Thread Detection in Dynamic Text Message Streams

Dou Shen[†], Qiang Yang[†], Jian-Tao Sun[‡], Zheng Chen[‡]

[†]Department of Computer Science and Engineering,
Hong Kong University of Science and Technology
{dshen,qyang}@cs.ust.hk

[‡]Microsoft Research Asia, Beijing, P.R.China
{jtsun, zhengc}@microsoft.com

ABSTRACT

Text message stream is a newly emerging type of Web data which is produced in enormous quantities with the popularity of Instant Messaging and Internet Relay Chat. It is beneficial for detecting the threads contained in the text stream for various applications, including information retrieval, expert recognition and even crime prevention. Despite its importance, not much research has been conducted so far on this problem due to the characteristics of the data in which the messages are usually very short and incomplete. In this paper, we present a stringent definition of the thread detection task and our preliminary solution to it. We propose three variations of a single-pass clustering algorithm for exploiting the temporal information in the streams. An algorithm based on linguistic features is also put forward to exploit the discourse structure information. We conducted several experiments to compare our approaches with some existing algorithms on a real dataset. The results show that all three variations of the single-pass algorithm outperform the basic single-pass algorithm. Our proposed algorithm based on linguistic features improves the performance relatively by 69.5% and 9.7% when compared with the basic single-pass algorithm and the best variation algorithm in terms of F1 respectively.

Categories and Subject Descriptors

H.4.m [**Information Systems Application**]: Miscellaneous; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

General Terms

Algorithms, Experimentation

Keywords

Text Stream, Thread Detection, Single-Pass Clustering, Linguistic Features

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-369-7/06/0008 ...$5.00.

1. INTRODUCTION

Dynamic text message streams are rapidly growing on the Internet. These data are produced in an enormous quantity with the popularity of Instant Messaging (IM), Internet Relay Chat (IRC) and online chat rooms. As reported in [6], IM is widely used among enterprises and other organizations as well as in personal communications. The number of users worldwide now exceeds 200 million on the major free services. An example of the dynamic text stream is found in an online chatting group in an intranet forum between professors and students. In this forum, professors and students discuss the problems which are not considered in class and students can share with others what they have learned. A part of this kind of message stream is given in Figure 1.

…...
A: How do we compile the C files?
B: Hi, what checksum did you get for the sample input?
C: using g++ also works!
D: checksum is 230? i encrypted all chacters including whitespaces.
A: But a required .h file is said to be missing...Not on Unix NOR VC++
C: Okay... Let me check it with Gary and S.mon. But, could you tell me which files are missing????
…...

Figure 1: An example of the text message stream.

It is true that most of the message streams produced through IM or IRC are casual chitchat which are neither meant to be searched nor analyzed. However, there is a remarkable category of streams containing valuable knowledge. The above educational example is a good example. If we can find out what problems are most bewildering for students from the online discussions between professors and students, the professors can schedule their class material and time more appropriately. For another example, many open-source projects have channels for developer and user support. Having this valuable information archived and analyzed with good search facilities can make them work more effectively. However, as shown in Figure 1, the messages from different participants on different topics are heavily interwoven. It is not appropriate to treat a message stream as a whole. Moreover, most messages in the stream are too short to be meaningful by themselves, as a single message can not convey a relatively rounded topic without consid-

ering the context information. Therefore, it is necessary to detect the threads in a stream such that each thread is about a specific topic.

Besides the above motivation, some other scenarios also push us to detect threads from message streams. While most participants in chat rooms are engaged in legitimate information exchange, chat rooms can also be used as a forum for discussions of dangerous activities, such as the abduction of children. Therefore, we need to monitor online conversations to aid crime detection or even crime prevention. However, we can not rely on people to monitor the conversations actively for a prolonged time. Since it is prohibitively expensive and usually hard for a person to keep track of all the topics in the text streams where the topics are diverse and the participants change frequently. In addition, a person could easily lose track of the multiple threads in the conversations when the size of the data is large. What is more, users are reluctant to chat when the conversation is being monitored due to privacy concerns.

To detect the threads, we need to consider the characteristics of the message stream carefully which are double edged swords. On the one hand, the structure of message streams is totally different from written discourse because message streams do not contain linear discussions of any single topic, but rather contains partially threaded and interwoven topics that oscillate in short, incomplete messages. Therefore, we can not rely on the tools for traditional text mining tasks, such as the Topic Detection and Tracking (TDT) [1]. On the other hand, a dynamic text stream provides some linguistic features which are not available in other kinds of text. For example, the pairs such as "Could you help me on ...", "Yes, I could " are valuable hints for organizing the messages into threads and finding out the topic of each thread. How to make maximal use of such knowledge for thread detection is an important issue.

