

Mining Chat-room Conversations for Social and Semantic Interactions

Faisal M. Khan, Todd A. Fisher, Lori Shuler, Tianhao Wu and William M. Pottenger
Computer Science and Engineering, Lehigh University
{fmk2, taf2, lase, tiw2, billp}@cse.lehigh.edu
LU-CSE-02-011

ABSTRACT

The ephemeral nature of human communication via networks today poses interesting and challenging problems for information technologists. The Intelink intelligence network, for example, has a need to monitor chat-room conversations to ensure the integrity of sensitive data being transmitted via the network. However, the sheer volume of communication in venues such as email, newsgroups, and chat precludes manual techniques of information management. It has been estimated that over 430 million instant messages, for example, are exchanged each day on the America Online network [3]. Although a not insignificant fraction of such data may be temporarily archived (e.g., newsgroups), no systematic mechanisms exist for accumulating these artifacts of communication in a form that lends itself to the construction of models of semantics [12]. In essence, dynamic techniques of analysis are needed if textual data of this nature is to be effectively mined.

This article reports our progress in developing a text mining tool for analysis of chat-room conversations. Central to our efforts is the development of functionality to answer questions such as "What topics are being discussed in a chat-room?", "Who is discussing which topics?" and "Who is interacting with whom?" The objective of our research is to develop technology that can automatically identify such patterns of interaction in both social and semantic terms. In this article we report our preliminary findings in identifying threads of conversation in multi-topic, multi-person chat-rooms. We have achieved promising results in terms of precision and recall by employing pattern recognition techniques based on finite state automata. We also report the design of our approach to building models of social and semantic interactions based on our HDDITM text mining infrastructure [13].

1.0 Introduction

The goal of the project is to develop a computational approach towards understanding social and semantic interactions in textual mediums. Chat-room conversation has been identified as the communication medium of interest due to its increasing popularity and the need for research in the area. The 430 million daily instant messages on AOL's network alone provide a treasure trove of knowledge [3]. Chat conversation is radically different from various other mediums due to its often informal

nature. Existing text mining techniques rely on more structured, formal corpuses containing research papers, abstracts, reports, etc. Approaches toward understanding the dynamics of chat conversation are limited, and as usage grows the need for automated analysis increases. Due to the dynamic nature of chat conversations, dynamic modeling of social interactions and their contextual topics is a genuine research challenge.

The social and semantic relationships extracted from chat conversations can lead to a better understanding of human relations and interactions. Models can be used to understand social clusters and conversational content automatically, a process otherwise involving a significant commitment of manual effort.

This research is being conducted at the behest of the Intelink intelligence network. Intelink is a secure military communications channel used for important and critical exchanges of information. Intelink's goal is to monitor chat conversation over the network and map relationships between employees and their topics of conversation to determine the appropriateness of usage and the effectiveness of the communication network. They are interested in information such as the frequency of employee communication, the topics discussed, conversational participants and the focus of the conversations.

This research has, however, applications beyond the scope of Intelink's needs. In fact, the techniques under development apply to any organization with an internal communication network, and even to Internet users in general. Any patron of chat services such as AOL Instant Messenger (AIM) and IRC (Internet Relay Chat) can benefit from utilizing such a tool.

2.0 Related Work

Understanding social and semantic relationships independent of one another is not a new field. Much significant Social Science research and Information Retrieval research has been conducted in these two fields. We thus do not attempt to survey the literature of social science or IR research here, but rather give a sampling of recent research that is related to our efforts in text mining of chat data. Extracting the information from a chaotic, unstructured forum such as chat to form combined social and semantic models of interaction is however a new area of work.

The Galaxy of News [14] project at the MIT Media Lab is an example of early related work in the area of semantic (not social) modeling. It is a system

to visualize independent documents and form relationships between them, specifically between news stories. It employed a process to parse the content of news stories and develop weighted relationships between them. This would create a knowledge base of relational networks that could be used to traverse similar news stories, and permit visual interaction with the user. The TDT (Topic Detection and Tracking) research efforts organized by NIST also fall under this category [18].

The Intelligent Network News Reader, HISHO [11] searches through linked news stories for common ones with no need of user input for relevant information. It forms topic clusters of articles, and even keeps track of so called “topic branching articles” for an increased range of semantic connectivity.

The Conversation Map [15] is another product of the MIT Media Laboratory that is more closely related to our work. The system is a Usenet Newsgroup browser that analyzes Usenet postings and creates social and semantic networks of the postings. Due to the inherent threaded nature of Usenet postings, however, social relationships are easily modeled. It provides a graphical interface to track and observe social and semantic relationships from topic to topic.

Upon studying the various factors influencing chat efficiency, The Virtual Worlds Group at Microsoft has developed the Status Client, “a prototype of an interface that shows the status of each user, as determined by keyboard activity” [16]. Through providing more information to users, such as what each member is currently typing, the goal is to simplify the chat environment. A reduction has been observed in the number of posting required to clear out-of-turn and misplaced postings. The issue of chat history loss has been addressed by a “multi-channel timeline” displaying postings in a real time interface similar to a timeline, with a separate row for each user’s postings. This system, however, is focused on improving chat interfaces rather than on automatic analysis of social and semantic interactions.

