12TH ICCRTS
“Adapting C2 to the 21st Century”

Supporting Chat Exploitation in DoD Enterprises

Track Session:

Network-Centric Experimentation and Applications

Authors:

Christopher D. Berube, *The MITRE Corporation*
Janet M. Hitzeman, Ph.D., *The MITRE Corporation*
Roderick J. Holland, *The MITRE Corporation*
Robert L. Anapol, *The MITRE Corporation*
Stephen R. Moore, *The MITRE Corporation*

Point of Contact:

Christopher D. Berube
The MITRE Corporation
MS M380
202 Burlington Road, Bedford, MA 01730
781-271-8303
cdb@mitre.org

Abstract

The use of internet chat to support information exploitation in time-sensitive environments has been explosive in recent military conflicts. During Operation Iraqi Freedom, the widespread use of chat among operators and intelligence analysts provides increased information flow which, when properly managed, provides increased situational awareness in support of collaborative decision-making processes. However, while the use of chat in military environments has a positive operational effect, it is also difficult to effectively manage and exploit. Specifically, it has been noted among operators that it is often difficult to maintain awareness of what is going on in and across multiple chat rooms at the same time.

Our research focuses on the application of information extraction to the problem of providing automated "chat alerts" of pertinent events to operators monitoring multiple chat rooms. In particular, we demonstrate that information extraction allows accurate pinpointing of tactical entity class data (e.g., air mission) in military chat. We also describe a software prototype designed to support information extraction, and featuring graphics for user-specified chat profiles and views of extracted chat entity and event class data in real-time and retrospective processing.

1 Introduction

The use of internet chat to support information generation and exploitation in time-sensitive environments has been explosive in recent military conflicts, such as Operation Iraqi Freedom (OIF). During OIF, the widespread use of chat among intelligence analysts, planners and others from various disciplines provided improved "information flow" which in turn provided increased situational awareness and decreased response times in collaborative decision-making processes.[1] However, the ability to maximize the benefit of chat in supporting aspects of situational awareness and decision-making is limited by the volume of chat that may need to be monitored in a single chat room and the great number of chat rooms that require monitoring. Therefore, there is need for an automated means of extracting tactically relevant information from multiple chat rooms.

In this paper, we discuss the results of a research project that targets this need as part of The MITRE Corporation's Technology Program. This project, "Facilitating Sense Making for Situational Awareness," is an Air Force-sponsored Mission-Oriented Investigation and Experimentation (MOIE) project currently in its second year. The primary goal of this research project is to develop a software system to support operators/analysts (i.e., "users") monitoring multiple chat rooms by automatically extracting entities and events in real time as they are being discussed. Here, information extraction in chat is performed using MITRE's Alembic information extraction tool.[2] The overall system gives the user a template for indicating their areas of interest, such as certain aircraft call signs associated with the conduct of air missions. Using this user profile, appropriate chat "alerts" are generated and displayed to a user along with details associated with the alert. The advantage for the user is that he is less likely to miss an event or entity of importance to him, and this increases the ease with which he can track entities and events in a large number of chat rooms at once.

2 System Design and Software

The design of our software system was guided by the desire to support chat exploitation needs within Department of Defense (DoD) enterprises. Through course of our research, and based on the work of other researchers, we noted that different classes of users of chat require different capabilities to successfully manage chat. Specifically, we identified three dimensions of support for chat exploitation, and designed our system to enable them:

- Granularity of analysis – extraction of information in single chat rooms as well as across multiple chat rooms; information extracted at the entity, event and summary levels
- Focus – "what is being said in chat" versus "who is saying it"
- Time – real-time versus retrospective ("forensic") analysis of chat.

[1] "War Planners Talk in 'Chat Rooms' To Exchange Targeting Data, Tips," 2 May 2003, Source: Defense Information & Electronics Report

[2] "Description of the *Alembic* System Used for MUC-6," John Aberdeen, John Burger, David Day, Lynette Hirschman, Patricia Robinson and Marc Vilain, *Proceedings of the 6th Conference on Message Understan*ding, Columbia, MD, 1993, pp. 141-155.

In our system, utterances in chat (i.e., chat “messages”) are provided either from a static log or in real-time from a chat server. The entry point for every chat message into the system is the Military Language Pre-Processor (MLPP) application. The MLPP provides a first-level of information extraction for each chat message and acts as a pre-processor for the Alembic information extraction application. The MLPP generates four classes of Java objects: ‘ChatMessage,’ ChatAnalysis,’ ‘ChatAlert,’ and ‘ChatDocument.’ Figure 1 illustrates the relationship and flow of information between elements of the system.

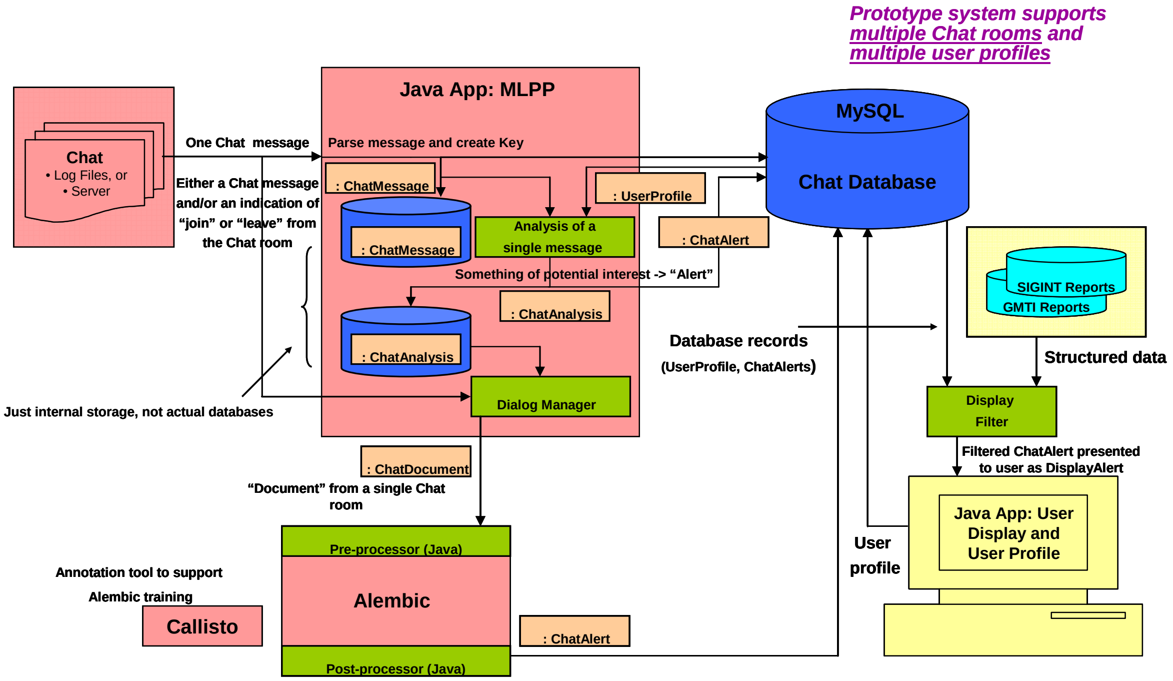

Prototype system supports multiple Chat rooms and multiple user profiles
Chat
• Log Files, or
• Server
One Chat message
Either a Chat message and/or an indication of “join” or “leave” from the Chat room
Java App: MLPP
Parse message and create Key
: ChatMessage
: ChatMessage
Analysis of a single message
: UserProfile
: ChatAlert
Something of potential interest -> “Alert”
: ChatAnalysis
: ChatAnalysis
Dialog Manager
Just internal storage, not actual databases
MySQL
Chat Database
Database records
(UserProfile, ChatAlerts)
SIGINT Reports
GMTI Reports
Structured data
Display
Filter
Filtered ChatAlert presented to user as DisplayAlert
Java App: User Display and User Profile
User profile
: ChatDocument
“Document” from a single Chat room
Pre-processor (Java)
Alembic
Post-processor (Java)
: ChatAlert
Annotation tool to support Alembic training
Callisto