In the past, little work has been proposed to address the thread detection problem. An exception is [9], where the authors identified thread starts based on some patterns provided by experts. In this paper, we choose the single-pass clustering algorithm which is popular in TDT as the baseline and then propose three variations of the single-pass clustering algorithm to take the temporal information into consideration. An algorithm based on the linguistic features for text message analysis is also put forward. We compare our proposed methods with the baseline on a dataset collected from online discussions among professors, teaching assistants and students. The experimental results show that by taking temporal information into consideration, the three variants of the single-pass clustering method can improve the performance significantly; by leveraging the linguistic features, further improvement is achieved.

In Section 2, we discuss the related work followed by the stringent problem definition in Section 3. The algorithms for detecting threads are shown in Section 4. Section 5 presents the experimental results on a real dataset together with the discussions of the results. We give the conclusion and future work in Section 6.

2. RELATED WORK

Our work is closely related to topic detection and tracking which is a longstanding problem [1, 20, 11]. TDT research develops algorithms for discovering and threading together topically related material in streams of data such as newswire and broadcast news in both English and Mandarin Chinese. However, our work is different from them in several aspects: (1) the basic element in TDT is a story about a certain topic in news streams while in our work it is a relatively short message which consists of one or several sentences conveying certain information. In our problem, it is hard to determine the topic from one single message. However TDT assumes that the content of each story is rich enough to reflect a specific topic. (2) in our work, there may be more than one thread at the same time, and the text of different threads intersects with each other. (3) the temporal information in our work plays an important role in detecting the thread. For example, the two messages "Could you help me on ...", "Yes. I could" would likely come together in a thread, with the former appearing ahead of the latter and not being far from each other. (4) the text stream produced by IM or IRC is highly dynamic and interactive. Two messages in a thread can be connected with each other not only through the semantic information they convey but also by some linguistic features. Our work is also related to text segmentation [7]. Text segmentation aims to identify the boundaries of topic changes in long text documents and/or document streams, while in our problem, we aim to find the threads from streams consisting of short messages.

Some other papers work on the same type of data [2, 4, 16, 9]. Most of them focus on detecting the latent topics contained in the data. In [2], the authors developed a text classification system named ChatTrack. ChatTrack creates a concept-based profile that represents a summary of the topics discussed in a chat room or by an individual participant. It archives the contents of chat room discussions in XML format. Subsets of this data can be classified upon request to create profiles of particular sessions or particular users. The classifier is pre-trained on more than 1,500 concepts, and it can also be extended to include concepts of interest to a particular analyst. However, the topics in chat rooms are very divergent and it is impossible to provide a universal classifier for all possible topics. In [4], Elnahrawy evaluated the text classification techniques (Naïve Bayes, k-nearest neighbor, and support vector machine) to find suitable methods of chat log classification for automated crime detection.

In [3], the authors adopted complexity pursuit for the topic identification problem in chat lines and news sources. This algorithm separates interesting components from a time series of data and identifies some underlying hidden topics in streaming data. In [8], a topographical visualization tool was put forward for a dynamically evolving text stream, based on the hidden Markov model (HMM). The system is based on the exploitation of the a priori domain knowledge, that is, there are relatively homogeneous temporal segments in a data stream. This system can be seen as a complementary tool for topic detection and tracking applications. However, the state number of HMM is required beforehand. Furthermore, it is hard to extend the tool to work in an online fashion without learning the parameters of HMM from training data.

Another related work is dialog summarization [18], which performs automatic transcription, tracking, retrieval, and summarization of meetings. The difference is that they do not assume there are many threads at the same time.

In [9], Khan et al. realizing the importance of identifying the threads, made a preliminary attempt. However, their

approach relies on some positive and negative patterns provided by experts, which limits its application.

We also want to mention another group of work on data which are similar to the text message streams we exploit [10, 19]. The data they studied are produced by Newsgroups, BBS (Bulletin Board System) or other forums. Although their data appear as a stream consisting of messages, they are organized explicitly into thread trees. The root of the thread tree is the first message posted by someone seeking an answer to his/her question or a message posted by someone who wants to initialize a discussion regarding a specific topic. The thread tree then expands as other people reply to this message and continue the discussion. Xi et al. studied the ranking functions for Newsgroup search in [19]. Their approach is based on some metadata, such as prior knowledge about the author of the message or the depth of the message. In [10], Kim et al. worked on the topic segmentation of hierarchical messages in Web discussion boards. Each thread tree is segmented into coherent units to facilitate indexing and retrieval. However, the structure of thread trees is not available in our problem. It is our objective to discover the structures of the message stream.

In [5], Hatzivassiloglou et al. investigated two linguistically motivated text features (noun phrase heads and proper names) in the context of document clustering. Their experimental results showed that linguistic features could improve the clustering performance. Some other early work also verified the effectiveness of linguistic features in the context of information retrieval [13, 14]. In this paper, we exploit two kinds of linguistic features different from the above mentioned ones. These features are more suitable for analyzing the discourse structure information in our problem.