Threaded Text [3] is a chat environment developed to address confusion among conversational threads. A chat room is organized into a series of trees, each tree representing a separate thread. Thread boundaries are easily recognizable since they are at the root of each tree. It has led to the observation that some of the fundamental reasons of chat’s chaotic nature is due to the lack of knowledge of “turns-in-progress,” “listening-in-progress” and social and historical context. This effort too, however, is focused on improving chat interfaces using manual techniques. In our survey of the literature we have not come across any work with the goal of automatic modeling of both social and semantic relations within chat data.

3.0 Modeling Social and Semantic Relations

The application under development at Lehigh University to model social and semantic interactions is the Social Semantic Builder (SSB). The SSB is a

relational modeling tool that utilizes HDDI™ [13][2][1] text mining infrastructure, and models inherent relationships between distinct conceptual and/or behavioral abstractions. The purpose of the SSB is to determine, analyze, and model the relationships and interactions between these abstract relational entities.

As an example, consider the domain of research papers. Abstractions within this application field include authors of the papers and the concepts they explore. Corpuses are constructed and used to cluster examples of the abstract entities according to their co-relational properties. In this example, two separate groups of clusters are created, one of authors who write together and another one of concepts that are similar within the document space. This allows questions such as “What topics are being discussed?”, “Who is discussing which topics?” and “Who is authoring with whom?” to be answered.

In this example the SSB uses the information retrieved from the generated clusters to create a meta-level co-relational structure modeling the associations between research paper authors and their conceptual content. The generated network of abstract entities and their relationships can then be used to analyze and discover previously unknown relationships between various abstract entries.

Although we have presented the SSB in the context of research article authors and content, social and semantic modeling can take place in any domain involving multiple authors and content (such as chat).

3.1 Approach to Modeling Semantic Relations

Within the semantic domain, our methodology utilizes *contextual transitivity* in the co-occurrence relation to identify regions of high-density concept clusters or subtopic regions of *semantic locality* [2]. The regions of semantic locality are based on higher-orders of co-occurrence in the co-occurrence relation [19]. These regions consist of clusters of concepts that commonly appear together and collectively create a *knowledge neighborhood*. The premise is that grouping similar concepts together leads to increased effectiveness and efficiency in query search and retrieval [9].

3.2 Approach to Modeling Social Relations

Desiring to employ the same process for developing social relationships as semantics ones, our methodology for modeling social relations utilizes the same clustering algorithm as the one for semantic modeling. Although the modeling of social relations is done on a different type of relation, it is one that is semantically expressed in text. The SSB is designed to extract and model any relation as long as the relation is semantically expressible in text. For example, a social relationship that can be modeled is *co-authorship* – in this case, authors are clustered together based on how often they write books, papers, articles, etc. together. Semantically analyzing co-

authorship involves treating author names as features in a collection of texts. These features are extracted, and clusters of author names are formed, similar to the manner in which clusters of concepts are formed as described above in section 3.1.

In the following section we provide additional detail of the overall modeling process employed in the SSB.

3.3 Overview of the Modeling Process

Throughout this section we will continue the example given previously that involves modeling social and semantic interactions between scientific research authors and articles.

Our methodology involves creating an overall Social/Semantic model consisting of three sub-models. In the example given previously, the three sub-models are an Author model, a Content model, and a Translation model that serves as a link between the first two. These sub-models are internal nodes of the Social Semantic Model, and are represented as HDDI™ [13] nodes.

The Content sub-model consists of concept clusters derived from title and abstract fields of the documents processed by the HDDI™ [13] system. To identify regions of high-density concept clusters, later grouped to form sub-models, higher order co-occurrence relationships are contextually utilized [19]. The partitioning algorithm, HDDI™'s sLoc [2], uses contextual transitivity in conjunction with co-occurrence modeling to determine groupings of similar concepts, described as regions of semantic locality. sLoc, constrained by a statistically determined heuristic threshold, extends the similarity relation to form a model based on the distribution and contextual structure of similarities in the collection.

As noted earlier, the model in theory is able to analyze relationships between any semantically expressible relations. In our running example, the Author sub-model is developed to model specific social relationships, in this case co-authorship. Since it is still a semantically expressible relation, the Content sub-model procedure can be reused. Co-authors form clusters of authors based on their frequency of collaborating together in writing books, papers, articles, etc. Author names are treated as features whose co-occurrence is examined in the collection to form co-authorship clusters.

The Translation sub-model maps the physical relationships between authors and the content they write about by clustering them together. It is a meta-model that is built using the two sub-models.

In our current implementation, once the three HDDI™ nodes (sub-models) are constructed, a similarity function is applied between the Author node and the Translation node and between the Content node and the Translation node. This process results in probability mappings between the three sub-models. These mappings represent the probabilistic strength (through the Translation node) between a cluster in the Author node and its corresponding cluster(s) within

the Content node, and vice-versa. The connections between authors and their content represent probabilities that a particular author or group of co-authors will write about a particular topic, or that an author or group of co-authors will write about a particular topic. These weighted paths through the Translation node, between Author and Content nodes, create a Social Semantic Model.

