Recovering Implicit Thread Structure in Newsgroup Style Conversations

Yi-Chia Wang, Mahesh Joshi
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{yichiaw,maheshj}@cs.cmu.edu

William Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Carolyn Rosé
Language Technologies Institute/
Human-Computer Interaction Institute
Carnegie Mellon University
Pittsburgh, PA 15213
cprose@cs.cmu.edu

Abstract

On-line discussions are composed of multiple inter-woven threads, regardless of whether that threaded structure is made explicit in the representation and presentation of the conversational data. Recovering the thread structure is valuable since it makes it possible to isolate discussion related to specific subtopics or related to particular conversational goals. In prior work, thread structure has been reconstructed using explicit meta-data features such as "posted by" and "reply to" relationships. The contribution of this paper is a novel approach to recovering thread structure in discussion forums where this explicit meta-data is missing. This approach uses a graph-based representation of a collection of messages where connections between messages are postulated based on inter-message similarity. We evaluate three variations of this simple baseline approach that exploit in different ways the temporal relationships between messages. The results show that the three proposed approaches outperform the simple threshold-cut baseline.

Introduction

As the language technologies community focuses more and more on the analysis of social media rather than primarily focusing on expository text, the characteristics that set conversational data apart must be considered explicitly. Whereas expository text is organized around the information that is presented, typically in a single coherent argument, conversation is organized around a shared task where multiple participants act and react to one another as the scenario unfolds. Because the interaction is not, and indeed cannot, be fully planned out ahead of time, the structure emerges from the complex and intertwined separate intentions of each of the actors as they participate in their joint task. It has been widely noted that the temporal ordering of contributions to an ongoing conversation can be exploited in the analysis of social media (Kleinberg, 2002; Kumar et al., 2003; Dubinko et al., 2006). Nevertheless, with multiple actors, each with their own goals and discourse intentions, it cannot be assumed that the conversation proceeds with a single focus at a time, which would result in a simple, treelike structure (Grosz and Sidner, 1986). Rather, conversational data may have a complex, multi-threaded structure (Rose et al., 1995; Wolf and Gibson, 2005). Thus, limiting the structural representation of conversational data to a singular sequential ordering is overly simplistic and limiting.

Isolating the individual threads that make up a complex, multi-party interaction plays a pivotal role in making sense out of the individual contributions on multiple levels. For example, from a group knowledge building perspective, teasing out the individual threads is valuable for characterizing the consensus building style of the actors (Wang et al., 2007). On the level of conversation acts, identifying which contributions fit together on a single thread is useful for identifying adjacency pairs and thus leveraging the notion of sequential relevance for disambiguating the actions performed (Stolcke et al., 2000). Thread structure can also be used to identify particularly influential contributions by analyzing backward links between messages to identify which contributions were responsible for stimulating the longest discussion threads. Polarity prediction within the area of sentiment analysis may benefit from the identification of thread structure. For example, if Participant C disagrees with B's disagreement with A's opinion, then C agrees with A. This cannot be correctly resolved from the characteristics of individual contributions in the absence of the discourse structure.

Consistent with this view, thread structure has been advantageously applied to many different research problems in the area of social media analysis, such as email summarization (Carenini et al., 2007), text classification task in discussion forums (Wang et al., 2007), and newsgroup search (Xi et al., 2004). In this prior work, meta-data that makes the thread structure explicit played an important role in identifying the parent child relationships between conversational contributions. Unfortunately, there are many popular conversation streams that are not organized explicitly into threads. The lack of explicit thread representation may limit the exploitation of thread structure in the analysis of social media. In contrast to this prior work, we develop an algorithm for recovering thread structure where it is not made explicit in associated meta-data that accompanies conversational contributions.

As an example of such a data set, for our analysis, we use a data set collected within a multi-party educational game environment called LegSim (Wilkerson and Fruland, 2006). The LegSim environment is a simulated legislature where students play the roles of state representatives, who are responsible for proposing and examining bills. In this environment, students engage in debates possessing a complex, threaded structure. However, that thread structure is neither explicit in the stored representation of the data nor from the interface that the students use to engage in the multi-party conversation.

The contribution of this paper is a novel approach to recovering thread structure in discussion forums where explicit meta-data for reconstructing the thread structure is missing. We refer to this task as the *thread structure recovery task*. In particular, thread structure recovery is the process whereby a parent message is explicitly linked to one or more responding child messages. Our approach consists of two steps. First, we use a connectivity matrix constructed using a shallow message similarity measure to represent the implicit relationships between messages within the same conversation stream. Each element in the matrix is a weighted link between a message pair, which measures the content similarity between the messages. The second step is to determine parent-child relationships using the connectivity matrix. A simple baseline approach merely uses a threshold to identify which similarity measures are strong enough to constitute an explicit link. We propose three variations of this baseline approach that leverage in different ways the temporal relationships between messages.