Figure 1: Chat Exploitation System Architecture

The ‘ChatMessage’ object contains the chat utterance as well as identifying information such as user name, timestamp and the name of the chat room. This additional information makes up part of the “key” used to store each ‘ChatMessage’ object in the MySQL Chat Database. The ‘ChatAnalysis’ object contains the results of the information extraction performed on a single chat message (i.e., they come in pairs). A ‘ChatAlert’ object contains a vector of ‘ChatAnalysis’ objects resulting from information extraction performed on one or more chat messages (which is possible using the Alembic application). ‘ChatAlert’ objects generated by MLPP are stored in the Chat Database.

The Dialog Manager within the MLPP application takes one or more pairs of ‘ChatMessage’ and ‘ChatAnalysis’ objects and combines them in a ‘ChatDocument’ object, which is the processed by the Alembic application. If Alembic extracts any additional information beyond what it was provided from the MLPP, a ‘ChatAlert’ object is generated by Alembic and stored in the Chat Database.

At this point, the Chat Database contains a set of ‘ChatAlert’ objects generated by both MLPP and Alembic. Selection of these objects to support the interests of a particular user is carried out by a ‘DisplayFilter’ object (which is actually contained in the MLPP application) by means of a ‘UserProfile’ object. Finally, the results of the information extraction by either MLPP or Alembic are displayed as part of the ‘UserDisplay’ Java application.

3 Causing ‘Chat Alert’ Generation

One weakness of using chat (or any other form of communication which requires the user to multiprocess) is that important events discussed in chat may be missed if the user takes a quick break or has his attention focused on another chat room. The purpose of providing users with chat alerts is to point out important events in all chat windows, reducing the possibility that an event is missed and thereby reducing the burden on the user.

The system first checks each chat message for all events which may be of interest to any user, and then filters the messages based on the particular user’s stated areas of interest. The process of examining a message and annotating alert information within that message is done in two phases: The military language pre-processor (MLPP) phase, which looks for information using regular expression rules, and the Alembic information extraction phase, which uses contextual information.

3.1 Military Language Pre-Processor (MLPP)

The MLPP application not only serves as the mechanism for parsing and distributing chat messages to other parts of the system, but it also provides a first-level of information extraction for a single chat message. Specifically, the MLPP provides the capability of implementing regular expression processing within a single chat message. The result of this processing provides a pre-processed chat message for the more complex information extraction carried out by Alembic.

The use of regular expressions in the MLPP allows for the extraction of semi-structured data types; that is, words or phrases recognized based on their “regularness” or format. Currently, there are seven semi-structured data types implemented in the MLPP application:

- Latitude and longitude – in decimal and degrees/minutes/seconds formats
- Basic Encyclopedia (BE) number – a military standard for geo-referencing
- Electronics Intelligence (ELINT) notation (ELNOT)
- Aircraft voice call sign – designator used for voice identification of an aircraft
- Military Grid Reference System (MGRS) – a military standard for geo-referencing
- Zulu time – military standard time.

The MLPP application is implemented in Java, using Java’s regular expressions package to provide the extraction of semi-structured data types. As an example of a regular expression implemented by Java, consider the extraction of the aircraft voice call sign using the following format:

- Four to twelve letters followed by one to two digits
    - The first letter must be upper-case
    - There can be a white space between the last letter and the first digit.

Below is a string (Regex) containing an implementation of this Java regular expression:

- Regex = "(\\p{Punct}*)(([A-Z])[a-zA-Z]{3,11})(\\s?\\d{1,2})(\\p{Punct}*)".

3.2 Regular Expressions vs. Information Extraction

Given a list of words that fall into a certain category, searching within a text for a word or phrase on such a list (i.e., *keyword search*) is the simplest way of detecting words which fall into that category. For example, a word list containing surnames may contain the word "Smith," allowing us to easily tag the word with SGML as in

<PERSON>Smith</PERSON>

Regular expressions are one step up from this simple search in that they recognize terms which fit a pattern. Zulu time (i.e., military time) is a good example of an expression that can be easily captured using the following description of a regular expression: One form of a Zulu time expression (such as "2010Z") must consist of a 5-character string ending in "Z" or "z" and beginning with four integers. A more complex example is an expression giving the latitude of some object, such as 0421322N. In this case, the "word" must consist of a string of digits that satisfy the constraints for angular position in degrees, minutes and seconds, followed by a single alphabetic character indicating the quadrant in which the object is located (e.g., "N" for the northern hemisphere).

Regular expressions and keyword processing have their limitations, however. For example, if a text refers to a name which is not on the surname word list or does not fit a regular expression pattern, it will not be recognized; expressions such as "Mr. Heemachandra" or "I'll be home around 2-ish" will be passed over if not properly accounted for by these methods. Overgeneralization can also be a problem, e.g., given a name such as "Smith" on a word list of names, the keyword processing system will tag the word as a name even in contexts such as "Smith & Co." which refers to an organization or to any reference of a blacksmith as a "smith," which refers to an occupation. Similarly, while regular expressions can capture unknown names of diseases, such as "hyperendebulovirus," the more powerful method of information extraction can also capture unknown diseases expressed as multi-word phrases, such as "precocycal hyperendebulous virus," and can tag appropriate terms such as "vironuclear plague" while excluding "a plague of terrorists." It does this in part

- by looking for a word that ends in "virus,"
- by looking for a disease-related noun from a list containing words such as "virus," "plague," and "flu" and then checks for an appropriate adjective in the text which precedes that noun, and
- partly by looking for context that is typically used with disease terms, e.g., "…caught the ___," "…died of the ___," "…complications from the ___."

Information extraction also has the capability of propagating decisions made earlier in the processing and in using context to disambiguate a word which can fit into more than one category. For an example of how propagation works, consider a text containing the phrase “Mr. Heemachandra” and a later reference to “Heemachandra.” The surname “Heemachandra” doesn’t appear on a word list of names, but information extraction uses the context of a title (“Mr.”) found to the left of the word as an indication that this phrase refers to a person. That information is propagated as the text is processed, so that, when the system encounters the phrase “Heemachandra” again but without a title, it will be tagged as a PERSON based on the previous categorization.

Suppose that the system encounters the phrase “Heemachandra” for a third time in the text, but this time in the phrase “Heemachandra, Inc.” One option is to propagate the earlier decision that “Heemachandra” is a name, but in this context that would be incorrect. However, the system may have another rule saying “In the pattern *‘X, Inc.’* the phrase *X* refers to an organization.” Patterns always take precedence over word lists and guesses based on propagation, thus allowing appropriate disambiguation in such cases.