4.0 Application Domain: Chat Conversations

The Social Semantic Builder is a utility with a variety of applications. We have discussed at length authors and research papers, but there are also students and courses, journalists and newspaper articles, etc. The SSB is designed in such a fashion that its general relational structure could be deployed for mapping between any type of entities. Each application has its own particular issues that need to be addressed, and in this work of course we address those issues in the context of the analysis of chat conversations.

Some of the questions particular to chat are "Who are the participants in a particular conversation?" "What are they talking about?" "How focused is their conversation?" "How are the participants socially interacting?" "What forms of language do they use to express themselves?"

A user (such as Intelink) interested in such chat relational models would be able to use the SSB for extracting such information. Chat conversational documents would be input into the SSB, and it would create models of chat participants and conversational topics. A user then could use the models to associate topics with participants, observing which participants discussed a particular topic. Information such as which participants were involved in discussions together, and the topic of those discussions would be readily available. The basic questions of "What topics are being discussed in a chat-room?", "Who is discussing which topics?" and "Who is interacting with whom?" can be answered.

4.1 Chat Input Issues

The SSB accepts input for analysis in documents with XML formatted content and author fields, utilizing the HDDI™ infrastructure for processing the documents. At present, the HDDI™ System processes the input and then constructs a collection of statistics for analysis. The models for the social and semantic domains are then created and linked together by the Social Semantic Model Builder.

Some application domains easily map to this input format such as research papers and newsgroup postings. The authors are identified at the beginning, and the body of the document or posting can be tagged as content. Chat conversation is not so structured. Chat is often a continuous medium with users entering and leaving a given chat room. Furthermore, even though a chat room may have many users logged in, not all of them may be participating. Of those users who are involved, they do not all participate in the

same discussion with all the other users. Often there are several conversations simultaneously taking place between users – a single participant may also be involved in multiple conversations at once. It is an extremely chaotic environment and at first glance seems to lack consistent structure.

All such chat conversations are interlaced throughout multiple postings; extracting “authors” and their content into single cohesive units for input to the SSB is a daunting task. Furthermore, in our research we have observed that there are numerous categories of chat. Factors such as number of participants, the topic(s) of chat, the familiarity of users with each other, etc. lead to radically different conversation styles. If a sampled session is of acquaintances discussing a common topic, for example, the conversation flow tends to be informal with little attention to grammar. If the session is, for instance, a help session or a discussion medium for a focused topic in which the users don’t know one another, the conversation is focused and formal and a larger vocabulary is used.

There are various other chat specific issues that we found must be considered. In chat conversation there are often misspelled words, or chat specific terms such as “brb, u, r, 2” (for “too” or “to”), and “c,” used. Emotion icons such as “:), :P, ;(“ are used extensively throughout chat conversations as well. Decoding these terms is crucial since within the chat-room conversation domain they hold semantic importance.

Our current approach is to develop a methodology for formatting chat conversational data into a collection of items for SSB input. Each item consists of postings relevant to a single topic, and the users who participated in that topic. Within this framework, the names/screen names of the posters identify authors and the items identify content. A co-authorship relationship is defined between users based on the content of their postings. Separate content and author clusters are created based on the co-occurrence of semantic features such as screen names and noun phrases.

5.0 Approach

Our current goal is to develop a methodology for mapping chat conversation information to input for the SSB. Mapping names isn’t a difficult task because screen names can be used for identifying authors in a chat conversation, and we employ a database of names compiled from the 1990 United States Census as a means of accurately identifying names in the conversation. The difficult task is to develop a high-precision technique for identifying and extracting items or topics from chat-room conversation. We made two attempts before settling on our current approach. In what follows we will briefly discuss our initial attempts, and then describe our current approach.

5.1 Method 1: Chronological Definition of Items

Our initial premise was that at any given point in time a chat conversation would be focused on one topic. Thus using a heuristically determined time period, a conversation could be separated into segments, each being an individual document.

Early in development though, this premise proved flawed – through manual study of actual chat conversations we noticed that conversation isn’t contiguous as expected but rather fluctuates rapidly. Attempting to use a time interval for identification of an item would not result in well-defined, topically distinct items.

5.2 Method 2: Using a Topic Segmentor

Our second approach involved the utilization of Marti A. Hearst’s Text Tiling algorithm [7] to locate topic boundaries within expository text. The TextTiling algorithm would report topic (or segment) boundaries, and each segment could be classified as an item consisting of a group of co-authors. This would result in acceptable SSB input format.

The motivation for this approach was that the TextTiling algorithm is designed to separate expository text into paragraphs, and our process would abstract each paragraph of a conversation into a topic. Unfortunately, the algorithm was designed for structured data such as research articles and other more organized corpuses. Within chat however, multiple topics occur within a set of contiguous postings. This resulted in the TextTiling algorithm returning unpredictable results since the topics are fragmented in the original data.

