1999, pp 111-116.

[33] Michael Pazzani, Daniel Billsus. “Learning and Revising User Profiles: The Identification of Interesting Web Sites”. *Machine Learning 27,* 1997, pp 313-331.

[34] Susan Gauch, Jason Chaffee, Alexander Pretschner; “Ontology-Based User Profiles for Search and Browsing”. *Submitted to User Modeling and User-Adapted Interaction (UMUAI) journal.* June 2002.

Appendix A.

Administrative messages from Microsoft MSN messenger

1. Never give out your password or credit card number in an instant message conversation.

2. <nickname/email id> has left the conversation.

3. <nick name/email id> has been added to the conversation.

4. <nick name/email id> would like to send you the file <filename> (46 Kb). Transfer time is less than 1 minute with a 28.8 modem. Do you want to Accept (Alt+T) or Decline (Alt+D) the invitation?

5. Transfer of file <filename> from <nick name/email id> has been accepted. Starting transfer...

6. You have successfully received <filename> from <nick name/sign-in name>. Before opening this file, you may want to scan it with a virus-scanning program.

7. The following message could not be delivered to all recipients: