Topic Detection and Tracking for Threaded Discussion Communities

Mingliang Zhu Weiming Hu Ou Wu
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
{mlzhu, wmhu, wuou}@nlpr.ia.ac.cn

Abstract

The threaded discussion communities are one of the most common forms of online communities, which are becoming more and more popular among web users. Everyday a huge amount of new discussions are added to these communities, which are difficult to summarize and search. In this paper, we propose a topic detection and tracking (TDT) method for the discussion threads. Most existing TDT methods deal with the news stories, but the language used in discussion data are much more casual, oral and informal compared with news data. To solve this problem, we design several extensions to the basic TDT framework, focusing on the very nature of discussion data, including a thread/post activity validation step, a term pos-weighting strategy, and a two-level decision framework considering not only the content similarity but also the user activity information. Experiment results show that our proposed method greatly improves current TDT methods in real discussion community environment. The discussion data can be better organized for searching and visualization with the help of TDT.

1. Introduction

As the web getting larger and more popular, the users of the web are becoming more and more active. The web nowadays is not only a media to spread information, but also a place for people to express themselves. As a result, all kinds of web communities are attracting users all around the world. For example, the ShuiMu Community [5], one of the biggest BBSes in China, typically has more than 10,000 users logged-in at the same time and averagely has about 100,000 new posts added each day.

The threaded discussion communities are one of the most common forms of web communities. Typical discussion forums and BBSes are examples of them. Here are some basic concepts for threaded discussion communities (refer to Figure 1):

Post: each time a user says something is called a post. It is the atom object in discussion communities. A post is attached with 4 properties: the timestamp, the author (the user who makes the post), the title (all posts in a *thread* share the same title, see below) and the content (what the user says).

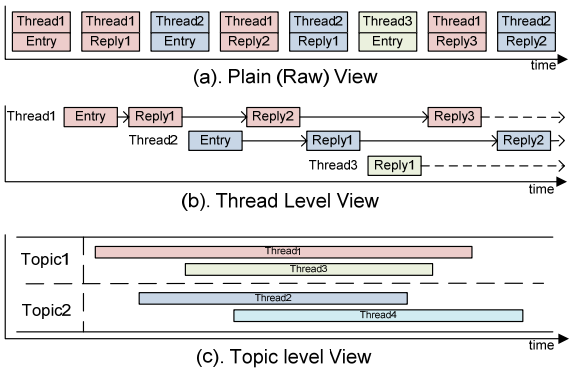

Figure 1. Different levels of views on threaded discussion communities

Thread: posts are organized in threads. Each post belongs to and belongs only to one thread, and a thread consists of a series (one or more) of posts. The first post in thread is called the *entry*, which proposes some subject to talk about, and set up the *title* for the thread at the same time. Each of the other posts in the thread is called a *reply*, since its content is this user's reply to either the *entry* post or a previous *reply* post.

Board: boards are subsections of a community, in which discussions are all within the same fields. The name "board" came from BBSes, but is used to describe the same concept in other communities. A typical comprehensive community may have all kinds of *boards*, such as sports, music, computer-tech, etc.

The discussion data is a great challenge to search engines. The query based search model does not work well on the post level or even thread level. On the other hand, the community users may want a summary of the bulk amount of the discussion data to see "what's hot" or "what's going on". The simple "reply-count" strategy is not good enough for this demand.

Usually the discussions are with some trends: most posts are just discussing a small number of *topics* (will be verified in the Experiment section). One common situation is that when something great happens in people's lives, there may be different threads discuss-

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.50

ing different aspects of this certain event, such as the event itself, its background, the people or places related to the event. Another example is that when someone proposes some interesting subject in a thread to talk, others that have same or opposite ideas may open new thread to claim their opinions; on the other hand, as the discussions go on, similar or related subjects may be proposed in new threads.