3. PROBLEM DEFINITION

3.1 Basic Terms

For convenience of discussion, we first clarify the definitions of two concepts in our problem.

Message : A message is an utterance which consists of one or more sentences. It is what each participant expresses at one time. The nodes on the top of Figure 2 denote the messages. The messages can be classified into there categories: a start message which raises a new series of messages focusing on a special topic (T_{1S} and T_{2S} in Figure 2); a reply message which is a response to a previous message (T_{12}, T_{22} and so on in Figure 2); an end message which ends a topic (T_{1E} in Figure 2). An end message is a special case of reply message. Since the end message has a distinct role, we distinguish it from other types of messages. If we can detect the end messages accurately, we can improve the thread detection task since we would not assign messages onto the threads which have been ended. Besides that, end messages can provide some potentially semantic information. For example, if a student posts a question, and after several go-rounds of discussions, the student may end this topic with a message like "It is ok. Thanks" (positive) or "It still does not work. Thank you, anyway" (negative). It is clear that the end messages can indicate the quality of the discussion. If another student raises the same question, we can provide the discussion threads with positive end messages to him instead of those with negative end messages.

Thread : a thread is a series of messages which starts with a start message and then a sequence of reply messages. In our work, we assume that one thread corresponds to one topic. That is, all the messages in a thread are focused on the same topic. For example, T_{1S}, T_{12}, T_{13}, T_{14}, T_{1E} in Figure 2 form a thread. As shown in Figure 2, the messages in a thread form a tree in which the start message is the root of the tree and each reply message is the child of one of the previous messages but not necessarily the nearest neighbor. The thread defined in this paper is a little different from that in the usual sense. Usually, a thread refers to the conversation among several participants from the beginning to the end. It may contain more than one topic. However, for the convenience of the application scenarios presented in Section 1, that is information retrieval and topic monitoring, it is better to detect thread at the topic level. Therefore, we constrain the thread to be a smaller unit compared with the usual concept of thread.

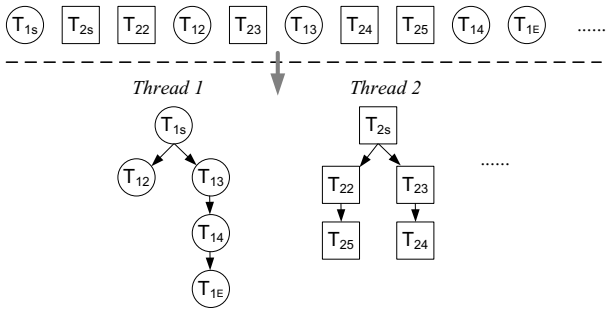

Figure 2: An illustration of the goal of thread detection in dynamic text stream.

3.2 Definition of Thread Detection Task

As we have discussed, the basic and important step for processing the text stream data is to find out how many topics are being discussed and what they are. The detection task can be classified into two categories: off-line detection and on-line detection. The former focuses on the detection of topics in an accumulated collection, and the latter strives to identify the threads from online conversations. In this paper, we work on the off-line detection which is easily evaluated. In fact, most of the algorithms can be applied to on-line detection. In the following, we define the problem more precisely.

Thread Detection : We assume that each thread contains a single topic and each topic may be discussed in several threads. The reason for us to consider thread detection instead of topic detection is that it is easy for us to conduct further research based on threads, for example, to detect the relationships between participants, to study the evolvement of a topic and so on. As we have shown, a thread is a series of messages which starts with a starting message followed by reply messages. The thread detection task is to find out all

the threads in the given text stream without knowing the number of threads beforehand. An illustration is given in Figure 2. It is perfect to detect the tree of a thread accurately as shown in Figure 2. However, it is more difficult than the task to just find out the messages belonging to the same thread. This paper focuses on the latter task although all the proposed algorithms can solve the former one by minor modification. Then, the thread detection problem can be converted to a special kind of clustering problem and can be evaluated with the standard criteria.

We make some assumptions for the thread detection task in this paper. Firstly, we assume that the author of each message is unknown. In fact, in most text messages streams, such information is available. We omit it for the following reasons: (1) some users may change their names during conversation; (2) a user can participate in several different threads at the same. In both cases, information about the authors may give us false leads for thread detection. Therefore, in our work, we only focus on the textual and temporal information in the text streams. However, it may be interesting to detect the true identity of the authors with multiple aliases in Internet chat rooms. We leave it to our future work. Another assumption is that we have no clue about the number of threads in a text message stream. This assumption is reasonable since the text message streams are of variable length and there is no prior knowledge about them.