While regular expressions are considered to have perfect Precision (i.e., when they tag a phrase, they tag it correctly), they may suffer in terms of Recall (i.e., they may miss phrases that should be tagged because they don’t exactly fit the regular expression). Information extraction systems capture a much wider variety of phrase types, but must be refined in an iterative fashion (i.e., run the system, fix errors, run it again) in order to push towards better and better Precision and Recall. Even well-training information extraction systems are not perfect; one example of an information extraction system using rules but trying to be too clever comes from Andrei Mikheev. His system tagged the phrase

Dairy Queen Cheesecake Factory

as

Dairy <PERSON>Queen Cheesecake</PERSON> Factory.

The reason for using information extraction in spite of its imperfect reliability is that it can tag important information that cannot be tagged by regular expressions. For example, our texts may contain a number (an integer, for example) that could fit into several different regular expressions. Information extraction can choose the correct tag by looking for context such as words like “bty” (here, this an abbreviation for an air defense battery) or “mission number” (here, this refers to the mission number associated with the conduct of an air mission) and can even use lack of relevant context to conclude that the number is not of interest, as it is when the speaker is saying, “He’s starting on his 2nd shift.”

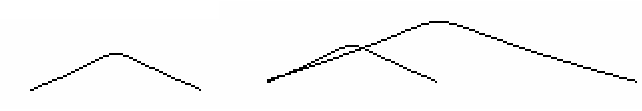
4 The Alembic Information Extraction System

4.1 The *phraser* and rules for combining phrases

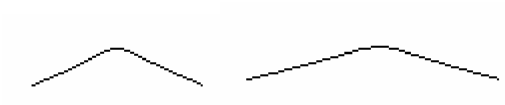
Alembic is a modular system, and several modules must process the data before phrases of interest can be tagged, such as tagging the part of speech of each word and determining where the sentences begin and end. The *phraser* module then chunks the words into phrases such as noun phrases, verb phrases, prepositional phrases and the like:

<S><NP>the dog</NP> <VP>drank <NP>water</NP> </VP></S>

These phrase labels allow the *interpretation rules* to relate actions to their agents, objects, locations, times, etc. The relations between these roles are shown for two examples below:



[$_{\textbf{Agent}}$ The dog] [[$_{\textbf{Action}}$ drank] [$_{\textbf{Object}}$ water]] [$_{\textbf{PP}}$ in [$_{\textbf{Loc}}$ the yard]]

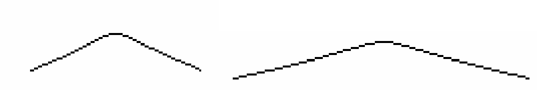


[$_{\textbf{Agent}}$ John] [$_{\textbf{Action}}$ drank in] [$_{\textbf{Theme}}$ the beautiful view]]

The interpretation rules look for an agent for each action plus any appropriate theme/object and any phrases indicating time, place, etc. More complex modifiers such as relative clauses (“the soldier who shot the missile”) and appositives (“Diane Feinstein, the former governor of California”) must also be chunked into phrases and then those phrases must be interpreted according to their relationship with the main event described by the verb.

An example of the type likely to be found in our chat data is shown below:

radar active ivo 42 12 N 012 42 E



[$_{\textbf{ELINT-type}}$ radar] [$_{\textbf{ELINT-state}}$ active] [$_{\textbf{PP}}$ ivo [$_{\textbf{Location}}$ 42 12 N 012 42 E]]

Here the interpretation rules expect ELINTs to have states and locations. The prepositional phrase (PP) indicates an object phrase which is likely to describe a location. A *context rule* then applies as described earlier, marking the Location phrase as an ELINT-Location because it is near other entities which have been tagged as being in the ELINT class.

4.2 Forming *Eventlets*

The final module used in the current version of this chat tool combines a set of entities tagged as members of the same event class into an *eventlet*. Participants in the OIF chats discuss events as they observe them. These chats are cooperative and focused, meaning that the participants want to communicate what they observe to the others, and, as a result, they each describe the aspects of an event that they can observe from their position and with their equipment.

We use a heuristic based on the nature of this type of chat, which is to assume that each chat message contains a partial description of an event, or an *eventlet*. This heuristic allows us to combine the different tagged entities belonging to the same event class into one eventlet, via a template. For example, give the utterance

[Speaker 1] *Al Azah spoonrest is active.*

the system tags the entities as below:

[Speaker 1] <ELINT><Location><Name>*Al Azah*</Name></Location></ELINT> <ELINT><Name>*spoonrest*</Name></ELINT> *is* <ELINT><State>*active*</State></ELINT>

and places them into an eventlet template as follows:

<ELINT><LOCATION><NAME>*Al Azah*</NAME></LOCATION>

        <NAME>*spoonrest*</NAME>

        <STATE>*active*</STATE>

</ELINT>

In future work, we will combine multiple messages which describe eventlets into one larger description of an event. Consider the following example:

[Speaker 1] *Al Azah spoonrest is active.*

[Speaker 2] *Al Azah spoonrest at 422812N 0711647W 1.3 x 0.2 164.4.*

The eventlet creation module will first build one representation for each of the above sentences. It will then look for similarities in the representations in order to determine whether the same topic is under discussion. In this case, both the location name and the ELINT class match. If we say that two representations match if a minimum of two slots match, then these representations match our criterion and can be combined as below:

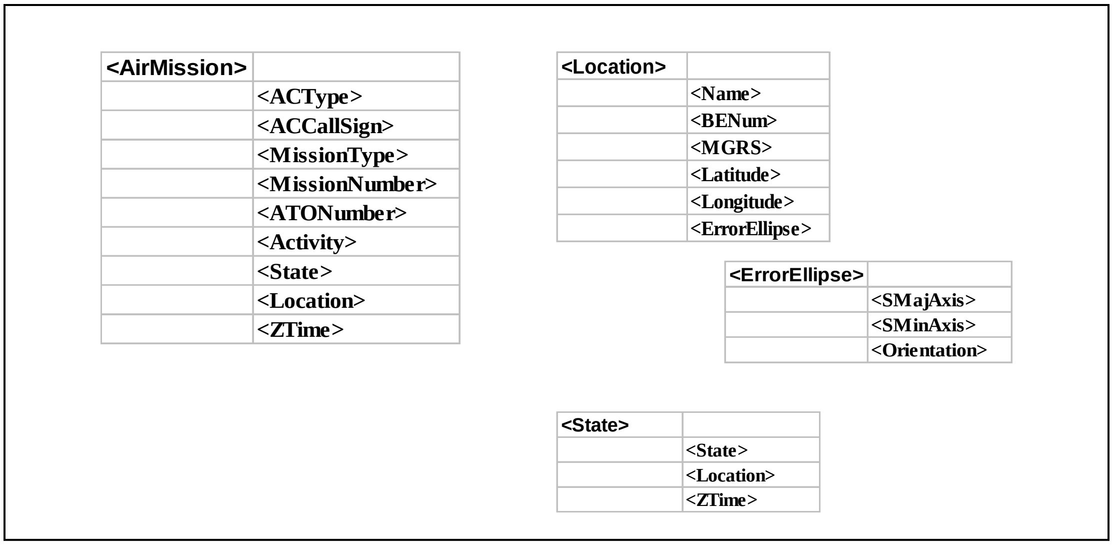
```
<ELINT><LOCATION> <NAME>Al Azah</NAME>
                  <LATITUDE>422812N </LATITUDE>
                  <LONGITUDE>0711647W </LONGITUDE>
                  <ERRORELLIPSE>  <SMAJAXIS>1.3</SMAJAXIS>
                                  <SMINAXIS>0.2</SMINAXIS>
                                  <ORIENTATION>164.4</ORIENTATION>
                  </ERRORELLIPSE>
                  <NAME>spoonrest</NAME>
                  <STATE>active</STATE>
       </LOCATION>
</ELINT>
```

We can change the required number of matching slots required and then test which number gives the best results.

5 Entity Class Development

The entity classes and their attributes whose instances in chat are subject to information extraction by MLPP and Alembic applications were developed based on an analysis of the chat logs collected over a period of time during OIF. This analysis involved the computation of relative frequency of occurrence all for “non-trival” words (i.e., excluding “the,” “and,” etc.) across all chat logs from a particular chat room. Once an entity class was proposed, a set of attributes associated with this entity class was developed, again using relative frequency of occurrence along with knowledge of the appropriate domain. The more than 1000 individual chat logs covered roughly 100 different chat rooms, spanning “themes” such as flight/mission operations, search and rescue, imagery reports and un-manned aerial vehicle (UAV) operations.

Currently, we have defined nine entity classes: <AirDefense>, <AirTarget>, <AirMission>, <ELINT>, <ErrorEllipse>, <GroundTarget>, <Location>, <MASINT> and <Missile>. As an example, consider the <AirMission> entity class and its attributes as illustrated in Figure 2:



<AirMission>	
	<ACType>
	<ACCallSign>
	<MissionType>
	<MissionNumber>
	<ATONumber>
	<Activity>
	<State>
	<Location>
	<ZTime>

<Location>	
	<Name>
	<BENum>
	<MGRS>
	<Latitude>
	<Longitude>
	<ErrorEllipse>

<ErrorEllipse>	
	<SMajAxis>
	<SMinAxis>
	<Orientation>

<State>	
	<State>
	<Location>
	<ZTime>

Figure 2: <AirMission> Entity Class

This class encapsulates information related to the conduct of a friendly air mission, including such attributes as aircraft call sign (<ACCallSign>, mission number (<MissionNumber>) and state (<State>). (This last attribute is actually a supporting entity class, with attributes of its own.)

Word lists contain domain vocabulary which is relatively stable across time, users of chat, and theaters-of-interest. In the case of the <AirMission> class, lists are implemented for the following attributes: aircraft type (<ACType>, mission type <MissionType>, activity <Activity> and state <State>. Below is an excerpt from the <ACType> list:

Designator	Name	Nickname
AH-1	Super Cobra	
A-4	Skyhawk	
OH-6	Cayuse	
AH-6	Little Bird	
A-6	Intruder	
EA-6	Prowler	
A-7	Corsair	
AV-8	Harrier	
A-10, OA-10	Thunderbolt	Warthog or Hog
A-12	Avenger	

Table 1: Subset of Values for <ACType> Attribute

An example of a single chat message referring to an air mission might be:

Honcho 06 has just launched – will be on CAP at 1815z.

Information extraction would then apply the following tags to this chat message:

<AirMission><ACCallSign>*Honcho 06*</ACCallSign></AirMission> *has just* <AirMission><Activity>*launched*</Activity></AirMission> *– will be on* <AirMission><MissionType>*CAP*</MissionType></AirMission> *at* <AirMission><ZTime>*1815z*</ZTime></AirMission>.

This annotation requires Alembic to make use of a word list containing mission types, such as *CAP*. MLPP will recognize the regular expressions *Honcho 06* and *1815z* as <ACCallSign> and <ZTime> attributes, respectively. Finally, in order to determine whether the expression *launched* indicates that an aircraft or a missile was launched, Alembic uses the surrounding context containing <AirMission> tags to make the educated guess that the launch activity is part of the same <AirMission> entity.

6 The Dialog Manager for Alembic

As chat comes into the system through the MLPP application, it has to be segmented into "documents" that Alembic can process; The Dialog Manager fulfills this role. An object of the Dialog Manager class, which is instantiated in the MLPP, collects chat messages and the results of MLPP's information extraction, then uses a Heuristic Class to determine if a document has been completed. This document, which is a contiguous collection of chat messages from a single chat room, is then formatted using HTML and gets sent via HTTP to a Java servlet running in Tomcat. The servlet feeds the document in the form of a 'ChatDocument' object through Alembic and sends its output back to the Dialog Manager, which then parses it and forms a chat alert if certain conditions are satisfied. The resulting 'ChatAlert' object is then placed in the database. Figure 3 illustrates this processing.

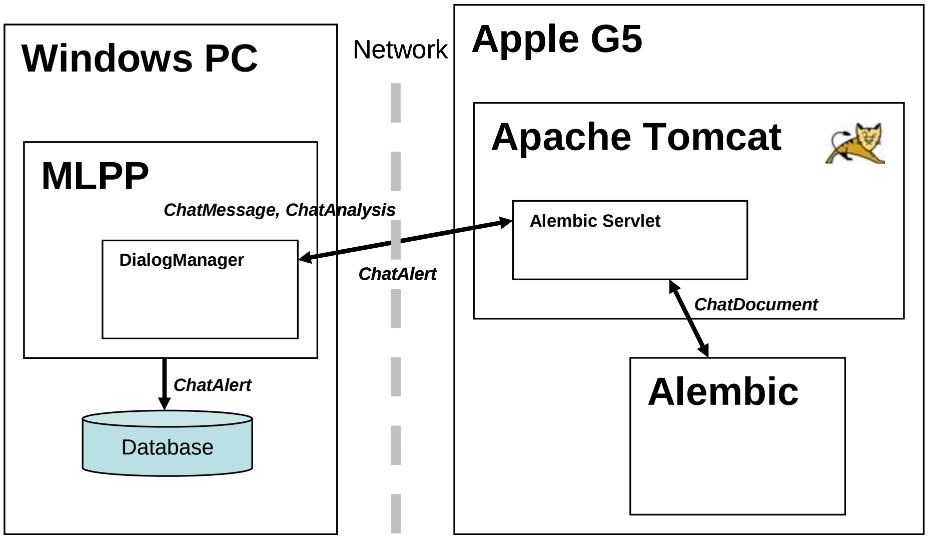


Figure 3: Interaction between MLPP and Alembic Processes

6.1 Alembic Pre-Processor

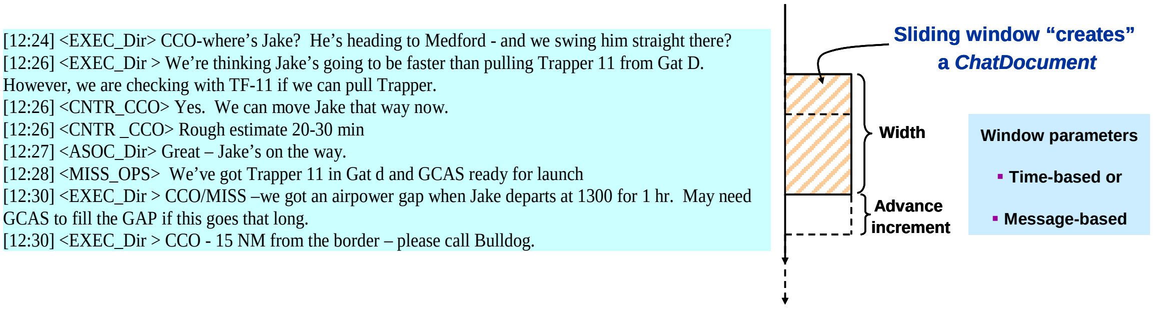
The DialogManager class receives a ‘ChatMessage’ and ‘ChatAnalysis’ object for every chat message that MLPP processes. The ‘ChatMessage’ objects are separated out by chat room and stored. As each message comes in, the Heuristic class processes the vector of chat messages for the chat room that the new message belongs to, and determines if a document is ready or not. The Heuristic class then pulls out the messages for the document, formats them for Alembic using the DocumentCreator class, and then updates the vector of messages by removing messages that won’t be included in the next document. This formatting includes the timestamp and user (i.e., speaker) ID that accompanies the chat message.