An automatic online algorithm for topic detection and tracking (TDT) for threaded discussion communities is proposed in this paper, which is an extension for traditional topic detection and tracking [1] algorithms. Traditional TDT mainly deals with news stories. But the content in online communities is quite different from common news stories, making it much more difficult to handle. The first problem with discussion data is that the language used is more oral, casual and informal. Even worse, misspellings and "Internet slangs" appear heavily in discussions. These make discussion contents not easy to understand even for humans with little online experience. The second problem is that the subject of a thread is often implicit in its content. For example, in a series of threads discussing a soccer game, a thread with title "I think No. 18 should be substituted for!" and with replies saying "He plays awful today", "Yeah, much worse than last week against …" would appear. But literally there may be no apparent connections between this thread and others of the same event. A third problem is that as the discussion goes on, the users may become away from the original topic. One common example is that acquaintance users may say "hello", "what's going on" and begin to talk about trivial matters in each other's lives. In contrast, the language used in news stories is always formal, accurate and all element of the event should appear clearly in the content.

We propose a set of extensions to the basic model that is widely used in traditional TDT tasks to address the problems in threaded discussion communities, including: (1) a post and thread activity validation step is introduced to filter out posts and threads that do not provide informative contents. Uninformative contents bring a lot of noise. (2) a term pos-weighting strategy is designed for discussion data so that the analyzing can focus on the central part of the content. And (3) the user activities (authorship information) are taken account to topic detection and tracking - posts are submitted by different community users, and this is a major difference between discussion data and news data. Each user has respective interest and pattern, and so that the consideration of user activity is a great complement to the discussion contents.

With the topic detection and tracking, the discussion community data can be organized and indexed at a higher level, making it much easier to search and visualize. Mining can be done within topics to extract their underlying trends. Furthermore, besides online communities, many other data have the threaded structure, such as email messages. Similar algorithm may also help to improve the user experience for these data.

2. Related work

Topic detection and tracking (TDT) has been widely studied for years [1][3][7][8][12][14][15]. Most of them are designed for analyzing news stories, which are much "cleaner" compared with discussion content in online communities. As a result, the existing techniques may not achieve good performance on discussion contents, which is proved in our experiments (see section 4). Among the online TDT algorithms, the incremental TF-IDF model [1][3][14][15] is one of the dominant content relevance measurements.

A couple of related work on online communities was reported. Kim et al [6] proposed a method to segment topics in a single discussion thread (hierarchy). Topic segmentation is also developed for online chat data, such as in [2] and [13]. All these addressed the problem of topic analysis in the online community environment, but they actually deal with the problem of segmenting different topics in a single thread (all chat contents in a single chat room or IRC channel can be viewed as a single data-stream) rather than detecting topics among multiple threads. The segmenting problem is then transformed into finding certain posts that the topic changes. In our problem, different topics are overlapped and multiple topics may be discussed at the same time, so these segmentation methods cannot be applied to solve our problem. There is also related work on community user modeling. Social networks [9] are widely used to model the user interactions in online communities. Steyvers et al [11] proposed a model combining the authorship information and the latent topic model for text mining.

3. Topic Detection and Tracking for Threaded Discussion Communities

3.1. Overview

Our method for topic detection and tracking for threaded discussion communities is described in this section. Figure 2 demonstrates the outline of our method. Every time a new post comes all its properties is extracted in the pre-processing step. Then the activity validation is taken for the post and the thread containing the post. If the thread is active enough, its content status and user status is updated and the topic list is

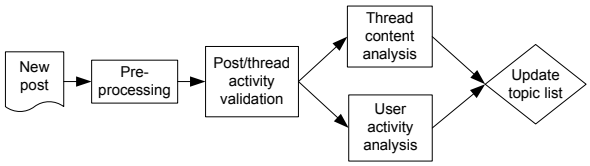

Figure 2. Outline of the algorithm framework

then updated based on the result of both the content analysis and the user activity analysis.

3.2. Pre-processing and post/thread activity validation

Posts are the atom objects in threaded discussion communities. The pre-processing step extracts the structural properties from the raw post data, including the post title, author, posting time, content, as well as the thread id which the post belongs to. The title (only for entry posts; the reply posts always share the same title with the entry) and the content is further tokenized into term sequences, and stop-words are removed.

After pre-processing, a post activity validation step is taken, in which the *informativeness* of posts are tested. In the online discussion community environment, there are many posts with little information about what topic they are talking about, such as "That's great", "I agree with you". These posts bring a lot of noise for topic analysis. Filtering out uninformative posts is a little like commonly used stop-word removal, but provides control of useless information on the whole post level rather than term level, which may be more precise and effective.

A standard one-class SVM classifier [10] is used for post activity validation. One-class SVM was proposed by Schölkopf [10] for estimating the support of a high-dimensional distribution, and so that is able to generate classifiers based on a training set that is consist of only positive (or negative) samples. In our problem, the uninformative (negative) posts may have common patterns but informative (positive) posts may vary a lot. It is not possible to provide a training set to cover the distribution of all positive samples. In our method, the post activity validation classifier is trained by a set of pre-labeled negative (uninformative) posts. The details of the one-class SVM can be found in the original paper, and here we only show some key formulas.

Let $\boldsymbol{x}_i \in \mathbb{R}^n$, $i = 1, \ldots, l$ be the pre-labeled negative posts, where each $\boldsymbol{x}_i$ is the term frequency (TF) vector of the content (title is not included) of each post, n is the size of the vocabulary and l is the number of the training samples. Let k be a kernel function and Φ be the corresponding feature map that maps the feature space $\mathbb{R}^n$ into an inner product space $\mathcal{F}$:

$$\Phi : \mathbb{R}^n \to \mathcal{F},$$
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j)). \quad (1)$$

In our method, the typical Gaussian kernel is used:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / c} \quad (2)$$

To separate the date set from the outliers, the following quadratic program is to be solved:

$$\min_{w \in F, \xi \in \mathbb{R}^l, \rho \in \mathbb{R}} \quad \frac{1}{2}\| w \|^2 + \frac{1}{vl}\sum_i \xi_i - \rho \quad (3)$$
$$\text{subject to} \quad (w \cdot \Phi(\boldsymbol{x}_i)) \geq \rho - \xi_i, \xi_i \geq 0$$

where $v \in (0,1]$ is a parameter to control the trade-off between the coverage and the "volume" of the distribution region learned, and ξ_i are non-zero slack variables.

Suppose w^* and ρ^* (ξ_i are penalized in the objective function in equation (3)) are the solution to the target in equation (3), then the decision function is:

$$f(\boldsymbol{x}) = \text{sgn}(\rho^* - (w^* \cdot \Phi(\boldsymbol{x})) \quad (4)$$

where output -1 indicates the post to be uninformative (within the distribution of the training set) and +1 to be informative. The details of solving the quadratic program in equation (3) are beyond the scope of this paper and can be found in [10].

Threads that only contain a single uninformative post are treated as *inactive*. Each inactive thread is determined to be within a separate topic. As we examined real discussion transcripts, these inactive threads are usually paid little attention to and are likely to be forgotten soon with no further discussion on its topic. However, if new posts come into an inactive thread, it shall be activated and its topic status is updated. But all uninformative posts are still ignored in the following content analysis step.

3.3. Content similarity and term weighting

Content similarity is strong evidence that different threads are in the same topic. The thread content similarity calculation in our algorithm is based on the typical incremental TF-IDF model which is widely used in traditional TDT algorithms. We extend the base model with a particular term weighting strategy designed for discussion threads, as well as a modified frequency update strategy for the online community environment.

In our algorithm, a pos-weighted term frequency (TF) vector is kept for each seen thread and a global document frequency (DF) vector is kept among all threads. The frequency values in TF are weighted by the position at which the term appears in the thread. The pos-weight w_{pos} is assigned as follows

$$w_{pos} = \begin{cases} 5, & \text{terms in } \textit{title} \\ 1, & \text{first 40 terms in } \textit{entry} \text{ (if informative)} \\ 1, & \text{first 15 terms in each of the first 16 informative } \textit{replies} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The pos-weighted strategy defined in equation (5) favors terms appeared in the title, and the contents are regulated to certain lengths. These make the TF vectors focus on the most informative part of a thread. Furthermore, all post since the 18th are simply eliminated in the TF vector because (1) the discussion may get away from the original topic and turn into some trivial matters, while there is usually enough information to determine the topic for a thread in its first 17 posts, and (2) according to the definition of w_{pos}, the TF vector do not have to be updated any more after seeing 17 post (the time at which the newest post in the thread comes is still updated however, which is to be used later). Keeping 17 posts is a balance of reserving enough information of a thread and the consumption of computing resources. The DF value for a term indicates how many of seen threads contain this term.

Both the TF and DF vectors are updated incrementally and periodically. The initial DF vector DF_0 is generated from a (could be empty) base set $\mathcal{B}$, and is updated at time t as:

$$df_t(w) = df_{t-1}(w) + df_{Ct}(w) \quad (6)$$

where $df_t(w)$ is the DF value for term w at time t, and $df_{Ct}(w)$ is the number of threads whose new coming posts during time span t-1 and t contain term w while old posts before time t-1 never contained w. The TF vector for a thread is initialized when it is opened, and is updated at time t as:

$$tf_t(d, w) = tf_{t-1}(d, w) + tf_{Ct}(d, w) \quad (7)$$

until 17 informative posts of it have arrived (as discussed earlier in this section). $tf_t(d, w)$ denotes the pos-weighted TF value for term w in thread d at time t, and $tf_{Ct}(d, w)$ denotes the pos-weighted count of term w in new coming posts of thread w (if any) during time span t-1 and t.

The Hellinger distance is used to calculate the content similarity of two threads:

$$csim_t(d_1, d_2) = \sum_w \sqrt{w_t^{tf \cdot idf}(d_1, w) \cdot w_t^{tf \cdot idf}(d_2, w)} \quad (8)$$

Where $w_t^{tf.idf}(d, w)$ is the TF-IDF weight of term w in thread d at time t, which is calculated based on the TF and DF vectors:

$$w_t^{tf \cdot idf}(d, w) = \frac{1}{Z_t(d)} tf_t(d, w) \cdot \log \frac{N_t}{df_t(w)} \quad (9)$$

where N_t is the total number of seen threads (including those in the base set $\mathcal{B}$), by which the DF vector is generated. $Z_t(d)$ is a normalization factor:

$$Z_t(d) = \sum_w tf_t(d, w) \cdot \log \frac{N_t}{df_t(w)} \quad (10)$$

3.4. User activity analyze

One significant difference between discussion threads and other web documents is that threads consist of posts composed by different community users. So it is natural to utilize user information for topic analysis of discussions. One of the most widely used community user model is the social network [9], which describes relationships among community users. But it seems that the user interactions may not directly help to infer the topic of a thread. Alternatively, we make use of the following assumption: a certain community user is interested in some certain topics, and tend to discuss in threads that belong to the topics that he or she is interested in. There are already some text mining methods using similar authorship assumption such as in [11]. Based on this assumption, if a certain number of users take part in both of two threads, these two threads are probably in the same topic. In our algorithm, we introduce a UF-ITUF (user frequency-inverse thread user frequency) model to measure the similarity of the group of users who take part in two threads.

