Revista Romana de Interactiune Om-Calculator 9 (3) 2016, 217-232 © MatrixRom

Recovering implicit thread structure in chat conversations

Andrei Dulceanu

Politehnica University of Bucharest, Romania
E-mail: andrei.dulceanu@gmail.com

Abstract. The analysis of chat conversations is a cumbersome task because of the number of different discussion threads that may occur at a certain moment. While most participants in a chat session tend to discuss one topic at a time, interferences appear due to environment asynchrony. This paper presents an approach for recovering implicit thread structure of a chat conversation by using a pipeline centered on semantic similarity between short phrases. Temporal, social and lexical aspects of the conversation are blended in a single model which predicts for each utterance not only the thread it belongs to, but also the utterance most related to in its thread.

Keywords: chat conversation, speech act, thread, disentanglement, semantic similarity, WordNet.

1. Introduction

With the ever increasing popularity of chat systems, blogs (through comments) and microblogging services like Twitter (which recently hit half a billion tweets per day), text streams composed of short messages are becoming more and more common nowadays. Following this kind of conversation is very hard most of the times due to lack of context and absence of strong indicators about whom each participant is answering to. Moreover, when the other participant is identified, it is hard to tell which of its previous utterances generated the response. Although this paper focuses on chat conversations, the insight gained here can help studying other types of interactions, like the afore-mentioned (blog comments, tweets, etc.).

Disentanglement, or thread structure recovery in chat conversations, can be defined as the task of re-arranging all the utterances in several logical threads, according to their contents and to the topic they are debating. Elsner & Charniak (2008) define it as “the clustering task of dividing a transcript into a set of distinct conversations”. The term “thread” was firstly coined in email conversations analysis. The associated task of finding these threads was defined by Yeh & Harnly (2006) as "relating messages by parent-child relationships, grouping messages together based on which messages are replies to other ones".

Carefully reading the two definitions above, we can come up with a subtle difference between them. The first one speaks only about grouping the utterances in several clusters, which ignores the causality between them. An order can be imposed by using the temporal information, if this exists. The second definition puts the relationship between messages at its core, underlining the cause-effect structure of the email corpus studied. The method presented in this paper is more related to the second approach; therefore, the goal is to separate the conversation into different threads, the utterances in each thread being grouped by parent-child relationships.

This is not, for sure, a trivial task. To better understand and visualize it, Figure 1 presents a short excerpt of a chat conversation between Computer Science students.

```
1) 08:12:32 Mihai: I rarely find a solution for a
problem on something else than forums.
2) 08:12:59 Mihai: Wikis are too general.
3) 08:13:06 Cosmin: because it's easy to post a message
- so, easy to generate
4) 08:13:09 Mihai: blogs are rarely to the point
5) 08:13:10 Angela: forums are ok, but wiki is better
6) 08:13:23 Cristi: ☺ first off, the benefit of a
technical blog, as opposed to other solutions is that it
gives people the opportunity to put down their ideas in
a way that doesn't get lost
7) 08:13:25 Angela: don't forget the spam
8) 08:13:30 Mihai: and of course, as I said, good SEO
makes it child's play to find info
9) 08:13:31 Angela: on forums
```

Figure 1. An example of a multi-threaded discussion

In order to better judge the utterances above, it must be noted that Mihai and Cosmin are forum supporters, Angela is a wiki supporter and Cristi supports blogs. The first utterance is the most important statement of the conversation, expressing Mihai's opinion about forums' superiority versus wikis or blogs. This attracts Cosmin's agreement, who adds more to the explanation, but also Cristi's and Angela's disagreement. Angela's last two utterances can be seen as a whole. This is a very common habit of chat users, which put the speed of communication on the first place, generating short messages which do not have any meaning, if read separately. Another interesting aspect here is that overall, the excerpt is very cohesive, fact preserved throughout the whole conversation. Figure 2 illustrates a possible interpretation of the interwoven threads present in the quoted fragment.

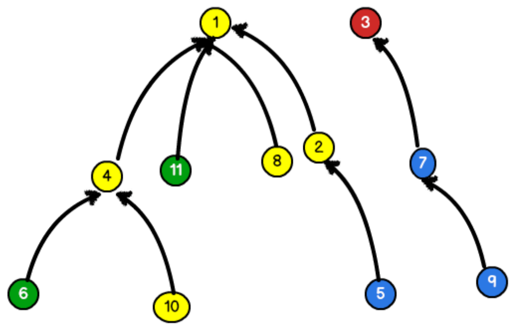

Figure 2. Two possible threads and the edges connecting the utterances; participants are shown in different colors

Practical applications which may benefit from identifying conversational threads include those focused on summarization, conversation monitoring or those which aim to mine for a deeper knowledge from this kind of text stream. For example, looking at Figure 2, one might conclude that utterance 1 is the one which generated most replies in the excerpt, while the speaker colored in yellow (Mihai) was the most influential. Moreover, participants can be ranked by the number of utterances which generated replies, in order to see which users facilitated collaboration. All the statistics presented can be features of an automated conversation evaluator.

Thread detection is accomplished in a two-step process: an iteration through utterances in the conversation which are checked against those in each thread (second iteration). At first, of course, there are no threads and the first utterance will be assigned by default to thread-1. Another iteration through the list of utterances is performed, and for each, an **affinity** measure between the utterance and existing threads is computed. If the affinity is greater than an established threshold, the utterance is assigned to its matching thread, otherwise a new thread will be started.

The interesting part of this method is that affinity is defined as a maximum joint measure between the similarity to existing utterances in the thread (as a whole) and maximum similarity between current utterance and each individual utterance in the thread. In other words, once an utterance is selected to be included in a specific thread, the utterance with the greatest similarity (both semantic and social) to it is also known.

The paper continues with the presentation of relevant work concerning conversational threads in both emails and chat conversations, lexical chains identification and different ways for computing similarity measures. Section 3 provides a detailed overview of the problem, while Section 4 comes with the detailed solution and processing done. The experimental work and its evaluation are presented in Section 5. The paper ends with conclusions and future work.

2. Related work

The work presented in this paper is related to two main research areas: thread detection in email and chat conversations and lexical chains computation.

A. Thread detection in email and chat conversations

The approaches concerning the identification of threads in email and chat conversations can be divided in two main categories: the ones which use clustering and the ones which use probabilistic generative topic models for the task.

Elsner & Charniak (2008) introduce a graph partitioning algorithm model. For this, a maximum-entropy classifier first labels utterances in two categories: alike and different. As labeling all the utterances in the conversation only one time would not give any insight about its structure, the classifier is run multiple times, considering a time window of approximately 129 seconds. Features used for classification include a mixture of chat-specific, discourse and content, considering distance in time between utterances, name mentions, cue words and repetitions, to name - only a few. The second part of the algorithm is using a greedy technique for partitioning the conversation. In order to include an utterance in a specific cluster, the results of the classifier on previous utterances are consulted. The winning cluster is that for which the current utterance displays the greatest resemblance according to the classifier. It is unanimously regarded that the results obtained with this algorithm constitute the baseline for future research.

Wang & Oard (2009) cover threading in chat conversations. The authors try to build “a context-sensitive document for each message” by exploiting “temporal and social aspects of the conversations”. Each message is expanded to contain some of the previous messages based on utterance by the same author, their containing a mention to another author’s name or by time proximity. A single pass clustering is applied on the expanded messages using cosine to measure similarity. Results show that the method proposed outperforms the best previously known technique of Elsner & Charniak (2008).

Shen et al (2006) adapted the topic detection and tracking (TDT) work for the task at hand, but with slight variations, taking into account temporal information, different length of messages vs. stories, and interactivity of short message streams vs. classical, static texts. The hypothesis used is that each thread corresponds to one topic and only one, while a topic may be discussed in several threads. Moreover, author information from each message is not used because it does not add much value to the task, but may generate false leads. Although the algorithm used is similar to the one used by Wang & Oard (2009), some variations make it unique: the use of “discourse structure information”, of messages and of personal pronouns of the subject. Discourse structure information is, in my view, another way of looking at dialogue acts, Stolcke et al (2000), but only a few subsets of them are used: statements, questions, requests and conditionals. It is stated that these two variations, temporal and by using linguistic features outperform baseline single-pass clustering algorithm by 54.6% and 9.7% respectively.

Mayfield et al (2012) propose a model which “annotates a conversation by utterance, groups utterances topics by local structure into sequences, and assigns sequences to threads”. The first annotation of utterances is performed using a supervised probabilistic classifier which computes a probabilistic distribution over four negotiation labels (information giving, information requesting, feedback, others). Afterwards, a two pass clustering algorithm is used for grouping individual sentences into groups and again for uniting the groups into clusters. This is achieved by using a binary probabilistic classifier which uses a time feature (between sentences and groups of sentences) and a coherence metric computed based on cosine similarity between the centroids of clusters. The performance of this approach matched, but did not outperform the results achieved by Elsner & Charniak (2008).

Viewing messages as nodes from a graph-based representation of the conversation is described by Wang et al (2008). Edges between nodes are weighted using cosine similarity measure over TF.IDF weighted term vectors. An adjacency matrix is then used for constructing the threads, in which vertices are connected only if their respective value from similarity matrix computed in step 1 exceeds an empirical chosen threshold. The baseline version of the algorithm is modified by introducing three penalizing functions: the first one considers only messages in a fixed time window, the second one considers a dynamic time window and the latter penalizes similarity by time distance between messages. The methods are evaluated only by comparing the baseline and the penalized versions of the algorithm.

Although it treats the problem of topic segmentation in emails, the work of Joty et al (2010) contains an interesting overview of two state-of-the-art models employed for this task that could be used also for chat disentanglement: LDA described by Blei et al (2003) and LCSeg, by Galley et al (2003). New LDA and LCSeg breeds are proposed, in which the original model is enriched to make use of knowledge from a "fragment quotation graph" (FQG). The FQG is computed by traversing the entire collection of emails and identifying distinct fragment which are considered vertices. The edges in this graph are given by inclusion relations between distinct fragments in the whole email corpus. Evaluation proves that complementing initial models with FQG has given better results than the bare models.

Our work is also related to the work of Trausan-Matu et al (2007), in which it is introduced a tool for visualizing and analyzing multi-party chat conversations. Chat topics are identified by using word repetitions and WordNet ontology, Miller (1995), which helps unifying candidate concepts based on their synonymy. Moreover, cue words or static patterns like "let's talk about email" or "what about wikis" are used to improve topic identification. Threads are reconstructed with a co-reference resolution algorithm which analyzes previous utterances pertaining to a list of statically predefined patterns. In addition, parts of the threads can be reconstructed by using a chat environment facility, which allows explicit referencing of utterances. Knowing that utterances are linked two by two, it can be concluded through transitivity that all three are connected and belong to the same thread.

B. Lexical chains computation

Another thread of related work is lexical chains computation because this task shares the lexical goal of grouping together semantically related words. Going one step further and considering the utterances in which these words appear would provide a basic solution for conversation disentanglement. In our work, we were inspired by Jayarajan et al (2008), which propose an alternative representation for documents, using lexical chains. The interesting part is the preprocessing done for word sense disambiguation in which words are disambiguated by looking at their sentence/paragraph contexts. After POS-tagging all the tokens in a sentence, only nouns are used because they are "better at reflecting the topics contained in a document". Their idea of using only identity and synonymy relations is also used in my algorithm.

In order to improve the finding of topical relations, the method exposed by Moldovan & Novischi (2002) was used. Here the authors propose exploiting glosses of WordNet concepts in building improved lexical chains for question answering. They give two sentences as an example: "Jim was hungry" and "He opened the refrigerator". Although for a human reader the connection between them is already obvious, it is hard to infer it only by using synonymy and identity relations, as stated in the previous paragraph. The missing link is the word "food", which appears in both glosses: "feeling a need or desire to eat food" [hungry] and "a kitchen appliance in which food can be stored at low temperature" [refrigerator].

3. Corpus and annotation

The corpus used in this research comprised of three conversations summing up 846 utterances. Participants used ConcertChat, designed by Holmer et al (2006), a chat client which facilitates collaboration by allowing users to reference partial or whole utterances and by providing a whiteboard widget on which users can draw. CS seniors in a Human-Computer Interaction course were selected to deliberate on which technology among blogs, wikis, chat and forums is better to disseminate information. The only rule of the debate was to cover all the technologies and to come up with an idea of including them all in a product. The last requirement was given in order to study consensus reaching.

Initially stored in *xml* files, the transcripts were converted to *Excel* for a better visualization and for easier annotation. Each utterance has the following attributes: id, author, text and timestamp. Although there was also a reference id attribute, pointing to the utterance which is referenced by current utterance (if exists), this was only used in the final stage for verifying the results. In addition to these attributes, a new one was introduced: speech act. This was added only as a pilot for the current stage of the research and contained only three speech acts: *statement, question* and *answer*.

The annotation was performed manually, by a single annotator, for each of the three conversations used. A comparison of methods for automatically identifying a wider range of speech acts is presented by Dulceanu & Trausan-Matu (2011). In the same paper, the authors experimented a heuristic algorithm for finding implicit links among utterances based on automatically assigned speech acts. The best results were obtained for question-answer links which were identified based on the naïve intuition that an answer comes in response to the nearest question. This is the reason for which only *question*, *answer* and *statement* speech acts were used.

4. Method

4.1 Preprocessing

At first the whole conversation was read from the *Excel* file and the content of each utterance was represented using the Bag of Words model introduced by Salton et al (1975). Before tokenization, contractions like “wasn’t”, “haven’t”, “won’t”, etc. were replaced with their full counterparts like “was not”, “have not”, “will not”. This step was needed for a better extraction of the base form of the verb and for grabbing the negative particle “not” which may help future processing. Tokens of the names of all participants in the chats were kept in a dedicated author list, which was consulted before performing a spell check of each word in the utterance. Some of the words were considered conversation specific and were not spell checked. These included author names (even partial references), proper names and other words which weren't found by the WordNet implementation used (odd enough these included "blog" and "wiki").

The next step was stemming each verb in the utterance. Because Porter stemmer, introduced by Porter (1980), doesn't always produce a valid word, the results were corrected by using Jazzy spell checker (http://jazzy.sourceforge.net/). This returned the single most appropriate suggestion for the word to be corrected. Another unit of work performed in this stage was computing word frequencies for each token.

4.2 Thread recovery method

Once all the pre-processing in the previous step is done the thread recovery task can be started. For this, a first pass through the collection of utterances is done and for each utterance the most appropriate thread for it is returned. This can be an existing thread or a new one based on computed thread affinity over existing threads. In understanding the rest of the algorithm some definitions must be given.

Word to word similarity is a real number in the interval [0, 1] which expresses the semantic similarity between two words. This is computed by taking into account identity, synonyms, occurrences of one word in the gloss of the other as shown by Moldovan & Novischi (2002) and finally on Lin similarity, described in Lin (1998). All these inputs are taken into account because relying only on Lin similarity didn't provide expected results. The result is normalized by dividing it to the inverse of the *tf* (term frequency). The table below synthesizes the computation of this metric:

Table 1. Word to word similarity

Case	Value
The two words are identical	1
The two words are synonyms according to WordNet	0.75
One of the words appears in the gloss of the other	0.5

Inter-sentence similarity (□) is a real number in the interval [0, 1] which defines the similarity between two sentences.

Inter-utterance similarity (□) is a real number in the interval [0, 1] which defines the similarity between two utterances based on inter-sentence similarity, on the match between their authors (author match), on the distance in seconds between them (time affinity) and on their associated speech acts (speech act affinity).