6.2 Alembic Post-Processing

The DialogManager class handles processing the output from Alembic as well. When the HTML response comes back from Tomcat, the DialogManager reads it message by message. Each message has the necessary information to connect the newly tagged message back with its original ‘ChatMessage’ and ‘ChatAnalysis’ objects. If any of the messages in the document have tags added by the Alembic processing, a single ‘ChatAlert’ object is created containing all the ‘ChatMessage’ and ‘ChatAnalysis’ objects for the document. The MLPP then checks to see if a new ‘ChatAlert’ object is available from the Dialog Manager, and sends it off to the database if there is.

6.3 DocumentHeuristic Class

A class implementing the DocumentHeuristic interface is used to specify the rules for determining if a document is available for processing. Swapping in a different implementation of this interface changes how the system will determine new documents for Alembic. The implementation we used provides a sliding window of variable width (size) and advance increment. Advancement can be either by time (message times) or by number of messages. Figure 4 illustrates a notional example of how the sliding window creates a ‘ChatDocument’ object for Alembic processing.



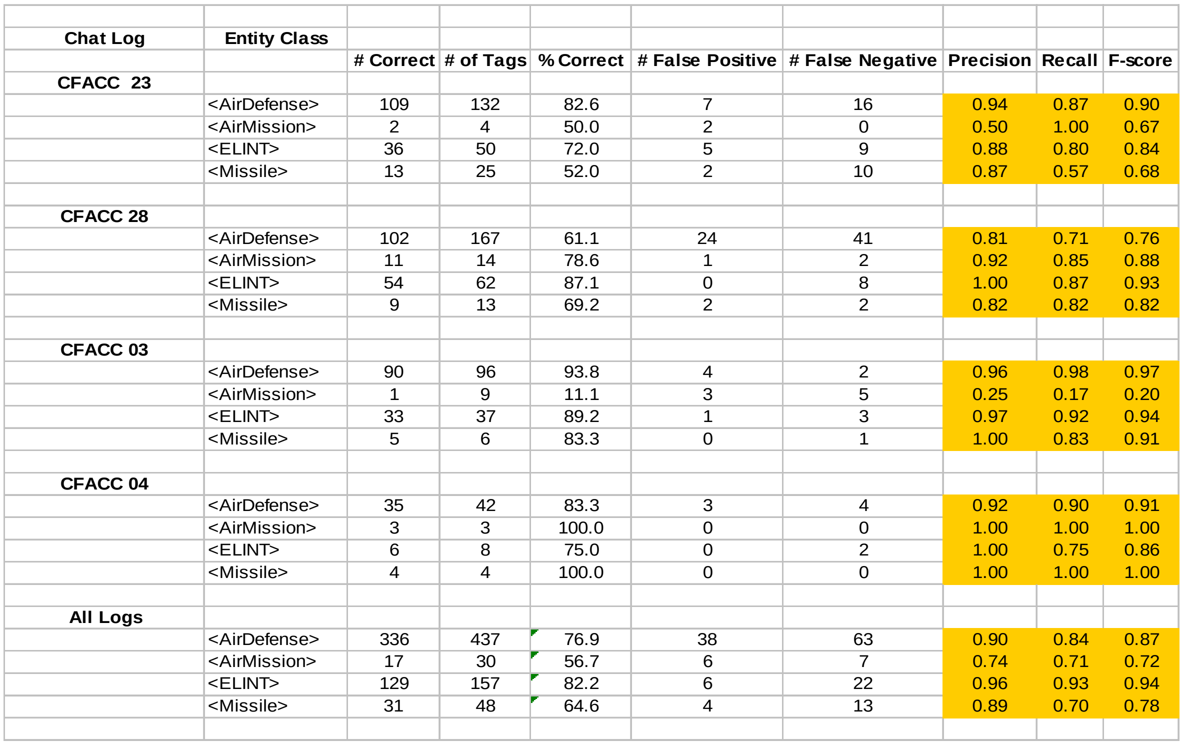
[12:24] <EXEC_Dir> CCO-where’s Jake? He’s heading to Medford - and we swing him straight there?
[12:26] <EXEC_Dir > We’re thinking Jake’s going to be faster than pulling Trapper 11 from Gat D. However, we are checking with TF-11 if we can pull Trapper.
[12:26] <CNTR_CCO> Yes. We can move Jake that way now.
[12:26] <CNTR _CCO> Rough estimate 20-30 min
[12:27] <ASOC_Dir> Great – Jake’s on the way.
[12:28] <MISS_OPS> We’ve got Trapper 11 in Gat d and GCAS ready for launch
[12:30] <EXEC_Dir > CCO/MISS –we got an airpower gap when Jake departs at 1300 for 1 hr. May need GCAS to fill the GAP if this goes that long.
[12:30] <EXEC_Dir > CCO - 15 NM from the border – please call Bulldog.

Figure 4: Example of Dialog Manger’s “Sliding Window”

The simplest case we used was a width of one message and an advance increment of one message, but we also tried using a width of ten minutes with and advance increment of one minute. The current version of the system uses the message-based approach with both width and advance increment set to one for generating documents. These parameter values were chosen since at present the scope of Alembic processing is restricted to a single chat message. Future versions of the system will require that more than one chat message be available to support the Alembic extraction of “events” in chat, as discussed in Section 4.2.

7 Alembic Information Extraction Model Performance

In order to measure the performance of the Alembic information extraction model for these chats, we required a gold standard set of chats which had been annotated by hand for the different entity classes. Because annotating by hand is time-consuming, our initial results are based on a small amount of data from two chat rooms. The development of the gold standard is continuing, and we plan to have more data annotated and more examples from a wider variety of chat rooms in order to show the applicability of this system to different chat topics.



Chat Log	Entity Class								
		# Correct	# of Tags	% Correct	# False Positive	# False Negative	Precision	Recall	F-score
CFACC 23									
	<AirDefense>	109	132	82.6	7	16	0.94	0.87	0.90
	<AirMission>	2	4	50.0	2	0	0.50	1.00	0.67
	<ELINT>	36	50	72.0	5	9	0.88	0.80	0.84
	<Missile>	13	25	52.0	2	10	0.87	0.57	0.68
CFACC 28									
	<AirDefense>	102	167	61.1	24	41	0.81	0.71	0.76
	<AirMission>	11	14	78.6	1	2	0.92	0.85	0.88
	<ELINT>	54	62	87.1	0	8	1.00	0.87	0.93
	<Missile>	9	13	69.2	2	2	0.82	0.82	0.82
CFACC 03									
	<AirDefense>	90	96	93.8	4	2	0.96	0.98	0.97
	<AirMission>	1	9	11.1	3	5	0.25	0.17	0.20
	<ELINT>	33	37	89.2	1	3	0.97	0.92	0.94
	<Missile>	5	6	83.3	0	1	1.00	0.83	0.91
CFACC 04									
	<AirDefense>	35	42	83.3	3	4	0.92	0.90	0.91
	<AirMission>	3	3	100.0	0	0	1.00	1.00	1.00
	<ELINT>	6	8	75.0	0	2	1.00	0.75	0.86
	<Missile>	4	4	100.0	0	0	1.00	1.00	1.00
All Logs									
	<AirDefense>	336	437	76.9	38	63	0.90	0.84	0.87
	<AirMission>	17	30	56.7	6	7	0.74	0.71	0.72
	<ELINT>	129	157	82.2	6	22	0.96	0.93	0.94
	<Missile>	31	48	64.6	4	13	0.89	0.70	0.78