The UF-ITUF model is much like the TF-IDF model which is used to calculate content similarity. A UF vector $[uf(d_i, u_1), uf(d_i, u_2), \ldots, uf(d_i, u_n)]$ is maintained for each thread d_i, where $uf(d_i, u_j)$ is the number of posts composed by user u_j in d_i, and n is the total number of users in the community. A global TUF vector $[tuf(u_1), tuf(u_2), \ldots, tuf(u_n)]$ is also maintained where $tuf(u_i)$ denotes the number of threads in which user u_i ever posted. The TUF is used as a punishment term (ITUF) in similarity calculation. Intuitively, if a user posted in a lot of threads, his or her participation is not expected to be as distinguishing as those who posted in fewer threads. Given the UF and the ITUF vectors, the Hellinger distance can be used as user group similarity measure of two threads:

$$usim(d_1, d_2) = \sum_u \sqrt{w^{uf \cdot ituf}(d_1, u) \cdot w^{uf \cdot ituf}(d_2, u)} \quad (11)$$

where $w^{uf.ituf}(d, u)$ is the UF-IDUF weight of user u in thread d, which is defined similar to the TF-IDF

weight used in section 3.3:

$$w^{uf\cdot ituf}(d,u) = \frac{1}{Z_{(u)}(d)} uf(d,u) \cdot \log \frac{N}{tuf(u)} \qquad (12)$$

$$Z_{(u)}(d) = \sum_{u} uf(d,u) \cdot \log \frac{N}{tuf(u)}$$

The UF and TUF vectors are also updated incrementally. When new posts/threads come, the new authorship information is added to the UF and TUF vectors maintained.

3.5. Making the decision

The content similarity and user activity similarity are combined to decide whether a thread starts a new topic or discusses the same topic as some previous threads. At the end of each time period, after the content and user activity data is updated, the thread similarities are then calculated, and so as the topic collection is to be updated. Different from news stories, the posts of a thread do not come at the same time, but come one after another arbitrarily. At a certain time point, it is even not possible to tell whether a thread is finished (unless it is deleted from the community), although it may have been quiet for a while. As a result, the topic class of a thread may be updated when new posts of it or of previous threads come. Apparently, it is unrealistic to recalculate similarities for all seen threads every time to update the topic collection. However, topics on discussion communities usually have limited life times. In most cases, there would be no further discussion on a topic if there has been no post in that topic for about two days. As a result, threads whose latest post came more than two days ago are not included in topic collection update. Furthermore, the TF vector for a thread will not change after seeing 17 informative posts (refer to section 3.3) and after 20 for UF vector, so the topic class *fixes* for a thread after receiving 20 posts in our method, even if its previous thread may still be updating. These two strategies guarantee the online system to be running in real-time.

To determine the topic of a thread d_0, first we search for the thread d_c^* within the 2-day window w_{2d} that has the highest content similarity to d_0:

$$d_c^* = \arg\max_{d' \in win_{2d}} csim_t(d_0, d') \qquad (13)$$

and the corresponding similarity value $csim_t(d_0, d_c^*)$ is compared with a threshold θ_{c1}. If exceeds the threshold, the target thread d_0 is classified to be in an old topic. Otherwise, d_0 is *possibly* discussing a new topic. Similarly, for the user analysis results, the thread d_u^* in the 2-day window with highest user similarity to d_0 is searched for:

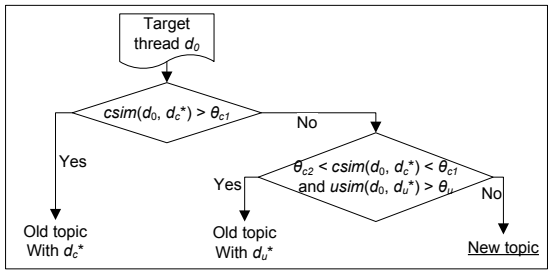

Figure3. Two-level decision process

$$d_u^* = \arg\max_{d' \in win_{2d}} usim_t(d_0, d') \qquad (14)$$

However, different from content similarity, the user group similarity can only be used as "verification" but not "falsification": we can decide that two threads are of the same topic if their content is similar while of different topics if the content differs, but for user group similarity, usually only the first statement holds, while two threads may still be discussing the same topic when they are conducted by different groups of users. To solve this problem, a two-level decision strategy is designed to combine the content and the user analysis results, illustrated in Figure3. An extra threshold θ_{c2} for content similarity is introduced satisfying:

$$0 < \theta_{c2} < \theta_{c1}. \qquad (15)$$

If the d_c^* found satisfies $csim_t(d_0, d_c^*) > \theta_{c1}$, then thread d_0 is decided to be in the same topic as thread d_c^*. If $csim_t(d_0, d_c^*) < \theta_{c2}$, thread d_0 should be discussing a new topic. And finally if $\theta_{c2} < csim_t(d_0, d_c^*) < \theta_{c1}$, the user group similarity is further considered: if $usim_t(d_0, d_u^*) > \theta_u$, d_0 is supposed be of the same topic as thread d_u^*, or otherwise d_0 is discussing a new topic. Since there are three thresholds to make the decision, the parameter tuning is a little complicated than deciding only with contents similarities. First, candidate values for θ_{c1} and θ_u can be found by optimizing the performance with only content decision and only authorship decision separately (authorship is also used for "falsification" in this step). Then, the combination of the three parameters is searched around their possible values: θ_{c1} and θ_u are around their candidates values found in the previous step, and θ_{c2} is regulated in equation (15).

4. Experiments

In order to evaluate the performance of our proposed method, intensive experiments are conducted and the results are reported in this section.

Since there are no public data sets for threaded discussion communities, we create our experiment data sets by downloading posts from real online communities. The data are from the "NewExpress" board on the ShuiMu community [5], which is one of its most popular boards. All posts during Feb. 22, 2008 and Mar. 10, 2008 are downloaded by our spider. The raw post data are then parsed and post properties are extracted, including: the timestamp, the author, the title and the content. The thread relations of posts are also extracted. System posts such as community notifications are excluded from the data sets. There are totally 122307 posts of 13707 threads in the data set, averagely 6794.8 posts and 761.5 threads every day.

The data are then divided into two subsets: (1) the base set $\mathcal{B}$ for training the classifier in the post/thread activity validation step and generating the initial DF and TUF vectors; and (2) the testing set $\mathcal{T}$ for testing the performance of our proposed algorithm. The posts during Feb. 22 and Feb. 29 are used as the base set $\mathcal{B}$ and posts during Mar. 1 and Mar. 10 are used as the test set $\mathcal{T}$ (the posts after Mar. 1 but belonging to threads started before Mar. 1 are still put in $\mathcal{B}$). 9014 posts in set $\mathcal{B}$ are manually labeled with "informative" or "uninformative", in which 3503 uninformative posts are used to train the activity validation classifier. The LIBSVM library [4] is used for one-class SVM training and classification in our experiments. The threads in set $\mathcal{T}$ are manually clustered into topic collections, which are used as ground truth in the experiments. Totally 2980 topics are identified. A majority of the topics (more than 2000) consists of only one thread, but the rest (topics that consist of more than one threads) cover over 68% of all posts. That is to say, although there are a lot of topics, most of the posts are discussing only a small number of topics. This shows the necessity of DTD on the discussion communities.

The C_{Det} evaluation metric which is widely used in TDT methods is used to evaluate the performance of our method. To measure the performance, the TDT is divided into two sub tasks: new topic detection (NTD) and topic tracking (TT), each generating a Yes/No output. The NTD determines whether a thread is discussing a new topic that is never seen before. The TT task determines whether a given thread belongs to a given topic. By testing on each decision instance for the two tasks (each thread for the NTD, and each topic-thread pair for the TT), the decision error P_{Miss} (miss rate) and P_{FA} (false alarm rate) is calculated. *Miss* occurs when the system to output No for a Yes test instance, while *false alarm* is for outputting Yes for a No test instance. The C_{Det} metric is defined by combining P_{Miss} and P_{FA}:

Table 1. Summary of the data sets

	Base set $\mathcal{B}$	Test set $\mathcal{T}$	Total
Post	54340	67967	122307
Thread	6444	7263	13707
Informative label	5511*	N.A.	5511
Uninformative label	3503*	N.A.	3503
Topics	N.A.	2980	2980

* Only 9014 post in Base set $\mathcal{B}$ are labeled with informative or uninformative.