4. APPROACHES FOR THREAD DETECTION

Thread detection is in fact the task of grouping the messages in the text stream into different groups and each group represents a topic. We employ the single-pass clustering algorithm (incremental clustering) which is tested in TDT [20] as the baseline algorithm. To cater to the characteristics of text stream, we propose three variations of the single-pass clustering algorithm. We also put forward a new algorithm which is similar to the single-pass algorithm where we also employ linguistic features in it. In this paper, we do not adopt the well studied K-means algorithm, since the numbers of threads vary radically among different streams and are hard to be estimated.

4.1 Representation of Messages and Threads

To represent the *semantic information* of messages and threads (clusters of messages) in our detection algorithms, we employ the conventional vector-space model which uses the bag-of-terms representation [12]. A message is represented using a vector of terms (words or phrases), which are weighted using the term frequency (TF) and the Inverse Document Frequency (IDF) in this paper and are appropriately normalized. The normalized sum of all vectors corresponding to the messages in a thread is used to represent the thread. This representation is called a prototype or a centroid of the thread. We use the standard cosine similarity, i.e., the cosine value between vectors of objects (messages or threads) to measure their similarity.

We also introduce two kinds of linguistic features to reflect the *discourse structure information* of messages. One is the *Sentence Type* used in messages and another is the *Personal Pronouns*. English has four main sentence types: *Declarative Sentences* (to state an idea), *Interrogative Sentences* (to form questions), *Imperative Sentences* (to request or demand), and *Conditional Sentences* (to describe the consequences of a specific action, or the dependency between events or conditions). As to the personal pronouns, grammarians have divided them into three categories: the *first person* (such as I, me, my, we, our, and so on); the *second person* (such as you and your) and the *third person* (such as he, she, they, their, his, hers, him, her, and so on). In this paper, we only consider the category of personal pronouns of the subject. If the subject is noun, it is classified as third person. Note that it is nontrivial to identify the sentence type and the subject for a sentence. For simplicity, we employ some heuristically designed automatons. For example, an interrogative sentence usually starts with an interrogative word and ends by a question mark; the subject appears at the beginning in a declarative sentence. The intuition to introduce linguistic features is that the messages in a thread are the conversational utterances between participants and they are highly interactive. The sentence types and personal pronouns used in neighboring messages are not random, but follow some hidden rules. In this paper, we try to discover these rules and apply them for thread detection. For each message M, we will record the Sentence Types of the first and the last sentences in this message using M.STF, M.STL, together with the category of personal pronouns in them using the notations of M.PPF and M.PPL. STF and PPF refer to Sentence Type of the First sentence and Person Pronouns in the First sentence respectively. STL and PPL can be explained similarly. If there is only one sentence in M, then M.STF equals to M.STL and M.PPF equals to M.PPL, which represent the Sentence Type and the category of person pronouns in the single sentence respectively.

4.2 Thread Detection Approaches

In this section, we present the algorithms for the thread detection task. We first introduce our baseline algorithm: a single-pass clustering algorithm. Then we present three variations of the single-pass clustering algorithm as well as a novel algorithm which utilizes the linguistic features.

4.2.1 Single-Pass Clustering Algorithm (SP) and its Variants

Given a text stream in which the messages are sorted according to the occurring time, the basic idea of a single-pass clustering algorithm is as follows. First, take the first message from the stream and use it to form a thread. Next, for each of the remaining message, say M, compute its similarity with each existing thread. Let T be the thread that has the maximum similarity with M. If $sim(M,T)$ is greater than a threshold t_{sim}, which is to be determined empirically, then add M to T; otherwise, form a new thread based on M. Function $sim(M,T)$ is defined to be the similarity between M and the cluster T. The single-pass clustering algorithm is efficient as it considers each message once. We can not detect the number of threads in advance, but it is at most N where N is the number of messages. Therefore the time complexity is $O(N*N)$. We denote the single-pass clustering algorithm as SP_B.

In SP_B, whenever a new message M is added to a thread T, the centroid of T is updated as the normalized vector sum of messages in T. That is, all messages in a thread contribute to the centroid of the thread equally, no matter when it is added. However, by intuition, in the dynamic text stream,

old messages in a thread should play less important roles than new messages. In fact, we can adjust the performance of SP_B in two directions. By adjusting the threshold t_{sim}, we can obtain clusters at different levels of granularity. By changing the method for calculating centroids of the threads and the form of $sim(M,T)$, we can emphasize different factors which will affect the similarity measurement. We made much effort to exploit the dynamic nature of the text stream and the temporal properties of messages including time window and discounting strategy. These efforts are described in the following variations of SP_B.

In these variants, we exploit the temporal information. For simplicity, we use the messages' relative positions along the message stream to reflect the temporal information, instead of using the absolute occurring time. Besides the following three variants, we can define many others. However, these three are representative enough in that they involve different attempts to exploit the effectiveness of the temporal information.

1. SP with Weighted Centroid (SP_{WC})

   SP_{WC} is the same as SP_B except that the importance of the messages in a thread is determined according to their time of occurrence when calculating the centroid of the thread. The newer a message, the more important it is. The centroid of a thread T can be calculated as shown in equation (1) and then normalized properly.

   $$\vec{T} = \sum_{i=1}^{n} \frac{i}{n} \vec{M}_i \tag{1}$$

   where $\vec{T}$ is the centroid of thread T and $\vec{M}_i$ is the vector of message M_i; all the messages in T are ordered according to the time of occurrence and M_i is the i^{th} message among them; n is the number of messages belonging to T up to the considered message.

2. SP Using Nearest Neighbor (SP_{NN})

   Similar to SP_{WC}, SP_{NN} also takes the dynamic nature of the text stream into consideration but in a different way. A time window of size m is employed in SP_{NN}. SP_{NN} does not use the cosine value between message M and the centroid of T as the similarity measurement. It uses the maximal cosine value between M and the messages belonging to T within the window, which is given in equation (2).

   $$sim(M,T) = \max_{i=1}^{m} cosine(\vec{M}, \vec{M}_i) \tag{2}$$

   where the notations are same as in equation (1).

3. SP Using Weighted Nearest Neighbor (SP_{WNN})

   SP_{WNN} is different from SP_{NN} when calculating the similarity between messages M_i and M. The similarity depends on not only the cosine value between the vectors of these two messages, but also the time distance between them, as shown in equation (3).

   $$sim(M,T) = \max_{i=1}^{m} \frac{1}{dist(M,M_i)} cosine(\vec{M}, \vec{M}_i) \tag{3}$$

   where $dist(M,M_i)$ refers to the time distance between M and M_i in terms of the distance of positions where they appear along the text stream. For example, if M and M_i is the k^{th} and l^{th} message along the text message stream, then $dist(M,M_i)$ is defined as $k-l$. In fact, some more sophisticated forms of distance measurement can be leveraged here, which will be studied in our future work.

4.2.2 Linguistic Features Based Single-Pass Clustering Algorithm (SP_{LF})

As discussed above, the relationships between linguistic features in neighboring messages in a thread are not random, but follow some hidden rules. One way to describe these rules is by *conditional probability*. For simplicity, we assume the dependence satisfies the Markov chain property. That is, the likelihood of a feature in message M_j is entirely determined by the proceeding message M_i in the same thread. Further, we assume that the linguistic features used in the first sentence of message M_j are entirely determined by the last sentence in the proceeding message M_i. For example, in Figure 2, if T_{22} ends with an interrogative sentence which raises a question, it is supposed that T_{25} will begin with a declarative sentence to answer the question. What we are interested in is the likelihood of two messages being neighboring messages in a thread given the linguistic features describing them, that is, the probability $P(T(M_i,\ M_j)|M_i.STL,\ M_j.STF)$ and $P(T(M_i,\ M_j)|M_i.PPL,\ M_j.PPF)$. Given two messages M_i and M_j, we can define a Boolean function $T(M_i,M_j)$ as follows to represent the event where M_i and M_j are the neighboring messages in a thread:

$$T(M_i,M_j) = \begin{cases} 1 & \text{If } M_i \text{ and } M_j \text{ belongs to the same thread} \\ & \text{and } M_i \text{ is the preceeding message of } M_j \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

The probability $P(T(M_i,\ M_j)|M_i.STL,\ M_j.STF)$ and $P(T(M_i,\ M_j)|M_i.PPL,\ M_j.PPF)$ can be estimated based the Bayes Rule according to equation (5) and (6):

$$P(T(M_i,\ M_j)|M_i.STL,\ M_j.STF) = \frac{P(M_i.STL,\ M_j.STF|T(M_i,\ M_j))*P(T(M_i,\ M_j))}{P(M_i.STL,\ M_j.STF)} \tag{5}$$

$$P(T(M_i,\ M_j)|M_i.PPL,\ M_j.PPF) = \frac{P(M_i.PPL,\ M_j.PPF|T(M_i,\ M_j))*P(T(M_i,\ M_j))}{P(M_i.PPL,\ M_j.PPF)} \tag{6}$$

where the parameters on the right side of equations (5) and (6) can be estimated using Maximal Likelihood (ML) directly from the training data. To combine the two different kinds of linguistic features, we use a simple way of calculating $P(T(M_i,\ M_j))$ as follows:

$$\begin{aligned} &P\left(T(M_i,\ M_j)|Linguistic\ Features\ of\ M_i\ and\ M_j\right) \\ &= \ \tfrac{1}{2}(P(T(M_i,\ M_j)|M_i.STL,\ M_j.STF) \\ &+P(T(M_i,\ M_j)|M_i.PPL,\ M_j.PPF)) \end{aligned} \tag{7}$$