Table 2: The CFACC Chat Logs

The first chat room we looked at was Combined Forces Air Component Commander (CFACC). We had four chats annotated by hand, and worked the data by writing rules for the first chat log, training the system on that log (i.e., refining the rules to improve the score), and then using the next chat log as the test case for the rules. This chat log would give us an idea of how well the system performs on unseen data. We continued this process, training on the first and second logs and testing on the third, etc., and then moved onto the CFLCC chat room.

The results for the CFACC logs are shown in Table 2 above. The results are given in terms of % correct, but the more interesting scores are the F-scores.

The F-score is a weighted measure which combines values for Precision and Recall. Precision is the proportion of phrases which, when tagged, are tagged correctly; Recall is the proportion of phrases which were tagged out of all phrases which should have been tagged. An example of poor precision is observing a missile but tagging it as an airplane; an example of poor recall is blinking when an airplane goes by and not tagging it at all. For our application, it is more important to get high scores in Precision; the user does not want a system that cries wolf, saying that it found an important airplane when it is a missile of no importance.

The table of CFACC chats shows that there were more <AirDefense> and <ELINT> entities present than <AirMission> and <Missile> entities. Given that there were more data for Alembic to use for practice, it is not surprising that the <AirDefense> and <ELINT> tags routinely received higher F-scores.

Chat Log	**Entity Class**								
		# Correct	**# of Tags**	**% Correct**	**# False Positive**	**# False Negative**	**Precision**	**Recall**	**F-score**
CFLCC 14									
	<AirMission>	2	3	66.7	0	1	1.00	0.67	0.80
CFLCC 15									
	<AirMission>	9	14	64.3	1	5	0.90	0.64	0.75
CFLCC 17									
	<AirMission>	14	15	93.3	3	0	0.82	1.00	0.90
CFLCC 21									
	<AirMission>	43	50	86.0	2	6	0.96	0.88	0.91
CFLCC 22									
	<AirMission>	89	121	73.6	9	26	0.91	0.77	0.84
CFLCC 23									
	<AirMission>	5	8	62.5	5	1	0.50	0.83	0.63
CFLCC 25									
	<AirMission>	1	3	33.3	0	2	1.00	0.33	0.50
All Logs									
	<AirMission>	163	214	76.2	20	41	0.88	0.79	0.83

Table 3: The CFLCC Chat Logs

We then took the rules developed for the CFACC logs and used them for the Combined Forces Land Component Commander (CFLCC) logs. The results are shown in Table 3 above.

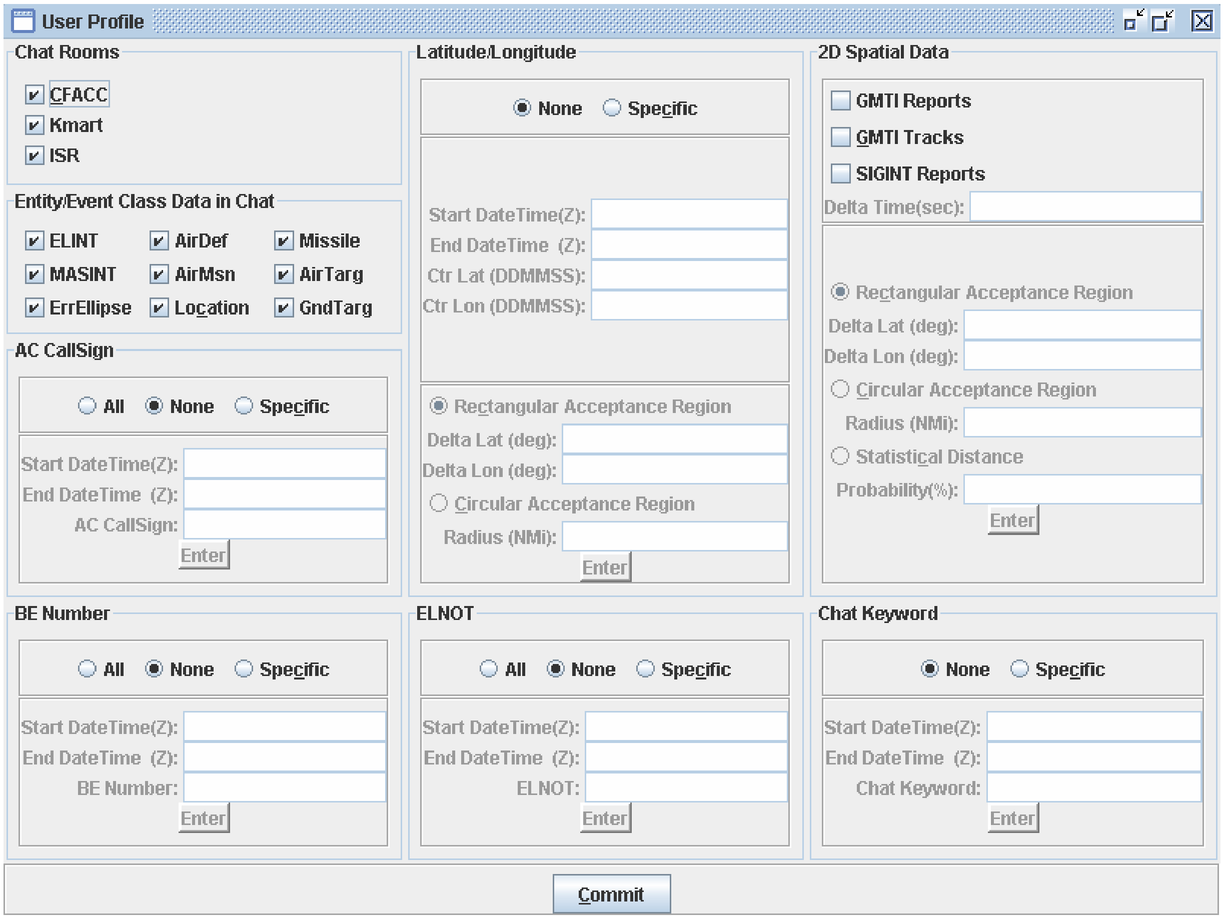
The CFLCC logs contained only instances of <AirMission> entity classes. Given a sudden abundance of data with which to train, the scores for the <AirMission> class quickly became much higher in CFLCC than they were for CFACC.

As we move on to other chat rooms we are keeping an eye open for rooms in which our entity classes are described using terms not previously seen in the current logs so that we can make this system more and more adaptable to diverse chat rooms.

8 User Applications

8.1 UserProfile Java Application

The ‘UserProfile’ Java application allows a user of the system to specify their preferences for information extraction and for the correlation of chat with structured data. Specifically, the conditions set via these preferences will determine whether a ‘ChatAlert’ object that has been generated by MLPP or Alembic processing is converted to a ‘DisplayAlert’ object, which can then be viewed using the ‘UserDisplay’ Java application. Figure 5 illustrates layout of this application’s screen.