Thread affinity (□) of an utterance to a thread is a real number in the interval [0, 1) which expresses the cohesion between an utterance and a thread in terms of maximum inter-utterance similarity and thread to utterance similarity.

Author match (□) is a real number in the interval [0, 1] which provides a measure for the connection between two utterances from their issuers' perspective. Utterances of the same author or which contain parts of the other author's name are advantaged. Therefore, if both utterances are uttered by the same author the value is 1, if one of them contains parts of the author name of the other is 0.75 and it is 0.25 otherwise.

Time affinity (□) is a real number in the interval [0, 1] for measuring time proximity between utterances. If the messages are following one another in less than 30 seconds the value is considered 1, otherwise it proportionally decreases with time passing.

Speech act affinity (□) is a real number in the interval [0, 1] and expresses the probability of an utterance labeled with a certain speech act to come right after an utterance labeled with another speech act. Since this was the last definition, we will continue with formulas for all these measures.

Table 2. Speech act affinity values

Speech Act	Speech Act B	Affinity
Question	Answer	0.65
Question	Statement	0.25
Question	Question	0.10
Statement	Statement	0.60
Statement	Question	0.20
Statement	Answer	0.10
Answer	Answer	0.35
Answer	Statement	0.45
Answer	Question	0.20

Probabilities are computed as if utterance with Speech Act A comes right after utterance with Speech

Act B

Equations for the rest of the metrics are given in a bottom up value, from simple to complex:

$$\tau = T * \frac{1}{\sqrt{\frac{time_{utt2} - time_{utt_1}}{time\_threshold}}} \quad .1$$

Equation (1) defines **time affinity** as the inverse square root of the time difference between the two utterances (in milliseconds) divided by a time threshold constant.

Obviously the hardest part was to define inter-sentence similarity. For that the measure proposed by Mihalcea et al (2006) was studied, which attempts at first to compute one word vs. the others similarities for each word in the first sentence against words in the second sentence, provided they are tagged as having the same part of speech. The scores obtained are weighted with *idf* scores for each word. The overall score is the arithmetic mean of the sum of similarities obtained for each sentence. This approach was not used in our research as *idf* was hard to define in the context of only one conversation (document). Thus, an adaptation of the metric described by Malik et al (2007) was used in which the normalization of the one word vs. the others similarities was performed by dividing this sum to the sum of sentences' lengths.

Finally, the inter-utterance similarity, incorporating also author match, time affinity and speech act affinity looks like this:

$$\zeta(s_1, s_2) = \frac{1}{2} \frac{\sum_{w \in s_1} \max Sim(w, s_2) + \sum_{w \in s_2} \max Sim(w, s_1)}{len_{common\_words}(s_1, s_2)} . \quad (2)$$

$$\begin{aligned} \xi(s_1, s_2) &= a * \zeta(s_1, s_2) + b * \alpha(s_1, s_2) + c * \tau(s_1, s_2) \\ &+ d * \sigma(s_1, s_2), \\ a + b + c + d &= 1 \end{aligned} \quad (3)$$

Thread affinity of an utterance s_1 to a thread T was defined as:

$$\theta(s_1,T)=\Delta*\max_{s_2\in T}\xi(s_1,s_2)+(1-\Delta)*\zeta(s_1,t), \quad 0<\Delta<1 \qquad (4)$$

where *t* represents a super-sentence made by concatenating the contents of each sentence in thread *T*. Equation (4) says that affinity is computed by adding the maximum similarity between the new utterance and another utterance already in the thread and the similarity to the thread (represented as a super-sentence which contains all words from the utterances in the thread).

Another thing that should be noted in (4) is the consequence derived from the first part of the equation: each utterance is put in a thread by firstly finding its most related counterpart already existing in the thread. This is how parent-child relationships between utterances are discovered. A threshold was employed for thread affinity score. If the score was greater than the threshold, the utterance was added to the thread or else a new thread was started.

5. Experiments

As described in Section 3, the corpus used for the experiments was made of three conversations. The goal was to reconstruct the thread structure for them and also to check the correctness of the algorithm used by comparing the links automatically found to the links manually referenced by the participants.