In the remainder of the paper, we will first review some important related work in section. Next, we will describe the formal definition of the task and dataset as well as the details of our algorithms. Then, we will present our evaluation by comparing the effectiveness of the alternative versions of our basic approach. We conclude with some directions for our continued investigation.

Related Work

As discussed in the previous section, some recent work has demonstrated that explicit thread structure can be advantageously utilized in the analysis of social media. First, it can be used for boosting the performance of automatic analysis. For example, Carvalho et al. exploited the sequential ordering among emails within the same conversation thread to improve the performance of email-act classification (Carvalho and Cohen, 2005). Also, Xi et al. (2004) introduced an effective ranking function for newsgroup search. In that work, Xi and colleagues extracted a set of meta features related to the structure of newsgroup discussion threads, such as the number of children of the current message, the position of the current message in the thread tree, and so on. Similarly, Wang and colleagues (2007) developed a novel approach to leveraging context for classifying newsgroup style discussion segments. Their solution was to use the explicit thread structure to identify pairs of potentially related segments of text. They then used shallow semantic similarity metrics to compute a feature that indicated the maximum similarity value between a segment contributed by one participant and other potentially related segments contributed by other participants in the discussion thread. This added feature significantly improved performance at identifying the consensus building style exhibited by the individual discussion contributions over a baseline using only word level features.

Recently, some research related to thread structure has been done in the context of email summarization work, where recovering thread structure is necessary for constructing coherent summaries of email conversations. To that end, Yeh et al. (2006) studied how to use quoted messages, string matching metrics, and email headers to reassemble the treelike thread structure of email conversations. Inspired by Yeh et al.'s work, Carenini et al. (2007) proposed a new email summarization framework, ClueWordSummarizer. They used what they refer to as a fragment quotation graph to organize conversations and subsequently construct lexical chains. They utilized quotation markers (e.g., ">") as well as quotation depth (the number of quotation marks in the beginning of every line) to build the quotation graph.

In addition to facilitating the automatic analysis of social media, identification of thread structure can be used to aid in the work that conversation analysts do with the data by hand. For example Trausan-Matu et al. (2007) present a conversation analysis support tool that provides a visualization of a threaded conversation. Topics are identified by means of expressions such as "Let's talk about" or "What about", and then the WordNet ontology is used for recovering threads by means of automatically constructed lexical chains. The visualization assists analysts in identifying meaningful subsets of messages to examine together as a coherent conversational unit.

Our work is most related to Shen et al.'s work (2006), in which they introduce a partial solution to the thread reconstruction task for synchronous chat data. In their work, a single-pass clustering algorithm was modified to exploit the temporal information in the streams by computing centroids that update dynamically as the analysis proceeds sequentially through the data stream. The purpose of this analysis is to identify subtopic clusters of contributions. Our work is similar in that we also exploit temporal relationships between contributions. However, their approach falls short of a complete solution in that it does not identify the explicit parent-child relationships between contributions, as we do in our work. Also, Shen et al. employ a different data structure and search technique. Specifically, they employ an incremental clustering algorithm in which they dynamically compute centroids for their clusters based on a combination of semantic similarity and temporal proximity. In our work, we introduce an approach to recover thread structure using a graph-based connectivity matrix, which similarly uses both similarity and temporal information. Our evaluation demonstrates the importance of both types

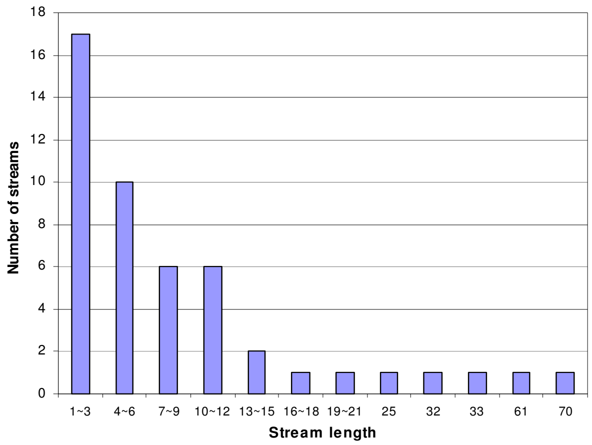


Figure 1. Distribution of Conversation Stream Size.

of information, and contrasts alternative approaches to combining temporal cues with similarity based predictions.

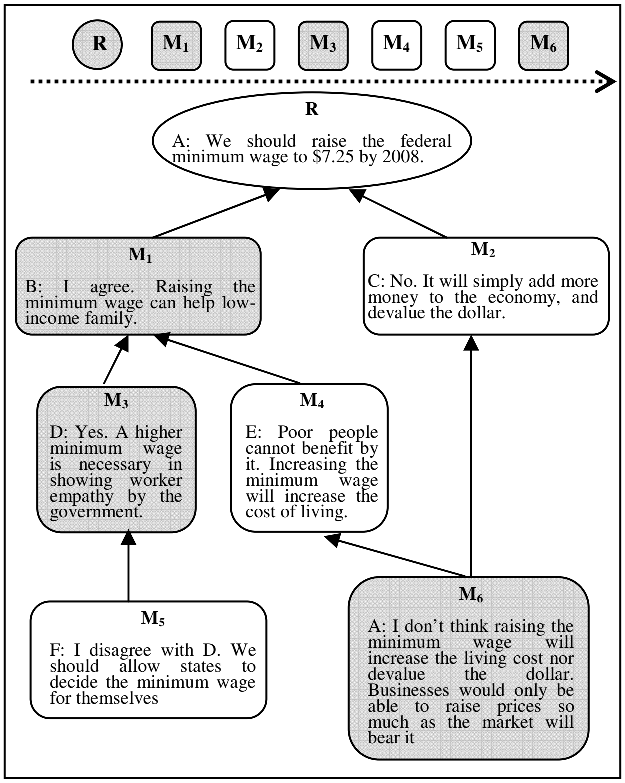
Problem Definition

In this section, we introduce our dataset and offer an example to illustrate the thread reconstruction problem.

LegSim Corpus

The LegSim[1] corpus was provided to us by John D. Wilkerson at University of Washington in Seattle. That dataset contains a collection of messages contributed within an educational multi-player web-based teaching tool called LegSim. LegSim has been in instructional use since 2000, and was used by approximately 1400 high school, community college and university students during the 2005-06 school year. It includes features that allow students to participate in the form of online discussions. It simulates a legislative environment where students in a government or public policy course assume representational roles and advance legislative proposals that reflect their own political priorities.

We chose the data from one college level Civics course from Fall 2006. 93 students participated in the course, 71 of which actively participated in the on-line discussion. They were able to post comments and opinions regarding the proposed legislation (a.k.a. a bill). Over the course of the semester, 478 messages were posted. In this data, the messages were organized into 48 separate streams, as determined by which bill they referred to. The histogram in Figure 1 shows the detailed distribution of number of messages per stream. The longest stream was composed of 70 messages, while the shortest one contains only 1 message. The average number of messages per stream is 9.98.



A: We should raise the federal minimum wage to $7.25 by 2008.

B: I agree. Raising the minimum wage can help low-income family.

C: No. It will simply add more money to the economy, and devalue the dollar.

D: Yes. A higher minimum wage is necessary in showing worker empathy by the government.

E: Poor people cannot benefit by it. Increasing the minimum wage will increase the cost of living.

F: I disagree with D. We should allow states to decide the minimum wage for themselves

A: I don't think raising the minimum wage will increase the living cost nor devalue the dollar. Businesses would only be able to raise prices so much as the market will bear it

Figure 2. A thread example in LegSim and the goal of thread recovery.

A Simple Thread Example

Figure 2 displays a simplified conversation example from the LegSim dataset. The nodes above the dashed arrow denote a discussion stream, arranged from left to right in the same order in which they were posted. Every discussion stream consists of discussion related to a specific topic, in this case a proposed bill that is under debate. For example, the topic of this example is whether the government should raise the minimum wage. Participant A posted the root message R to initiate the discussion. Following that is a sequence of reply messages (M1 to M6), which respond to previous messages. The graph illustrates the thread relationships among the messages in the discussion stream. A directed link in the graph represents a parent-child relationship, with arrows that point from child to parent (e.g., M3→M1). In conversational data, it is possible for a message to have multiple parents (i.e., one message can reply to several previous messages at the same time.) Take the message M6 as an example. It expresses a disagreement with both M2 and M4. Similarly, a parent may have more than one child. For example, both M3 and M4 respond to M1.

Because of the lack of explicit thread structure information in the dataset, before we began any of our experimentation, we manually reconstructed the thread structure by assigning links between messages based on their observed semantic relationship and explicit discourse markers. For example, the following is part of a discussion

[1] http://www.legsim.org/

example from the dataset in which two "representatives" are discussing whether they should pass a bill that proposes "Local Law Enforcement Celebration Day".

> Representative X said "Every day our Police Officers put their life in danger to protect and serve the people. I feel that this bill is the very least that we could do to express our gratitude. It is non-controversial so this should pass quickly."
>
> Representative Y said "I agree... it is the least we can do to show our appreciation... I think it should easily pass also... Thanks Rep Tung for proposing it."

It is clear semantically that the message by Representative Y comments on the message by Representative X. First, explicit discourse markers, namely "I agree" as well as "also", mark that this message is a response to an earlier message. The statement "It should pass" echoes part of the earlier message. Also, its sequential position, adjacent to the earlier message, offers a clue. So, during our manual annotation process, we assigned a parent-child link between these two message based on these types of clues.

We verified that our annotation of parent-child relationships was reliable. Since we expected that the thread structure would tend to be more complicated for longer threads, we evaluated our agreement over three randomly selected threads from those with longer lengths (within the upper most quartile). Altogether these three threads (of lengths 10, 19, and 32 respectively) constituted 15% of the corpus. Two humans annotated these three threads using the definition outlined above. The Cohen's kappa is 0.87 for the three threads. This shows that the annotators achieved a high level of consensus for the link annotation task.