Based on $P(T(M_i,\ M_j)|Linguistic\ Features\ of\ M_i\ and\ M_j)$, we now present the linguistic features-based Single Pass Algorithm (SP_{LF}). SP_{LF} is similar to SP_{NN} and SP_{WNN} except that the similarity between messages is not only dependent on the semantic similarity but also the linguistic

features, as shown in equation (8).

$$sim(M,T) = \max_{i=1}^{m} cosine(\vec{M}, \vec{M_i}) * P(T(M_i, M) | Linguistic\ Features\ of\ M_i\ and\ M) \quad (8)$$

An intuitive explanation of equation (8) is that the similarity between messages measured in terms of semantic information is adjusted according to the linguistic features of these messages.

5. EXPERIMENTS

We have described the single-pass clustering algorithm for our problem, as well as three variations of it and a new algorithm based on linguistic features. In this section, we empirically compare these algorithms. We introduce the experimental data set, our evaluation metrics and present the experimental results based on those metrics. The possible reasons for the different methods' performances are also explained.

5.1 Data Set

The data set used in this paper consists of real text streams produced in online conversations among professors, teaching assistants and students in 16 different classes. The thread information is provided by the authors of the messages. For simplicity, we use the numbers 1 to 16 to represent the 16 text streams. The statistical information of the three largest and three smallest streams measured by the number of messages is shown in Table 1 and Table 2. The average number of messages, threads and participants among the 16 streams are 102.8, 23.3 and 26.6 respectively.

Table 1: The 3 Largest Text Streams

ID of Text Stream	4	11	15
Number of Messages	381	181	176
Number of Threads	92	46	39
Number of Participants	68	39	34

Table 2: The 3 Smallest Text Streams

ID of Text Stream	2	16	9
Number of Messages	35	39	46
Number of Threads	7	7	9
Number of Participants	6	9	10

For SP_{LF}, we need the training data to estimate conditional likelihood parameters. Therefore a 3-fold cross validation procedure is applied in the experiments. That is, we randomly split the 16 streams into 3 folds (with one fold containing 6 streams and the other two containing 5 streams each) and at each time, we use two folds as training data and another fold as test data. Though all other thread detection algorithms do not need the training data, they are also tested on the same test data used by SP_{LF} for comparison purposes.

5.2 Evaluation Method

The precision, recall and F measure are commonly used metrics in information retrieval to evaluate the retrieval results [17]. They have been adapted to evaluate the performance of clustering [15]. Here, we explain these metrics in the context of the thread detection problem. For each detected thread, we calculate its precision and recall against each real thread. The F measure is defined by combining the precision and recall together. Specifically, for the detected thread j and the real thread i, the metrics can be calculated as follows:

$$Recall(i,j) = \frac{n_{ij}}{n_i} \quad (9)$$

$$Precision(i,j) = \frac{n_{ij}}{n_j} \quad (10)$$

$$F(i,j) = \frac{2 \times Precision(i,j) \times Recall(i,j)}{Precision(i,j) + Recall(i,j)} \quad (11)$$

where n_{ij} is the number of messages of the real thread i in the detected thread j, n_i is the number of messages in the real thread i, n_j is the number of messages in the detected thread j and F(i, j) is the F measure of the detected thread j and the real thread i.

The whole F measure of the detection result in a stream is defined as a weighted sum over all threads as follow:

$$F = \sum_i \frac{n_i}{n} \max_j (F(i,j)) \quad (12)$$

where the max is taken over all detected threads and n is the total number of messages. The results we report in the next section are in terms of the average F value among all the test streams.

5.3 Experimental Results and Analysis

We compared our proposed algorithm based on linguistic features and the 3 variations of the single-pass algorithm with the basic single-pass algorithm. The results are shown in the tables from Table 3 to Table 10. The results are obtained through three-fold cross validation procedure. The number in boldface is the highest F-value which can be achieved by the corresponding algorithm when tuning the parameters. In the next section, we give a detailed analysis of the parameters tuning procedure and explain why different algorithms reach peak performance under different parameter settings.

Table 3: Performance of SP_B when t_{sim} changes

t_{sim}	0.40	0.46	0.52	0.58	0.64
F	0.351	0.356	**0.361**	0.352	0.338

Table 4: Performance of SP_{WC} when t_{sim} changes

t_{sim}	0.48	0.54	0.60	0.66	0.72
F	0.342	0.392	0.392	**0.395**	0.389

From the results shown in from Table 3 to Table 10, we can order the different algorithms based on the performance: $SP_{LF} > SP_{NN} > SP_{WNN} > SP_{WC} > SP_B$. In the following part, we give an explanation for the performance differences among them.