5.2.2 Conversational flow identified as Threads

Analyses of these issues led us to classify topics in chat conversation as threads. Chat conversation in a chat medium flows and is intertwined as if there were multiple threads of conversation. Often, at a given instant, there are multiple conversations simultaneously occurring that result in multiple threads interweaved. Even the participants in the chat conversations have trouble identifying speakers and remembering who said what when. It has also been shown that within a particular room or channel, there are often multiple conversations or ‘threads’ going on at the same time, and a single user often participates in multiple threads at the same time [16].

5.2.3 Unraveling Threads with a Topic Segmentor

After identification of chat structure as threads, our approach was adapted to a process of extracting threads and storing each thread of conversation as a document. We employed Freddy Choi’s [5] C99 algorithm in order to accurately represent a single set of postings made by a single author as corresponding to a single topic. The method would use frequency counts of most common words in order to create vectors for each segment with similarity between

segments based on a cosine measure. Essentially the method involves determining arc weights for inter-segment relationships based on how many words in each segment are similar.

The goal of organizing data in this manner was to effectively unwind threads, and classify each thread as a separate item concerning a particular topic. Then we planned to employ a technique to determine which authors belonged to which content threads. Unfortunately our results indicated that in separating contents into groups, much information was lost. This is due to effects noted in Section 5.2.2 such as the fact that single authors often simultaneously participate in multiple conversations. Furthermore, some chat conversations lacked the organization and richness in unique words this method relies on.

5.3 Method 3: Adaptive Techniques for Threaded Conversation: Identifying Thread Starts

The initial methods failed primarily because they were designed for corpuses with more regular semantic structure. The discovery that chat conversation is modeled as threads did however lead us to a new technique for topic identification.

Our revised methodology starts by taking individual threads within the conversation and classifying them as either thread starts (the beginning of a topic) or non-thread starts. Our goal is to process each non-thread start and attach it to its appropriate thread-start.

In what follows we discuss our techniques for identifying thread starts and the validation of this approach based on our experimental results. Research is continuing at Lehigh University to attach non-thread starts to their respective thread starts.

5.3.1 Patterns in Thread Starts

Our methodology for identifying thread starts relies on certain relevant patterns. Human experts studying sampled chat conversations for common occurrences in thread starts developed these patterns. Most individual patterns do not indicate a thread start, but in combination with various others they do.

5.3.2 Positive Patterns

Our technique utilizes two categories of patterns. The first is a set of patterns used to positively identify thread starts. They were developed by human expert observations on common characteristics of thread starts. For instance, the most common manner of starting a new thread is by asking a question. Other participants in the chat forum respond, and thus a conversation develops. Our approach currently has eight unique positive patterns overall (see Figure 1).

Chat users often start new threads of conversations by greeting someone or using an attention getter such as “hi, hey”, etc. They then follow with a topic directed at the person whose attention they were trying to obtain. Another manner

to start a new thread is by addressing an individual with their name. Each such method is represented by a positive pattern that covers all occurrences.

One positive pattern usually isn’t enough to classify a posting as a thread start. Our methodology emphasizes combinations of patterns to determine thread starts. Postings such as “don’t you like that movie?”, “Bob that is interesting”, “hey I like that” fit a positive pattern but are not thread starts; they are rather, somewhere within the conversational thread.

Our approach relies on the fact that combinations of these patterns usually indicate a thread start. For example postings such as “Hi Bob, did you like that movie?”, “Joe, Where did you go for vacation?”, etc. are more likely thread starts. They indicate a new topic that will most likely be the subject of various postings creating a thread.

5.3.3 Negative Patterns

Positive patterns aren’t enough to accurately identify thread starts. Employing solely positive patterns results in many false positives – i.e., postings identified as thread starts that really aren’t. Consequently, our methodology also utilizes negative patterns to prune the false positives from the positive pattern results.

Negative patterns are based on characteristics observed by human experts as lacking in thread starts, or characteristics in postings that aren’t thread starts. At present we’ve developed two general types of negative patterns that are broken out into 13 sub-patterns (see Figure 1). The first are used to identify posting with anaphoric relations such as “that, she, he,” etc. These postings employ the anaphoric relation to refer to content previously mentioned. They are rarely used to begin new threads, but rather continue previous ones.

The second general type of negative pattern relates to posting length. If a posting is short it usually doesn’t indicate a thread start. Thread starts are longer postings since they may introduce a new topic that participants respond to. For example, short responses such as “yeah, I agree, yes, no” etc. are used to reply to an existing thread, not to continue a new one. This is intuitive – it is difficult to start a new thread with a short post. Our approach utilizes various size patterns to prune the positive pattern results of false positives.