The two human annotators were trained to use the following discourse clues to identify parent-child links when doing annotation. *Response markers*, namely "I agree with X" as well as "In response to X", mark that a message is a response to an earlier message. *Addition markers*, such as "We also have to remember…" and "That's still wrong", mark that the message has some additional comments for the earlier discussion or arguments against a previous post. *Echo markers* echo part of the earlier messages. *Question and answer* in a candidate parent and the current message respectively can link two messages together, too. Finally, *subtopic phrases* have the potential to chain messages together. For example, there are two subtopics (helping low-income families versus devaluing the dollar) in the discussion of the bill "whether the government should raise the minimum wage." The phrases "low-income family" and "poor people" mark that a message should belong to the former subtopic discussion. A phrase such as "devalue the dollar" or "economy", on the other hand, suggests that the message is talking about the later subtopic.

Among the 48 streams mentioned above, 39 streams have two or more messages. After manual reconstruction of the thread structure, 28 of these 39 streams contained at least one parent-child link. We use these 28 streams in the experiments reported below.

Definition of Thread Recovery Task

Given a discussion stream in which messages are sorted by the posting time, the thread recovery task is to construct the thread graph (as in the example in Figure 2). The links in the graph identify which of the previously contributed message the current message replies to. In our problem, three assumptions are made:

Assumption 1: The default parent of each reply message is the initial message of the stream. Thus, that is the assignment made by the algorithm if no parent is identified further down the thread. As we have mentioned above, all messages in the same stream are all about one single topic, and the root message is the one that starts the discussion.

Assumption 2: We assume each message can only link back to those messages that were posted before it in time.

Assumption 3: Except for the posting time of messages, no meta information is available, such as author names, quotation marks, and message titles. We make this assumption since this information is not always available, and the purpose of our experimentation is to determine what can be used to recover thread structure in the absence of this meta-information.

These assumptions take the form of constraints that are enforced in all versions of our algorithm described below.

Alternative Approaches to Thread Recovery

In this section, we first explain how to build a connectivity matrix to reflect semantic similarity between messages. A threshold is then applied to the weighted links in the matrix in order to identify the thread links. To leverage the sequential nature of conversation streams, we describe three variations to the baseline algorithm, which are able to take temporal information into account.

Data Representation

We utilize a vector space model to represent messages. A message is a vector of terms[2], which are weighed by the term frequency (TF) and inverse document frequency (IDF). Here, TF stands for the number of times a term occurs in a message. IDF is the log of the number of messages in the stream divided by the number of messages in the stream that contain the term. TF is normalized, as displayed in the formula below. There are many variations available for computing TF.IDF. Following is the one we used, which is the UMass version of the leading Okapi algorithm (Robertson et al., 1994):

[2] In our experiment, we used non-stemmed unigrams as features.

$$TF.IDF = \frac{tf}{tf + 0.5 + 1.5 \cdot \dfrac{msglength}{avg\_msglength}} \cdot \log\left(\frac{N}{df}\right)$$

where N is the number of messages in the conversation stream.

Intuitively, we selected TF.IDF for our term weights in order to emphasize the most salient terms in the messages in our message similarity measure. This is based on the intuition that what makes a message distinctive within its stream is what is most likely to elicit a response. However, there are instances where this may not be the case. Thus, while using TF.IDF term weights may have some desirable consequences, we cannot rule out at this point the possibility that a different term weight scheme may have been more appropriate. We leave experimentation along these lines for future research

Message-to-Message Graph Building

Given a time-ordered sequence of messages $M=\{\ m_i \mid 1 \leqq i \leqq n\ \}$[3] in a conversation stream, we view each message as a node in a graph. We then build a directed graph by creating edges from m_j to all messages before it in time (i.e. m_i where $i<j$). The associated weight of the edge is computed with the cosine similarity measure. We use a connectivity matrix $\boldsymbol{W}$ to represent the graph, where each element w_{ij} in $\boldsymbol{W}$ denotes the weighed edge from m_i to m_j in the graph. The formal definition of similarity matrix $W = [w_{ij}]_{n \times n}$ is given below:

$$w_{ij} = \begin{cases} \dfrac{\vec{m}_i \bullet \vec{m}_j}{\|\vec{m}_i\| * \|\vec{m}_j\|}, & \text{if } i > j \\ 0 & , \text{ otherwise} \end{cases}$$

where $\vec{m}_i$ is the term vector of message m_i. Note that according to the definition, $\boldsymbol{W}$ is a lower triangular matrix where the main diagonal is zero.