Although intuition would say that in order to construct the thread structure it is more important to follow the logical links between utterances, in all the forms they appear (semantic, time, speech acts, etc.), experiments have shown that slightly decreasing the value of Δ produces better results. Initially the value was set to 0.65, favoring almost double maximum inter-utterance similarity, but then it was observed that this way the number of threads discovered was very high. Thus the value was decreased and in order to come to more cohesive threads, the thread affinity threshold was also decreased.

The table below summarizes all the values of the parameters which produced the best results during experiments:

Table 3. Parameters used in experiments

Parameter	Source	Value
T	Time affinity	1.73
time_threshold	Time affinity	30.000
a	Inter-utterance similarity – Word to word similarity	0.35
b	Inter-utterance similarity – Author weight	0.20
c	Inter-utterance similarity – Time affinity weight	0.20
d	Inter-utterance similarity – Speech Act weight	0.25
Δ	Thread affinity	0.45
affinity_threshold	Thread affinity	0.50

A first look over the results has shown that there are too many threads that contain a single utterance. Analyzing that utterance, we observed that almost half of the cases it is a joke between participants or other kind of mockery for which a human would easily find its place in the conversation. From a lexical or semantic point of view it was hard to be put in any pattern or formula so that its presence as a single utterance thread makes sense. Fig. 3 presents a chart showing the number of threads found per conversation.

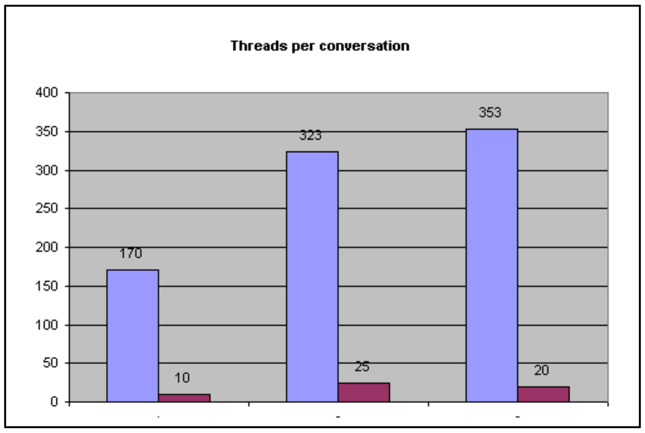

Figure 3. Number of threads per conversation

The length of the threads found is shown separately in fig. 4.

The total number of utterances in each conversation is shown in blue. One-utterance threads are shown in yellow and threads containing two or more utterances are shown in magenta.

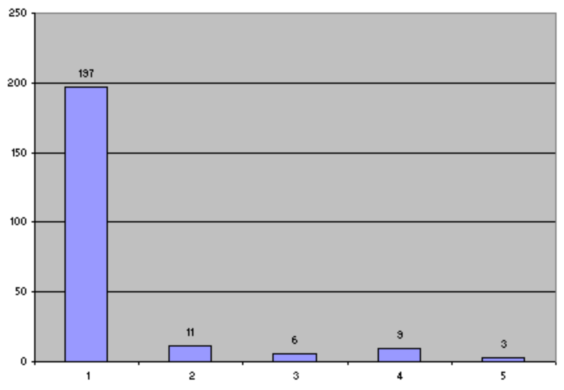

Figure 4. Length of thread categories

The explanation for each bar is:

- Bar 1: number of threads containing 1-5 utterances
- Bar 2: number of threads containing 6-10 utterances
- Bar 3: number of threads containing 11-20 utterances
- Bar 4: number of threads containing 21-40 utterances
- Bar 5: number of threads containing 41 utterances or more

6. Conclusion and future work

This paper introduced a method for recovering thread structure of chat conversation using a new method which focuses on semantic similarity, as well as on other inputs taken from the context of the conversation like speech acts, temporal distance or name mentions. The new measure for establishing to which thread an utterance should be assigned was called **affinity**. At the same time with computing affinity for including a new utterance in a thread (existing or new), the most related utterance from that thread is fetched. This is a novel approach, as all previous work focused only on reconstructing threads as a bulk, considering the utterances already ordered by their issuing time, and not generated as replies to previous messages.