It is easy to see that the 3 variations of the single-pass algorithm can achieve obvious improvement over the basic single-pass algorithm. This observation validates the effect

of the temporal information. When taking the temporal information into consideration, the variations can remove the impact of the distant messages whose thread is not active any more. Then it is possible for them to assign the right thread label to the currently processed messages.

Table 5: Performance of SP_{NN} when the similarity threshold is fixed as 0.53 (m means the window size)

m	6	7	8	9	10
F	0.521	**0.558**	0.550	0.540	0.486

Table 6: Performance of SP_{NN} when the window size is fixed at 7

t_{sim}	0.49	0.51	0.53	0.55	0.57
F	0.530	0.536	**0.558**	0.532	0.513

Table 7: Performance of SP_{WNN} when the similarity threshold is fixed at 0.42 (m means the window size)

m	10	15	25	30	35
F	0.385	0.411	0.430	**0.435**	0.420

Table 8: Performance of SP_{WNN} when the window size is fixed at 30

t_{sim}	0.36	0.38	0.40	0.42	0.44
F	0.380	0.378	0.421	**0.435**	0.407

Table 9: Performance of SP_{LF} when the similarity threshold is fixed at 0.19 (m means the window size)

m	11	12	13	14	15
F	0.562	**0.612**	0.606	0.591	0.556

Table 10: Performance of SP_{LF} when the window size is fixed at 8

t_{sim}	0.15	0.17	0.19	0.21	0.23
F	0.580	0.608	**0.612**	0.600	0.592

SP_{NN} increases the performance relatively by 41.3% in terms of F-value compared with SP_{WC}. The reason for the improvement is explained as follows. There are two major differences between SP_{NN} and SP_{WC}. The first one is that SP_{NN} only considers the messages within a window with size m; the second is that SP_{NN} determines the thread of the target message based on each single message in each existing thread instead of the centroid of each thread. These differences both contribute to the better performance of SP_{NN}. Due to the dynamic nature of a text stream, that is, some active topics may become inactive when the conversation moves, it is proper to neglect the impact of the old messages in the stream. Therefore, using a sliding window is better than the discounting strategy adopted by SP_{WC}. To get the centroid of a cluster, we need to combine the messages belonging to the cluster. However, after mixing up the messages, it may become harder for us to distinguish two topics if they share some common words. Then, it is better to record all the messages in the existing clusters and determine the similarity between the target message and a thread by computing the similarity between the target message and each distinct message in that thread, which is another reason for SP_{NN}'s better performance.

SP_{WNN} is similar to SP_{NN} except that the former discounts the similarity between the target message and the message in a window according to the distance between the two messages. In fact, in the text stream, one message might not always reply to its nearest neighbor. For example, in Figure 2, T_{25} replies to T_{22} instead of T_{24}. So if we discount the similarity between T_{25} and T_{22}, the similarity may fall below the predefined threshold and they would not be clustered together which leads to an error. This explains why SP_{WNN} is not as good as SP_{NN}.

From Table 9 and Table 10 we can see that SP_{LF} is the best one among the several algorithms including basic single-pass algorithm, and the three variations of basic single-pass algorithm. SP_{LF} improves the performance relatively by 69.5% and 9.7% when compared with the basic single-pass algorithm and the best variation respectively. In fact, SP_{LF} is modified on the basis of SP_{NN} and it takes the advantage of SP_{NN} as shown before. Besides that, SP_{LF} takes the linguistic features into consideration. In fact, in the text stream generated by conversations, the linguistic features play an important role in connecting the messages within a thread. That is why SP_{LF} outperforms SP_{NN}.

5.4 Parameters Tuning

The parameter for SP_B and SP_{WC} is the similarity threshold (t_{sim}) which is easy to tune since the performance of the algorithms depends solely on it. However, for SP_{NN}, SP_{WNN} and SP_{LF}, there are two parameters for each algorithm to tune at the same time. That is the size of the window (m) and the similarity threshold (t_{sim}). It is hard to search over all the parameter space to reach the peak performance for each algorithm. So we adopt a greedy approach by tuning the two parameters alternatively and repeatedly until the performance converges. That is, we first fix m and then tune t_{sim} to find out the best value of t_{sim}. After that, we fix t_{sim} as the best value we just obtained and tune m to get its best value. We repeat this process until the performance of the algorithm does not change any more. The tables from Table 5 to Table 10 show the last two steps for SP_{NN}, SP_{WNN} and SP_{LF} respectively.

From the results shown in Table 5 to Table 10, we can observe that the best values of the window size and the similarity threshold for different algorithms are quite different. We illustrate the observation by the window size. The window size means the reliable range within which we calculate the similarity between messages. If one message is out of the window, we can not rely on it to decide the thread of the target message even if the similarity between them is high. Therefore, it is reasonable that the best window size for SP_{WNN} is much larger than that for SP_{NN} since the former is guaranteed by the discounting strategy. Since SP_{LF} also takes a discounting strategy, based on linguistic features instead of the time distance, the best window size for it is larger than that for SP_{NN} but not as large as that for SP_{WNN}.