Thread Structure Recovery

Once we have the message-to-message matrix $\boldsymbol{W}$, which represents semantic similarity between messages, we are ready to reconstruct the thread structure for M. First, we describe our baseline algorithm (GR$_B$). In order to decide whether two messages should be linked by a parent-child relationship, a *threshold* is employed as the baseline. Only when the cosine similarity between messages, i.e., w_{ij} in $\boldsymbol{W}$, is larger than *threshold* we will assign a link for them. Considering each entry in $\boldsymbol{W}$, we get the baseline thread graph $G = [g_{ij}]_{n \times n}$.

$$g_{ij} = \begin{cases} 1, \text{ if } w_{ij} > threshold \\ 0, \text{ otherwise} \end{cases}$$

where g_{ij} indicates a link from message m_i to m_j and is set to 1 only if w_{ij} is larger than *threshold*. By adjusting the threshold, we can explore different degrees of similarity between messages.

After informally observing some thread instances in our dataset, we observed that a child message is usually not far away from its parent. As a consequence, we introduced three penalizing functions for $\boldsymbol{W}$, which can differently leverage the temporal ordering and distance between messages. We consider the relative position of messages instead of using real time-stamps.

Penalizing *W* Using Fixed Window Size (GR$_F$). The central idea behind the first variation on the baseline algorithm is that only messages within a certain window size are considered to be the parents of some message. More specifically, a fixed time window is employed to transform the similarity matrix $\boldsymbol{W}$. This yields a new similarity matrix $A = [a_{ij}]_{n \times n}$:

$$a_{ij} = \begin{cases} w_{ij} & , \text{ if } |i - j| < s \\ 0 & , \text{ otherwise} \end{cases}$$

where s is the window size we consider; and $|i - j|$ is the distance between message m_i and m_j. The algorithm for detecting links here (and also in the other two variations) is the same as the baseline except that we substitute the baseline similarity matrix $\boldsymbol{W}$ with $\boldsymbol{A}$. Note that there are two parameters – s and *threshold* – that need to be specified in this penalizing strategy.

Penalizing *W* Using Dynamic Window Size (GRD). Considering that the length of different conversation streams may vary, it might be beneficial if the window size can dynamically reflect the length of the stream. Thus, we use a dynamic window size and use it to compute the new weighted matrix $B = [b_{ij}]_{n \times n}$ as shown below:

$$b_{ij} = \begin{cases} w_{ij} & , \text{ if } |i - j| < p \cdot n \\ 0 & , \text{ otherwise} \end{cases}$$

where n is the number of messages belonging to the conversation stream M; p denotes the percentage of previous messages we want to consider when determining the parents of the current message. This approach also involves two parameters – p and *threshold*.

Penalizing W Using Time Distance (GRT). Finally, instead of using an indicator function to indicate a subset of elements in W, we penalize the similarity by the time distance between the two messages. The result similarity matrix $C = [c_{ij}]_{n \times n}$ is defined as following:

$$c_{ij} = \begin{cases} \left(\dfrac{1}{|\mathrm{i} - \mathrm{j}|}\right) \cdot w_{ij} & , \text{ if } i \neq j \\ 0 & , \text{ otherwise} \end{cases}$$

[3] Note that m_1 is the first message immediately following the start message. The reason that we didn't take the root message into account is because all messages in the same stream are supposed to discuss the same topic initiated by the root. Thus, it is likely that the similarity score between root and any of its child messages will normally be high relative to that of other pairs of child messages, which could interfere with thread recovery.