Pattern Number	Description	Attribute
1.	A single question mark is the last character in a posting	Positive
2.	“Who, what, when, where, how, why” is a single word anywhere in a posting.	Positive
3.	A single question mark anywhere in a posting	Positive
4.	One of “Do Did Does Are Is Were Was Shall Will Can Could Would Have Had Has” is a single word in anywhere	Positive

	of a posting	
5.	There is a NAME in the posting	Positive
6.	One of “hi, hey, yo, hello” is the beginning of a posting	Positive
7.	One of “hi, hey, yo, hello” is followed by at least one other word in a posting	Positive
8.	There is at least one “...” in a posting	Positive
9.	“he” is a single word in anywhere of a posting.	Negative
10.	“she” is a single word in anywhere of a posting.	Negative
11.	“him” is a single word in anywhere of a posting.	Negative
12.	“her” is a single word in anywhere of a posting.	Negative
13.	“it” is a single word in anywhere of a posting.	Negative
14.	“they” is a single word in anywhere of a posting.	Negative
15.	“them” is a single word in anywhere of a posting	Negative
16.	“those” is a single word in anywhere of a posting	Negative
17.	“these” is a single word in anywhere of a posting.	Negative
18.	“this” is a single word in anywhere of a posting.	Negative
19.	One or no words in a posting	Negative
20.	Only two words in a posting	Negative
21.	Only three words in a posting	Negative

Figure 1: Patterns Identifying Thread Starts

5.3.4 Regular Expressions in Pattern Recognition

Our approach in recognizing the patterns discussed above involves the use of finite state automata in the form of regular expressions. We have developed an algorithm that utilizes regular expressions to implement the patterns automatically and extract posting numbers that match the patterns. At this stage we are only employing patterns that can easily be implemented as regular expressions matched against a single posting. For example, the regular expression for the pattern “A single question mark is the last character in a posting” is $\{ALPHANUMSYM\}^*“?”$, where $ALPHANUMSYM$ is any character. If a regular expression matches a posting, its number is recorded for later processing.

Our methodology implements regular expressions that match the patterns in Figure 1 exactly. Each regular expression is tested to ensure that it only returns results for the desired pattern, and no others. Our process in developing a regular expression for a pattern involves initially writing a regular expression manually. Since the definition of the pattern is known, it is relatively straightforward to develop a regular expression to cover most, if not all cases of the pattern. A test set is applied to the regular expression.

The results are compared with results from a manually developed training set of patterns. Using the evaluation metrics precision and recall, we ensure that the regular expression results match the human expert results. Appendix A shows the results of training and testing our regular expression matching algorithms. Precision and recall are 100% in most cases.

5.3.5 Combinational Pattern Matching

Once results of regular expression matching have been generated, our methodology uses an algorithm to create various combinations of patterns. The algorithm creates combinations of sizes 1 to N, where N is the total number of patterns. For example, listed in Figure 1 are 21 patterns. The combinatorial program creates all the combinations of size 1, 2, 3, ...N possible from these 21 patterns.

The number of combinations possible is n choose s , where

$$n \text{ choose } s = n! / (s!(n-s)!)$$

n is the number of patterns and s is the size of the combination. Thus, for the 21 patterns in Figure 1 the total number of combinations would be $21 \text{ choose } 1 + 21 \text{ choose } 2 + 21 \text{ choose } 3 + \dots + 21 \text{ choose } 21$, or approximately 2 million combinations.

In each combination, positive and negative patterns are combined separately. Then the set of all combined positive patterns is pruned by the set of all combined negative patterns. In Section 6 we discuss which pattern combinations work best for detecting thread starts.

5.3.6 And/Or Operations in Pattern Matching

In the course of our research we decided that negative and positive patterns could be combined using either the logical AND or logical OR operators. With the AND operator, only postings common to both patterns would be in the combined set whereas with the OR operator, posting in either pattern would be in the combined set. As our results in Section 6 show, employing the OR operator when combining both positive and negative patterns results in the best performance overall in terms of precision and recall. Our process applies the OR operator to positive patterns to get the greatest number of postings that could possibly be thread starts, then the negative patterns (also using the OR operator) are applied to remove the greatest number of false positives, thereby yielding an outcome with high precision and recall.

6.0 Results

Chat conversations are often dissimilar, with a wide range of participants, and the conversational topics may be focused or fragmented and are often changing rapidly. In order to obtain preliminary results in validating our methodology, we applied it on two different chat conversational styles. Due to the classified nature of the Intelink data, however, we

have constructed our own test data sets. We describe this process in what follows.

6.1 Test Set Construction

Testing was performed on two main samples of chat conversation. The first was a slightly focused conversation (topics changed, but during discussion, conversation was focused) on AOL Instant Messenger (AIM) between three researchers; it consisted of approximately 500 postings and is referred to in Section 6 on results as Data Set 1. The second sample was a larger, unfocused AIM conversation with a total of 12 participants and is referred to as Data Set 2. This latter sample was extremely chaotic with various degrees of focus and topic fluctuation for its duration of approximately 1600 postings. Each of these conversations possesses unique properties and we report results for each. As research progresses, various other styles of conversation will be employed for testing.

Once sample conversational data had been obtained it was manually analyzed by three researchers. Each of the two data sets was analyzed and thread starts identified by the three researchers working independently. Upon conclusion, the three results were correlated to form a “ground truth” of thread starts for each of the two sample conversations.