6. CONCLUSION AND FUTURE WORK

In this paper, we explored the issue of thread detection in dynamic text message streams, which is considered as a newly emerging kind of data on the Internet. After in-

troducing the stringent definition of the task, we proposed three variations of the single-pass clustering algorithm and a novel algorithm based on linguistic features. The variations take the temporal information into consideration and improve the performance by at most 54.6% when compared with the basic single-pass algorithm. The linguistic features in this paper include Sentence Type and the Personal Pronouns used in messages. By modeling the relationship between neighboring messages in a thread in terms of dependent probability distribution of linguistic features in the messages, our proposed method outperforms the best variation of the single-pass clustering algorithm by 9.7%.

Although we obtained promising results compared to the baseline through our proposed solutions, there is much room for improvement. Firstly, the linguistic features in this paper are relatively simple and the way to identify the features are heuristic. We need to find out more advanced linguistic features which can indicate the relationships between messages more accurately. Secondly, we will try some other sophisticated approaches to combine different linguistic features. Thirdly, the way we utilize temporal information is straightforward which may limit the performance of some algorithms such as SP_{WC}. We will try other approaches in the future. Fourthly, though we mentioned the importance of the end message in a thread, we did not study it explicitly. We will design some specific methods for discovering it. Finally, to validate our proposed algorithms, we will test them on some much larger data sets in our future work.

7. ACKNOWLEDGMENTS

Dou Shen and Qiang Yang are supported by a grant from NEC (NECLC05/06.EG01). We thank the anonymous reviewers for their useful comments.

8. REFERENCES

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking pilot study. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, pages 194–218, 1998.

[2] J. Bengel, S. Gauch, E. Mittur, and R. Vijayaraghavan. Chattrack: Chat room topic detection using classification. In *2nd Symposium on Intelligence and Security Informatics (ISI-2004).*, page 266-277, Tucson, Arizona., June 2004.

[3] E. Bingham, A. Kabán, and M. Girolami. Topic identification in dynamical text by complexity pursuit. *Neural Process. Lett.*, 17(1):69–83, 2003.

[4] E. Elnahrawy. Log-based chat room monitoring using text categorization: A comparative study. In St.Thomas, editor, *Proceedings of the IASTED International Conference on Information and Knowledge Sharing (IKS 2002)*, US Virgin Islands, USA, November 2002.

[5] V. Hatzivassiloglou, L. Gravano, and A. Maganti. An investigation of linguistic features and clustering algorithms for topical document clustering. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 224–231, 2000.

[6] http://www3.gartner.com/3_consulting_services/marketplace/instMessaging.jsp.

[7] X. Ji and H. Zha. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 322–329, 2003.

[8] A. Kabán and M. A. Girolami. A dynamic probabilistic model to visualise topic evolution in text streams. *J. Intell. Inf. Syst.*, 18(2-3):107–125, 2002.

[9] F. M. Khan, T. A. Fisher, L. Shuler, T. Wu, and W. M. Pottenger. Mining chatroom conversations for social and semantic interactions. Technical Report LU-CSE-02-011, Lehigh University, 2002.

[10] J. W. Kim, K. S. Candan, and M. E. Dönderler. Topic segmentation of message hierarchies for indexing and navigation support. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 322–331, 2005.

[11] G. Kumaran and J. Allan. Text classification and named entities for new event detection. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 297–304, 2004.

[12] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[13] G. Salton and M. Smith. On the application of syntactic methodologies in automatic text analysis. In *SIGIR '89: Proceedings of the 12th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–150, 1989.

[14] A. F. Smeaton. Progress in the application of natural language processing to information retrieval tasks. *Comput. J.*, 35(3):268–278, 1992.

[15] K. G. Steinbach, M. and V. Kumar. A comparison of document clustering techniques. Technical report 00-034, Department of Computer Science and Engineering, University of Minnesota, 2000.

[16] V. H. Tuulos and H. Tirri. Combining topic models and social networks for chat data mining. In *WI '04: Proceedings of the Web Intelligence, IEEE/WIC/ACM International Conference on (WI'04)*, pages 206–213, Washington, DC, USA, 2004. IEEE Computer Society.

[17] R. C. van. *Information Retrieval*. Butterworths, London, second edition edition, 1979.

[18] A. Waibel, M. Bett, M. Finke, and R. Stiefelhagen. Meeting brower: Tracking and summarizing meetings. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[19] W. Xi, J. Lind, and E. Brill. Learning effective ranking functions for newsgroup search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 394–401, 2004.

[20] Y. Yang, T. Pierce, and J. Carbonell. A study of retrospective and on-line event detection. In *SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 28–36, 1998.