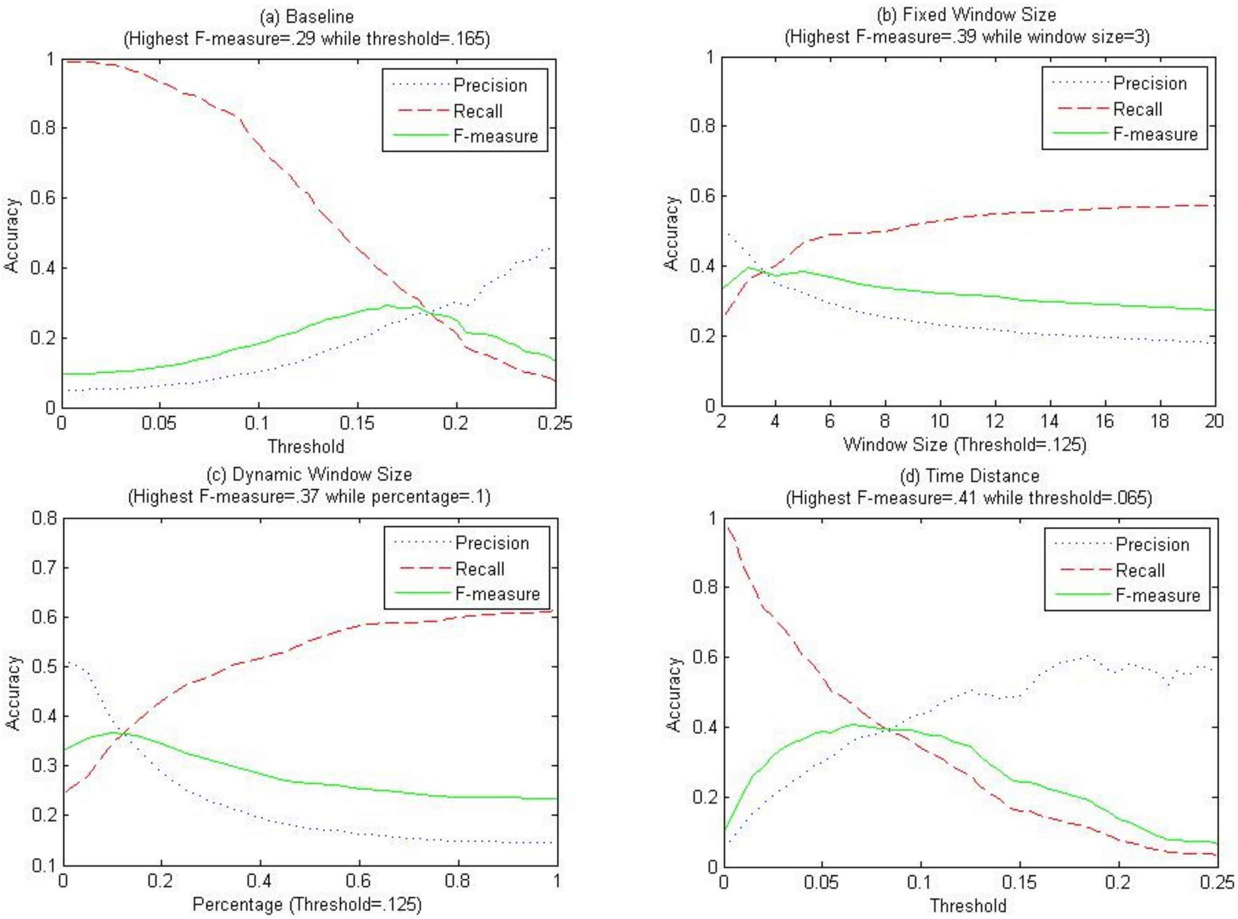


Figure 3. Performance of GR_B, GR_F, GR_D, and GR_T while parameters vary.

where the reciprocal of $|i-j|$ is used to decrease the cosine similarity value in term of the distance between the two messages. For this variation, we only need to consider the similarity *threshold*.

Experiments

We have described the baseline algorithm for recovering thread structure as well as the three variations of it that make use of temporal information. In this section, we compare these algorithms in order to evaluate the contribution of temporal information to the thread structure recovery task. We first introduce our dataset, the evaluation metrics, and then the experimental results. The final part of this section shows the results of parameter tuning.

Evaluation Methods

We use precision, recall, and F-measure to evaluate our results. We explain the definition for each of the three metrics in terms of the thread recovery task as follows:

$$Precision = \frac{|(real\ links) \wedge (predicted\ links)|}{predicted\ links}$$

$$Recall = \frac{|(real\ links) \wedge (predicted\ links)|}{real\ links}$$

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

where we employ the balanced *F-measure* in which precision and recall are evenly weighted.

Results

In our evaluation, we compare the three variant algorithms with the baseline algorithm that only takes similarity into account. The performance curves are shown in Figure 3.

Before we present a statistical comparison of tuned performance across conditions, we will first illustrate how our four approaches vary as we manipulate the parameters that are associated with them. Both Time/Distance and Similarity are used to narrow the range of previous messages that are considered as potential parent messages.

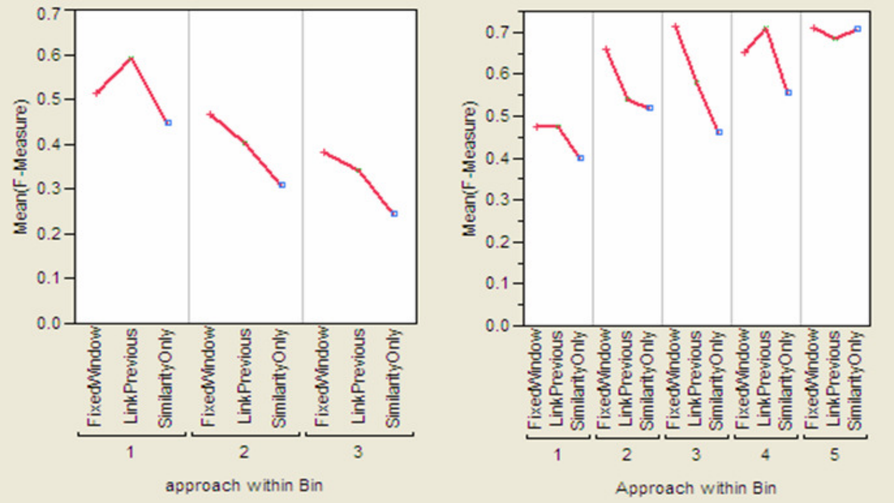


Figure 4. The leftmost graph shows how the three approaches compare with one another as thread length increases. Here, bin 1 contains threads of length 2-5, bin 2 threads of length 6-13, and bin 3 threads of length 14-70. The rightmost graph shows that the three approaches compare with one another as average message length increases. Here, bin 1 contains messages of length 0-40, bin 2 contains messages of length 40-60, bin 3 contains messages of length 60-85, bin 4 contains messages of length 85-120, and bin 5 contains messages of length greater than 120

Thus, as the threshold on Time or Distance is made more restrictive, precision tends to increase, whereas recall tends to decrease. Similarly, as the threshold on Similarity is made more restrictive, we see the same pattern.