6.2 Evaluation Metrics

Our evaluation methodology is based on the standard metrics of precision and recall.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Or

$$\text{Precision} = \frac{\text{correct thread starts identified by patterns}}{\text{all thread starts identified by patterns}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Or

$$\text{Recall} = \frac{\text{correct thread start identified by patterns}}{\text{all thread starts in ground truth}}$$

Pattern Number	Precision	Recall	TP
1.	28.57	22.53	16
2.	23.52	16.9	12
3.	28.07	22.53	16
4.	26.31	7.042	5
5.	41.66	14.08	10
6.	40	5.63	4
7.	50	5.63	4
8.	28.57	2.81	2
9.	NONE	0	0
10.	0	0	0
11.	0	0	0

12.	NONE	0	0
13.	7.40741	5.63	4
14.	14.28	1.4	1
15.	0	0	0
16.	NONE	0	0
17.	0	0	0
18.	54.54	16.9	12
19.	2.830	4.22	3
20.	8.333	5.63	4
21.	10.52	8.45	6

Figure 2: Precision and Recall of All Patterns

Figure 2 depicts the results of individual patterns and their precision, recall and true positives in identifying thread starts. If a pattern has relatively high precision and recall, it is considered a positive pattern. If a pattern has high recall it is considered as a positive pattern since positive patterns are used to identify thread starts, which are then pruned by negative patterns. If a pattern has high precision, it is considered positive, since it means the pattern is pure, i.e., most of its results are thread starts. Negative patterns are those with low precision and recall since they are used to prune the postings that matched positive patterns. It is interesting to note that although we initially classified pattern 18 as a negative pattern, in fact has the characteristics of a positive pattern. In future experiments we will use pattern 18 as a positive pattern and compare the resulting precision and recall.

The regular expressions used to compute the patterns were also tested to ensure high precision and recall – these results are presented in Appendix A.

6.3 Pattern Combinations

Patterns combined	Precision	Recall
01_06_10_12_13_14_17_19_20_21	47.4286	35.9307
01_03_06_10_12_13_14_17_19_20_21	47.4286	35.9307
01_03_06_07_10_12_13_14_17_19_20_21	47.4286	35.9307
03_07_09_10_12_13_14_17_19_20_21	47.3988	35.4978
01_02_04_05_06_08_10_17_19_20_21	35.0685	55.4113
01_02_03_04_05_06_08_10_17_19_20_21	35.0685	55.4113
01_02_04_05_06_08_14_19_20_21	35.2778	54.9784
01_02_03_04_05_06_08_14_19_20_21	35.2778	54.9784

Figure 3: Highest Precision/Recall Combined Patterns

Figure 3 depicts some of the highly ranked combinational precision and recall results. It is interesting to note that most of the high performance combinations include either pattern 1 or 3, both a means of identifying a question. Both of these patterns have the highest recall in Figure 2.

In presenting our results we chose to average precision and recall values of all combinations produced (cf. section 5.3.5). The averaged results are broken down further by the nature of the combining operator. Since both logical AND and logical OR have been employed, four combinations of operations are possible: the AND of the positive patterns filtered

by the AND of the negative patterns; the AND of the positive patterns filtered by the OR of the negative patterns; the OR of the positive patterns filtered by the AND of the negative patterns; the OR of the positive patterns filtered by the OR of the negative patterns. These four possible experiments are identified as C1, C2, C3 and C4 respectively in what follows.

By employing this approach, we can identify not only the optimal combination of individual patterns but also the optimal operators to use in combining patterns.

Figures 4 and 5 display the average precision and recall results for these four experiments for each data set described in Section 6.1. Results were grouped into intervals of 10%. For each point plotted on the graph, values were calculated by averaging precision and recall for entries within the interval.

Three conclusions can be drawn from Figures 4 and 5. First, C4 appears to have better performance than C1, C2 and C3. Although C1 and C2 initially have higher precision, their rate of precision decay as recall increases is high. After a 15% increase in recall, all four curves reach an approximately equal precision level. In contrast, C4 results in relatively good levels of recall while maintaining precision. This seems to indicate that the logical OR of the positive patterns combined with the logical OR of the negative patterns yields the highest performance overall.

Secondly, the four logical operator combinations are clearly divided into two groups for low values of recall. Both C1 and C2 initially have higher precision than C3 or C4. Since these groups differ in how positive patterns are combined (C1 and C2 use a logical AND while C3 and C4 use a logical OR), it seems that the logical operator used to combine positive patterns affects the results to a greater degree than the logical operator used to combine negative patterns.