User Profile

Chat Rooms

☑ CFACC
☑ Kmart
☑ ISR

Entity/Event Class Data in Chat

☑ ELINT ☑ AirDef ☑ Missile
☑ MASINT ☑ AirMsn ☑ AirTarg
☑ ErrEllipse ☑ Location ☑ GndTarg

AC CallSign

○ All ● None ○ Specific

Start DateTime(Z):
End DateTime (Z):
AC CallSign:
Enter

Latitude/Longitude

● None ○ Specific

Start DateTime(Z):
End DateTime (Z):
Ctr Lat (DDMMSS):
Ctr Lon (DDMMSS):

● Rectangular Acceptance Region
Delta Lat (deg):
Delta Lon (deg):
○ Circular Acceptance Region
Radius (NMi):
Enter

2D Spatial Data

□ GMTI Reports
□ GMTI Tracks
□ SIGINT Reports
Delta Time(sec):

● Rectangular Acceptance Region
Delta Lat (deg):
Delta Lon (deg):
○ Circular Acceptance Region
Radius (NMi):
○ Statistical Distance
Probability(%):
Enter

BE Number

○ All ● None ○ Specific

Start DateTime(Z):
End DateTime (Z):
BE Number:
Enter

ELNOT

○ All ● None ○ Specific

Start DateTime(Z):
End DateTime (Z):
ELNOT:
Enter

Chat Keyword

● None ○ Specific

Start DateTime(Z):
End DateTime (Z):
Chat Keyword:
Enter

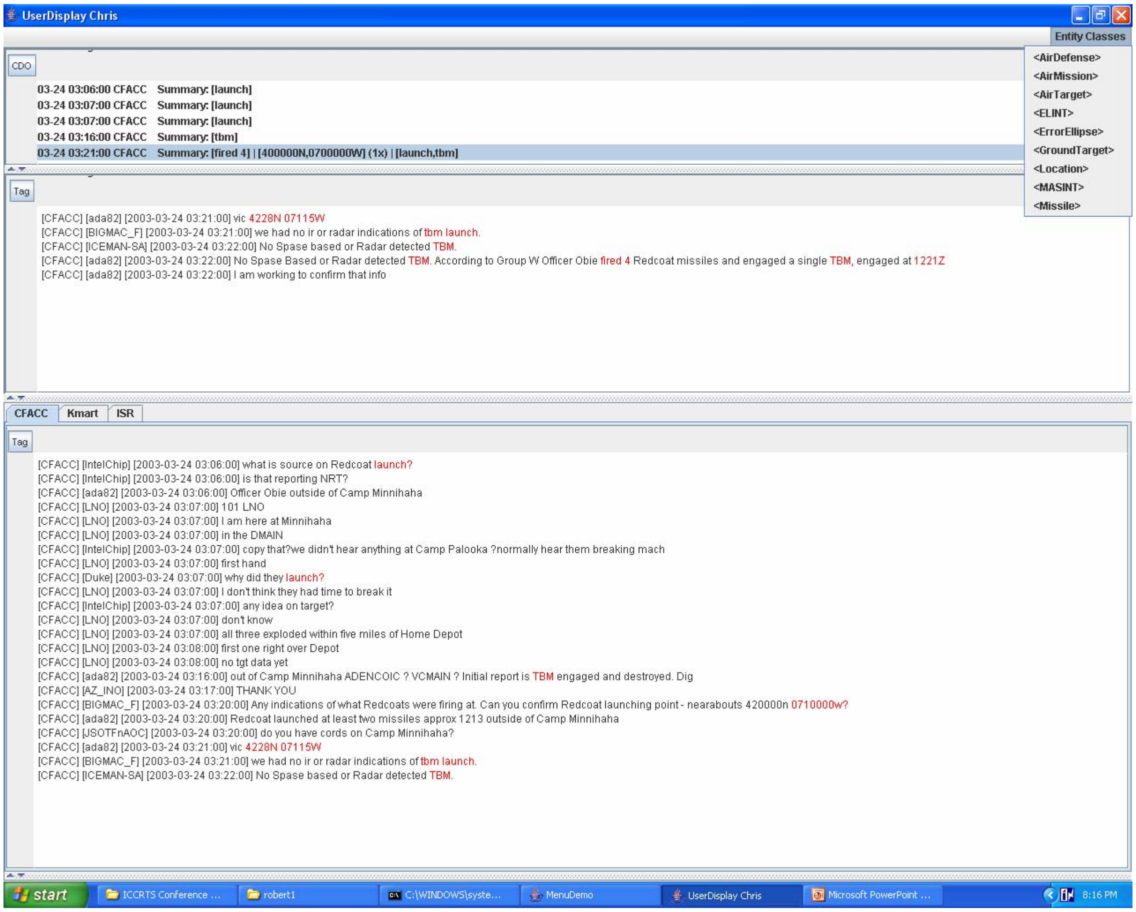
Commit

Figure 5: UserProfile Application Screen

In addition to allowing the user to select chat rooms and entity/event class of interest, the profile allows for the specifying the extraction of semi-structured data types based on a list of values. In the case of aircraft voice call sign, BE number and ELNOT, a temporal filter is applied in the extraction process. In the case of latitude/longitude, a spatial filter (either rectangular or circular) is also applied as part of the extraction process. In the case of chat "keywords" (basically, any word or phrase), a temporal filter is applied.