Figure 3 (a) displays how the results of baseline GR$_B$ vary as the *threshold* is manipulated. The highest F-measure score is 0.29, which is achieved when the *threshold* is 0.165. Figure 3 (b) and (c) describe the results of GR$_F$ and GR$_D$ respectively. To observe the sole effect of varying the window size, we fix the *threshold* at 0.125. In (b), we find that the F-measure score decreases while increasing the fixed window size *s* from 2 to 20. The best F-measure is 0.39, which is achieved when *s* is set to 3. In Figure 3 (c), the percentage *p* is adjusted between 0 and 1. When *p* is fixed at 0.1 (i.e., so that about 10% of previous messages are considered to be parents of a current message based on the number of messages in each stream) the F-measure reaches its highest value, specifically 0.37. Figure 3 (d) presents the results using time distance to discount the similarity between messages. The highest F-measure score (0.41) is found when *threshold* is equal to 0.65.

Both GR$_F$ and GR$_T$ improve the performance over GR$_B$ by about 30% in terms of F-measure, which argues in favor of taking both similarity and Time/Distance into account when identifying parent-child relationships. For GR$_D$ we have an improvement of 28%. This validates our initial assumption that child messages usually appear close to their parents.

Parameter Tuning

In this subsection, we evaluate the tuned performance of our algorithms. We employ a leave-one-thread-out cross-validation methodology to tune the parameters and evaluate the performance on the held-out test thread on each fold. Note that the only parameter that is tuned for GR$_B$ and GR$_T$ is the similarity threshold (*threshold*). In contrast, there are two parameters that need to be tuned for both GR$_F$ (*threshold, s*) and GR$_D$ (*threshold, p*).

During the training phase, we evaluate all combinations of parameter values in order to identify the parameter settings that optimize the F-measure for each algorithm over the training data, and then apply the tuned model to the test thread. We then compute the weighted average of performance across threads (weighted by thread length so that decisions within long threads are valued equally to those within short threads). From Table 1, we can see the average F-measure for the baseline algorithm as well as for the three variants. We also include one additional baseline approach, which always selects the immediately previous message as the parent message. We refer to this as GR$_{P.}$

Table 1.Evaluation results of leave-one-thread-out cross-validation for each algorithm.

ALGORITHM	GR$_P$	GR$_B$	GR$_F$	GR$_D$	GR$_T$
F-MEASURE	0.36	0.28	0.39	0.34	0.40

Table 1 presents an overview of the results across approaches. We see that the best approach overall is GR$_{F,}$ which uses a fixed window size in addition to bound the search for potential parent messages using similarity above a threshold. Overall, the only statistically significant contrast is between the two endpoint approaches, namely the baseline that considers only similarity information (GR$_B$), and the variant algorithm that combines a fixed window size with similarity information (GR_F). Below we conduct a more fine grained analysis, comparing the performance of the best performing variant approach (GR_F), which we refer to as FixedWindow in the graphs below, with two baselines, namely SimilarityOnly (GR_B) and LinkPrevious (GR_P).

Figure 4 illustrates the comparison of the three approaches for threads of different lengths and parent-child pairs of messages with different average message lengths. In particular, in the leftmost graph, we see performance of the three approaches over threads in 3 sets, with increasing average thread length. We see that as thread length increases, performance decreases overall. And in the most difficult two thread length bins, we see the greatest advantage of the FixedWindow approach over the two baseline approaches. Within the set of threads that combines the threads from the most difficult two bins, the FixedWindow approach is significantly more effective than either of the two baseline approaches. The simple approach of linking to the most recent previous message is surprisingly effective, especially for very short threads. However, similarity information becomes more valuable as thread length increases.

In the rightmost graph in Figure 4, we see another comparison across the same three approaches, this time comparing performance across bins of pairs of potential parent-child messages of average message lengths within particular ranges. Average message length was computed by averaging the parent message length and child message length within the pair. Overall, performance tends to increase as average message length increases. Our intuition is that when average message length is too short, there is not enough information to get a meaningful similarity measure. For the longest messages, we don't see an advantage for the FixedWindow approach over the two baseline approaches. Our intuition is that similarity information is most useful for medium sized messages. As average message length increases above a certain level, more text than is relevant for the parent-child link is considered. We hypothesize that our approach could be improved further by taking discourse focus into account to select the most relevant portion of longer messages to use in the similarity comparisons.