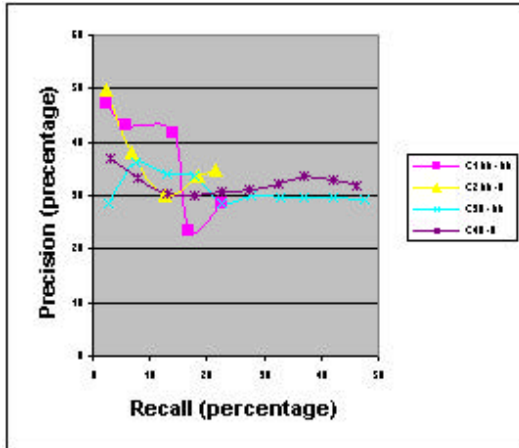


Figure 4: Precision vs. Recall for Data Set 1

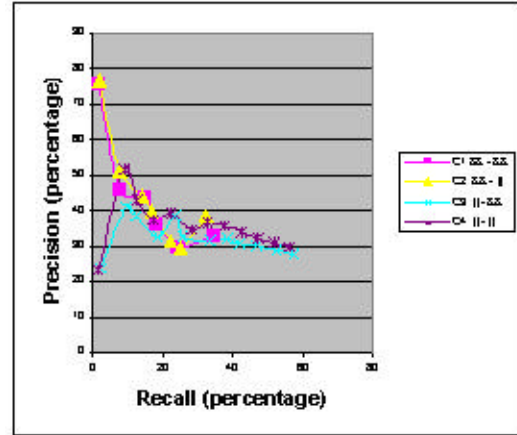


Figure 5: Precision vs. Recall for Data Set 2

6.4 The F Measure Metric

The third observation is the similarity of the C3 and C4 precision/recall curves. C4 has slightly better performance than C3 overall. It is difficult to conclude from these curves, however, whether C4 is clearly better than C3. To resolve this issue we employed the F-beta metric to evaluate performance.

F-beta is a single metric of evaluating precision and recall simultaneously. P is precision, R is recall and β is a factor that determines the relative importance of precision versus recall. A value of $\beta=1.0$ is often chosen for equal weighting of P and R.

$$F = \frac{(b^2 + 1)PR}{b^2P + R}$$

Figure 6 depicts the results of varying β and records the resulting F values. For all values of β , C4 has the greatest value of F (i.e., the best combination of precision and recall). We conclude from this observation that the best logical operator to combine both positive and negative patterns is the OR operator.

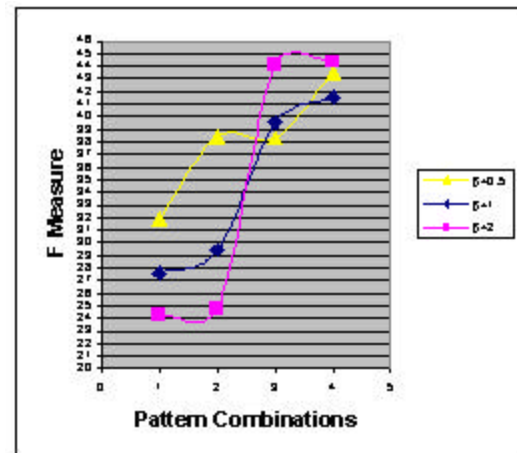


Figure 6: F-beta for Data Set 1

It is also notable that both Figures 4 and 5 display the expected inverse relationship between precision and recall as exhibited by the rough hyperbolic nature of the resulting curves. This confirms the general nature of our results given the common knowledge of this phenomenon in the information retrieval field.

7.0 Future Work

Our research goal of successfully mining social and semantic relationships from the chat medium is a long term objective. The research is currently in its primary stages and there is still much to be accomplished.

The immediate subsequent step is to continue fine-tuning the patterns employed, and the manner they are utilized to improve precision and recall. We continue to seek new patterns for identifying thread starts, but focus is shifting now towards patterns for identifying thread continuity and linking postings together. Once a complete method for connecting thread starts (or thread dominators) with non-thread starts and extracting a thread is developed, we will have accomplished our first objective of obtaining high-quality items from the chat input. These items can then be fed as input to the SSB model builder in order to form social and semantic relational models.

The patterns for linking threads require new techniques in addition to regular expression matching. Chat data is fragmented, informal speech. People often use slang, contractions, abbreviations and other colloquialisms in informal chat that an automated approach must account for before successfully grouping related postings into topical threads.

Another area of improvement is the method of pattern combinations. We are researching a transformation-based error-driven learning method to generate combinations of patterns. This will result in an optimal combination of patterns according to the logical AND, OR, and NOT operators. We hope to increase both precision and recall in this way.

Name identification is another issue that needs to be resolved. Depending on the category of chat data, participants in a forum may address each other by their screen names or real names. If they use screen names, this facilitates social interaction analysis since all screen names for a conversation are available. If they use real names however, a method must be developed to dynamically match real names to screen names. Real names and nicknames must be recognized as such within chat conversation and not dismissed as misspelled words.

There are various other chat issues to consider as well such as the idiosyncrasies of chat conversation, using irregular expressions such as brb, u, r, :), etc. The effect of time lapse may also be useful. The amount of time between postings, or spent typing a posting, could indicate the level of semantic focus during a conversation.

8.0 Conclusion

We have successfully developed an automatic approach to identifying thread starts, the locations within complex multi-person, multi-topic chat conversations at which new conversational threads begin. Results thus far have exceeded our initial expectations and serve as motivation for further research.

Our method involves exploiting manually identified patterns, applied negatively and positively by logical operators, to identify thread starts. The next research step is to develop a methodology for thread linkage and continuity in order to extract complete conversational threads. At that point, social and semantic models of conversation will be created using the HDDI™ Social Semantic Builder.

We are confident that extracting and mapping social and semantic relationships in a collection of data will prove extremely useful in textual data mining applications. Certainly the Intelink intelligence network, one of the motivating sponsors of this work, will find this application of genuine value. The potential applications are many, extending into nearly all sectors both industrial and academic. We will continue working toward our overall goal of developing a tool that will automatically derive social and semantic relationships from data expressible in textual form, whether collections of documents, chat data, or other forms of human written communication.