8.2 UserDisplay Java Application

The 'UserDisplay' Java application allows a user of the system to view the results of the information extraction and the correlation of chat with structured data. What a specific user is able to view depends upon how the individual's user profile was set up. It is important to note that an individual does not have to be involved in a chat session (i.e., using, for example, a Jabber chat client) to run the 'UserDisplay' application. All that is required is that individual have a user profile established and stored in the system database, and the user has access to this database.



```
03-24 03:06:00 CFACC   Summary: [launch]
03-24 03:07:00 CFACC   Summary: [launch]
03-24 03:07:00 CFACC   Summary: [launch]
03-24 03:16:00 CFACC   Summary: [tbm]
03-24 03:21:00 CFACC   Summary: [fired 4] | [400000N,0700000W] (1x) | [launch,tbm]
```

```
[CFACC] [ada82] [2003-03-24 03:21:00] vic 4228N 07115W
[CFACC] [BIGMAC_F] [2003-03-24 03:21:00] we had no ir or radar indications of tbm launch.
[CFACC] [ICEMAN-SA] [2003-03-24 03:22:00] No Spase based or Radar detected TBM.
[CFACC] [ada82] [2003-03-24 03:22:00] No Spase Based or Radar detected TBM. According to Group W Officer Obie fired 4 Redcoat missiles and engaged a single TBM, engaged at 1221Z
[CFACC] [ada82] [2003-03-24 03:22:00] I am working to confirm that info
```

```
[CFACC] [IntelChip] [2003-03-24 03:06:00] what is source on Redcoat launch?
[CFACC] [IntelChip] [2003-03-24 03:06:00] is that reporting NRT?
[CFACC] [ada82] [2003-03-24 03:06:00] Officer Obie outside of Camp Minnihaha
[CFACC] [LNO] [2003-03-24 03:07:00] 101 LNO
[CFACC] [LNO] [2003-03-24 03:07:00] I am here at Minnihaha
[CFACC] [LNO] [2003-03-24 03:07:00] in the DMAIN
[CFACC] [IntelChip] [2003-03-24 03:07:00] copy that?we didn't hear anything at Camp Palooka ?normally hear them breaking mach
[CFACC] [LNO] [2003-03-24 03:07:00] first hand
[CFACC] [Duke] [2003-03-24 03:07:00] why did they launch?
[CFACC] [LNO] [2003-03-24 03:07:00] I don't think they had time to break it
[CFACC] [IntelChip] [2003-03-24 03:07:00] any idea on target?
[CFACC] [LNO] [2003-03-24 03:07:00] don't know
[CFACC] [LNO] [2003-03-24 03:07:00] all three exploded within five miles of Home Depot
[CFACC] [LNO] [2003-03-24 03:08:00] first one right over Depot
[CFACC] [LNO] [2003-03-24 03:08:00] no tgt data yet
[CFACC] [ada82] [2003-03-24 03:16:00] out of Camp Minnihaha ADENCOIC ? VCMAIN ? Initial report is TBM engaged and destroyed. Dig
[CFACC] [AZ_INO] [2003-03-24 03:17:00] THANK YOU
[CFACC] [BIGMAC_F] [2003-03-24 03:20:00] Any indications of what Redcoats were firing at. Can you confirm Redcoat launching point - nearabouts 420000n 0710000w?
[CFACC] [ada82] [2003-03-24 03:20:00] Redcoat launched at least two missiles approx 1213 outside of Camp Minnihaha
[CFACC] [JSOTFnAOC] [2003-03-24 03:20:00] do you have cords on Camp Minnihaha?
[CFACC] [ada82] [2003-03-24 03:21:00] vic 4228N 07115W
[CFACC] [BIGMAC_F] [2003-03-24 03:21:00] we had no ir or radar indications of tbm launch.
[CFACC] [ICEMAN-SA] [2003-03-24 03:22:00] No Spase based or Radar detected TBM.
```

Figure 6: UserDisplay Application Screen

Figure 6 illustrates layout of this application's screen. It is divided into three sections. The top third displays the occurrence of 'DisplayAlert' objects; that is, 'ChatAlert' objects from the Chat Database that satisfied this user's profile (here, the user is "Chris"). An indication of an alert is accompanied by the date, timestamp, chat room and a one-line "summary" of the alert. The middle third displays the details of any selected alert (selection of an alert using a mouse click is indicated in blue). The chat that was extracted and resulted in the generation of this alert is highlighted in red. Lastly, the bottom third of the display contains all chat generated in a selected chat room (here, the name of the chat room is "CFACC").

9 Ongoing Research

In addition to the continued development and testing of an Alembic information extraction model for entity class data, we plan to combine the eventlets we find in a chat into full events, as described in Section 4.2, above.

A second area of ongoing research concerns the correlation of entity and event class data across multiple chat rooms. Correlation of entities or events extracted by Alembic will be based on a measure of the pair-wise similarity. A modified form of Gower's General Similarity Coefficient (GSC)[3] will be used as this measure of similarity. Figure 7 illustrates how Gower's GSC in its standard form may be used to compute similarity between two objects, "i" and "j":

$$S_{ij} = \frac{\sum_{k=1}^{n} w_{ijk} s_{ijk}}{\sum_{k=1}^{n} w_{ijk}}$$

n = number of class attributes
w_{ijk} = a weighting factor
s_{ijk} = similarity measure, dependent on type of attribute, e.g.,

$$s_{ijk} = 1 - \left| x_{ik} - x_{jk} \right| / (x_k^{\max} - x_k^{\min})$$

for quantitative attributes

Figure 7: Gower's General Similarity Coefficient

This modified version of Gower's GSC will take into account (among other things) the notion of similarity between related entity classes. For example, instances <ELINT> and <AirDefense> entity classes are at times related since the activity of radars associated with air defense systems may be detected and reported by ELINT assets.

A similarity matrix will be computed based on a set of Alembic-extracted entities and events from different chat rooms. As an example, consider four instances of entity and event classes from chat room "A" (numbered 1, 2, 3, 4), and four instances from chat room "B" (numbered 5, 6, 7, 8). Figure 8 illustrates a matrix containing Gower GSC similarities between extracted entities or events ("objects") in different chat rooms.

		1	2	3	4	5	6	7	8
1		1	0	0	0	0.9	0.3	0.55	0
2		0	1	0	0	0	0.85	0	0.4
3		0	0	1	0	0	0	0	0
4		0	0	0	1	0	0	0	0
5		0.9	0	0	0	1	0	0	0
6		0.3	0.85	0	0	0	1	0	0
7		0.55	0	0	0	0	0	1	0
8		0	0.4	0	0	0	0	0	1

Figure 8: Example Similarity Matrix

From this similarity matrix, either distinct pairs or groupings (i.e., "clusters") of correlated entities and events will be identified and presented to the user. In the case pairings of objects, a threshold value for minimum similarity will be used to prohibit the

[3] "A General Coefficient of Similarities and Some of Its Properties," J.C. Gower, **Biometrics**, Vol. 27, No. 4, Dec., 1971, pp. 857-871

weakest correlations from being presented to the user. In this simple example above, a threshold value of 0.80 (i.e., 80% similarity) would result in a pairing of objects 1 and 5 and objects 2 and 6 using a greedy nearest-neighbor approach to pairing. In general, however, the similarity matrix will be more complex, resulting in potential groupings and not just pairings of objects. As such, an agglomerative hierarchical clustering algorithm will be implemented and which, as a default, will be applied to the similarity matrix. The dendrogram (i.e., tree structure) produced by this hierarchical clustering algorithm will be automatically pruned based on heuristics and a minimum level of group similarity. Therefore, it will not be necessary for the user to get involved in this process.

Lastly, while the above example is based on similarities between entities and events from different chat rooms, there is no reason why the correlation process as described cannot be carried out both within and across chat rooms.

10 Summary

We have demonstrated the accurate extraction of entity class data (e.g., data pertaining to air mission, air defense and missile entities) from military chat rooms using MITRE's Alembic information extraction system. The success of our approach is due to two factors: First, our use of a rule-based approach to information extraction (i.e., use of lists and context rules for each entity class of interest) makes it possible to capture much of the jargon, abbreviations and acronyms present in military-style chat. Second, the nature of the chat itself, which was cooperative and focused, thus allowing us to use the heuristic that one chat message would be focused on describing a single event.

We have developed a Java-based software system that supports chat exploitation by providing a capability to store, process and display chat and the results of performing information extraction on chat. We have developed a basic set of entity classes which have proven useful in gathering information concerning events of interest, and we plan both to expand this set of entity classes and to make use of them in finding related events in other chat rooms. Central to this system is the development of tailored views of chat based on user-specified preferences, giving the system the means to deliver to the users only the chat alerts specific to their interests. The advantage of having a list of these alerts automatically generated is that the user is less likely to miss an important event while taking a quick break or having his attention focused on another chat room.

We performed our current research with a specific set of Air Force programs in mind: Air Operations Center (AOC) Weapons System; US Strategic Command (STRATCOM) operations centers; and the Airborne Warning and Control System (AWACS) airborne platform. Note that while we have developed entity class models for information in chat based on the entities and events of greatest interest in air and ground operations, our approach is extensible to DoD enterprises. For example, the development of information extraction models for logistics, space, and specific intelligence enterprises is possible given access to their types of chat and domain expertise.