Empirical evaluation has shown that the proposed model might be worth considering, but a solution needs to be found for short threads (1-5 utterances). In order to better understand the performance of the algorithm, a proper evaluation is needed. This would imply having two or more human judges manually re-create the thread structure for the studied conversations and then compute some sort of overlap between this gold standard and the threads obtained by using our method.

Regarding short threads, a new stage may be introduced in the algorithm for unifying these threads with other threads based on a different measure. Speaking of measures, the most important of all, inter-sentence similarity can be improved if word to word similarity could be computed over words in different part of speech categories.

A better visualization of the conversation can be achieved by drawing the tree-like shape resulting from following each link between utterances in the same thread.

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation.*Journal of machine Learning research*, *3*(Jan), 993-1022.

Dulceanu, A., & Trausan-Matu, Ş. (2011). Automatic links identification in chat conversations. *Academy of Romanian Scientists*, 65.

Elsner, M., & Charniak, E. (2008, June). You Talking to Me? A Corpus and Algorithm for Conversation Disentanglement. *In ACL* (pp. 834-842).

Galley, M., McKeown, K., Fosler-Lussier, E., & Jing, H. (2003, July). Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*(pp. 562-569). Association for Computational Linguistics.

Holmer, T., Kienle, A., & Wessner, M. (2006, October). Explicit referencing in learning chats: Needs and acceptance. In *European Conference on Technology Enhanced Learning* (pp. 170-184). Springer Berlin Heidelberg.

Jayarajan, D., Deodhare, D., & Ravindran, B. (2008). Lexical chains as document features.

Joty, S., Carenini, G., Murray, G., & Ng, R. T. (2010, October). Exploiting conversation structure in unsupervised topic segmentation for emails. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (pp. 388-398).

Association for Computational Linguistics.

Lin, D. (1998, July). An information-theoretic definition of similarity. In *ICML*(Vol. 98, pp. 296-304).

Mayfield, E., Adamson, D., & Rosé, C. P. (2012, July). Hierarchical conversation structure prediction in multi-party chat. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 60-69). Association for Computational Linguistics.

Malik, R., Subramaniam, L. V., & Kaushik, S. (2007, January). Automatically Selecting Answer Templates to Respond to Customer Emails. In *IJCAI* (Vol. 7, pp. 1659-1664).

Mihalcea, R., Corley, C., & Strapparava, C. (2006, July). Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI* (Vol. 6, pp. 775-780).

Miller, G. A. (1995). WordNet: a lexical database for English.*Communications of the ACM*, *38*(11), 39-41.

Moldovan, D., & Novischi, A. (2002, August). Lexical chains for question answering. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130-137.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613-620.

Shen, D., Yang, Q., Sun, J. T., & Chen, Z. (2006, August). Thread detection in dynamic text message streams. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 35-42). ACM.

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

Trausan-Matu, S., Rebedea, T., Dragan, A., & Alexandru, C. (2007). Visualisation of learners' contributions in chat conversations. *Blended learning*, 215-224.

Trausan-Matu, S. (2010, August). Automatic support for the analysis of online collaborative learning chat conversations. In *International Conference on Hybrid Learning* (pp. 383-394). Springer Berlin Heidelberg.

Wang, L., & Oard, D. W. (2009, May). Context-based message expansion for disentanglement of interleaved text conversations. *In Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics* (pp. 200-208). Association for Computational Linguistics.

Wang, Y. C., Joshi, M., Cohen, W. W., & Rosé, C. P. (2008, March). Recovering Implicit Thread Structure in Newsgroup Style Conversations. In *ICWSM*.

Yeh, J. Y., & Harnly, A. (2006, July). Email Thread Reassembly Using Similarity Matching. *In CEAS*.