Conclusions and Future Work

This paper introduced an approach to recover thread structure in discussion forums using a graph-based representation of messages. In our model, a connectivity matrix based on inter-message similarity for each conversation stream was computed. We presented three variations of a baseline approach, each of which exploit temporal information to reassemble thread structure on the basis of the similarity graph. The results show that the best variant algorithm can significantly improve performance over two baseline approaches, particularly for longer threads, and for parent-child message pairs of medium length.

As mentioned above, one way in which we plan to investigate ways of improving our performance is by manipulating the vector representation of the messages. In the work reported here, we have utilized term vectors using a TF.IDF weighting scheme. Semantic ontologies such as WordNet might be leveraged to enable generalizing over alternative ways of expressing the same idea as Trausan-Matu et al.'s work (2007). Alternatively, a dimensionality reduction technique such as Principal Components Analysis or Latent Semantic Analysis might have a similar effect.

Beyond investigating alternative weighting schemes, we plan to investigate alternative types of features that might improve the effectiveness of the similarity measures that we compute in identifying candidate parent messages. For example, features that take into account syntactic dependencies between words might better represent the position expressed by the author of a posted message to the extent that these syntactic dependencies enable the representation of argument structure to a limited extent.

In the work reported here, we have used cosine similarity to compute the similarity between messages. However, alternative similarity metrics such as Euclidean distance would have slightly different preferences in terms of relative similarity and might provide an interesting contrast in terms of performance. With a weighting scheme like TF.IDF, Euclidean distance might be more influenced by the relative salience of terms within texts.

In our current work, we are also investigating the thread recovery problem other kinds of social media, such as web blogs. We expect that differences in the nature of the interactions in these alternative settings will lead to interesting contrasts with respect to which representations of messages lead to the best results, as well as which thresholds and window sizes are appropriate.

Acknowledgements

We would like to thank John Wilkerson for sharing the LegSim dataset with us.

This project is supported by ONR Cognitive and Neural Sciences Division, Grant number N000140510043 and NSF Grant number SBE0354420.

References

Carenini, G., Ng, R.T., and Zhou, X. 2007. Summarizing Email Conversations with Clue Words. *16th International World Wide Web Conference* (ACM WWW'07) May 8-12, 2007, Banff, Alberta, Canada.

Dubinko, M., Kumar, R., Magnani, J., Novak, J., Raghavan, P., and Tomkins, A. 2006. Visualizing tags over time. In *Proceedings of the 15th International Conference on World Wide Web*.

Grosz, B. J. and Sidner, C. L.1986. Attention, intentions, and the structure of discourse. *Computational Linguistics* 12: 175–204.

Kleinberg, J. 2002. Bursty and Hierarchical Structure in Streams. *Proc. 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*.

Kumar, R., Novak, J., Raghavan, P., Tomkins, A. 2003. On the bursty evolution of Blogspace. *Proc. International WWW Conference*, 2003.

Robertson, S. E., S. Walker, M. Hancock-Beaulieu & M. Gatford.1994. Okapi in TREC-3. *Text Retrieval Conference TREC-3*, U.S. National Institute of Standards and Technology, Gaithersburg, USA. NIST Special Publication 500-225, pp. 109-126.

Rose, C. P., Di Eugenio, B., Levin, L. S., Van Ess-Dykema, C. 1995. Discourse Processing of Dialogues with Multiple Threads. *Proceedings of the Association for Computational Linguistics*.

Shen, D., Yang, Q., Sun, J.-T., Chen, Z. 2006. Thread detection in dynamic text message streams. In *Proceedings of the Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 35-42). ACM: New York.

Stolcke, A., Ries, K., Coccaro, N., Shriberg, J., Bates, R., Jurafsky, D., Taylor, P., Martin, R., Van Ess-Dykema, C. & Meteer, M. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3), 339–373, 2000.

Trausan-Matu, S., Rebedea, T., Dragan, A., Alexandru C. 2007. Visualisation of Learners' Contributions in Chat Conversations. *Workshop on Blended Learning 2007*, Edinburgh, United Kingdom.

Vitor R. Carvalho and William W. Cohen. 2005. On the Collective Classification of Email "Speech Acts". In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Salvador, Brazil.

Wang, Y. C., Joshi, M., Rosé, C. P. 2007. A Feature Based Approach to Leveraging Context for Classifying Newsgroup Style Discussion Segments. Poster in: *the 43th Conference on Association for Computational Linguistics* (ACL 2007), Prague.

Wilkerson, J., Fruland, R. 2006. Simulating A Federal Legislature. *Academic Exchange: Teaching Political Science*. 10 (4).

Wolf, F., Gibson, E. 2005. Representing Discourse Coherence: A Corpus-Based Study. *Computational Linguistics* June 2005, Vol. 31, No. 2: 249-287.

Xi, W., Lind, J., Brill, E. 2004. Learning Effective Ranking Functions for Newsgroup Search. In *Proceedings of SIGIR 2004*.

Yeh, J.-Y. and Harnly, A. 2006. Email thread reassembly using similarity matching. In *Third Conference on Email and Anti-Spam* (CEAS), July 27-28 2006.