9.0 Acknowledgements

This research was supported in part by the National Science Foundation, Grant EIA-0070457 in the Division of Experimental & Integrative Activities in conjunction with the Intelink intelligence network. Many thanks to Program Directors Rick Adrion and Larry Brandt for their assistance.

We also wish to acknowledge the faculty, students and staff in the Lehigh University CSE Department for their assistance in completing this work. In addition, we especially thank Dr. Alaina Kanfer and M. Cameron Jones for their early contributions to this effort.

Co-authors William M. Pottenger and Tianhao Wu would like to express their gratitude to their Lord and Savior, Jesus Christ, for His continual guidance and support in their lives.

References

- [1] Bader, R., M., Callahan, D. Grim, J. Krause, N. Miller and W. M. Pottenger. The Role of the HDDI™ Collection Builder in Hierarchical Distributed Dynamic Indexing. *Proceedings of the Textmine '01 Workshop, First SIAM International Conference on Data Mining*. April 2001.
- [2] Bouskila, F.D. and William M. Pottenger. The Role of Semantic Locality in Hierarchical Distributed Dynamic Indexing. *Proceedings of the International*

Conference on Artificial Intelligence (IC-AI'2000), Las Vegas, NV, June.

[3] B. Burkhalter, J. J. Cadiz and M. Smith. Conversation Trees and Threaded Chats. In the *Proceedings of the CSCW'00 Conference*. December 2-6, 2000, 97-105, Philadelphia, PA

[4] Chen, H. and K. J. Lynch. Automatic Construction of Networks of Concepts Characterizing Document Databases. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):885-902, September/October.

[5] Choi, Freddy *Linear text segmentation: approaches, advances and applications*. *Proceedings of CLUK3*, Brighton, England, 2000.

[6] Grefenstette, Gregory. *Explorations in Automatic Thesaurus Discovery*. Boston: Kluwer Academic Publishers, 1994.

[7] Hearst, Marti A. TextTiling: A Quantitative Approach to Discourse Segmentation, Technical Report UCB: S2K-93-24, 1993

[8] Lewis, David. An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task. *Proceedings of the Fifteenth Annual International ACM SIGIR conference on Research and development in information retrieval* Copenhagen, Denmark, 1992.

[9] Sparck-Jones, K. *Automatic Keyword Classification for Information Retrieval*. Butterworths, London, 1971.

[10] Kuntraruk, Jirada and William M. Pottenger. Massively Parallel Distributed Feature Extraction in Textual Data Mining Using HDDITM. In the *Proceedings of The Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*. San Francisco, CA, August 2001.

[11] Ozaku, Hiromi, Kiyotaka Uchimoto, Masaki Murata, Hitoshi Isahara. Topic Search for Intelligent Network News Reader HISHO. *Proceedings of the 2000 ACM symposium on Applied computing 2000*. Como, Italy, 2000.

[12] Pottenger, William M., Miranda R. Callahan, Michael A. Padgett. Distributed Information Management. *Annual Review of Information Science and Technology (ARIST Volume 35)*, 2001.

[13] Pottenger, William M., Yong-Bin Kim and Daryl D. Meling. HDDITM: Hierarchical Distributed Dynamic Indexing. In *Data Mining for Scientific and Engineering Applications*, Robert Grossman, Chandrika Kamath, Vipin Kumar and Raju Namburu, Eds., Kluwer Academic Publishers, July 2001.

[14] Rennison, Earl. Galaxy of News: An Approach to Visualizing and Understanding Expansive News Landscapes. *Proceedings of the ACM symposium on User interface software and technology*. Marina del Rey, California, United States, 1994.

[15] Sack, Warren. Conversation Map: A Content-Based Usenet Newsgroup Browser. *Proceedings of the 2000 international conference on Intelligent user interfaces*. 233 -240. New Orleans, Louisiana, 2000.

[16] Vronay D., Smith M., and Drucker, S. Alternative Interfaces for Chat. *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology (UIST 99)*, 19-26.

[18] www.nist.gov/speech/tests/tdt/

[19] April Kontostathis and William M. Pottenger. Transitivity and the Co-occurrence Relation in LSI. In the *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August. (Submitted for review)

Appendix A

Regular Expression Number	Training dataset		Test dataset	
	Precision	Recall	Precision	Recall
1.	100%	100%	100%	100%
2.	100%	100%	100%	100%
3.	100%	100%	100%	100%
4.	78.94%	71.4%	75%	86%
5.	100%	92.3%	100%	71.4%
6.	100%	100%	100%	100%
7.	100%	100%	100%	100%
8.	100%	100%	100%	100%
9.	NONE	NONE	100%	100%
10.	100%	100%	100%	80%
11.	100%	100%	100%	100%
12.	NONE	NONE	100%	100%
13.	100%	100%	100%	87.5%
14.	100%	87.5%	100%	100%
15.	100%	100%	100%	100%
16.	NONE	NONE	100%	100%
17.	100%	100%	100%	100%
18.	100%	100%	100%	100%
19.	99%	100%	100%	100%
20.	100%	88.9%	100%	100%
21.	95%	100%	100%	100%

Evaluation of Regular Expression Matching Tool