Table 2. Minimum normalized C_{Det} cost

	NTD	TT
Full system: Act. valid., term weight, user analysis	**0.759**	**0.363**
Act. validation, term weighting	0.788	0.380
Base system: title + content, no term weight	0.857	0.423

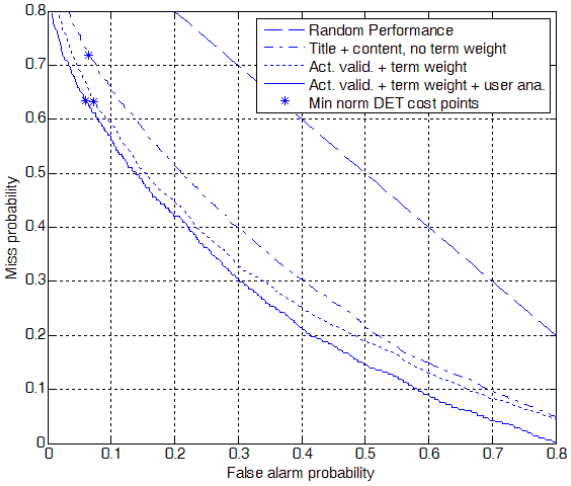

Figure 4. DET curve for new topic detection

$$C_{Det} = C_{Miss} \cdot P_{Miss} \cdot P_{target} + C_{FA} \cdot P_{FA} \cdot P_{nontarget} \quad (16)$$

C_{Miss} and C_{FA} are costs for misses and false alarms. C_{Miss} = 1 and C_{FA} = 0.2 are used in the experiment. P_{target} is the probability of seeing a Yes instance and $P_{nontarget}$ is the probability of seeing a No instance. Appearently P_{target} = 1 - $P_{nontarget}$ holds. The P_{target} is set to 0.3 for new topic detection and 0.02 for topic tracking. The reported C_{Det} cost is normalized as:

$$(C_{Det})_{norm} = \frac{C_{Det}}{\min(C_{Miss} \cdot P_{target}, C_{FA} \cdot P_{nontarget})} \quad (17)$$

The minimum C_{Det} costs are shown in Table 2 (parameters are tuned for NTD only, and results for TT are using parameters that minimize C_{Det} for NTD, since TT shares the same parameters with NTD and cannot

be separately tuned). Since we found no topic detection and tracking algorithms proposed for discussion communities, the classical content-only method (without thread/post activity validation, and no term weighting for post title and content) is used as a baseline for comparison. Also the result with thread/ post activity validation and term weighting but without user analysis is reported. The detection error tradeoff (DET) curve of the NTD is shown in Figure 4 (the curve for TT is not shown since parameters for the TT is not separately tuned in our problem).

The final results show that our proposed algorithm constantly outperforms the base system, yielding an improvement of 0.098 in NTD and 0.060 in TT to the minimum normalized C_{Det} cost. Considering Figure 4, the post/thread activity validation and term pos-weighting improved the minimum C_{Det} cost for the NTD by 0.069, but the improvement is not much on the side of curve that favors low miss rate. The main reason of this is that for many of the discussion threads, their topic-distinguishing terms are implicit but can be implied in their context. Most terms in these threads are oral and trivial, so their content similarity to other threads are small and they are often be decided as a new topic. The results based on both content and user information outperforms the base system on both side of the curve, which proves that the user activity information is a strong complement to the content evidence, especially for threads whose topic is implicit.

5. Conclusion and future work

The online communities are becoming more and more popular along with the explosive development of the web. In this paper, we have proposed an algorithm to accomplish the topic detection and tracking task (TDT) in the threaded discussion community environments. Different from the news stories which most TDT methods deal with, the language used in online discussions is much more casual, oral and informal, making it much more difficult to recognize. We have introduced a thread/post activity validation step, a term pos-weighting strategy, and a two-level decision framework considering both content similarity and user activity information to improve the based system used in traditional TDT methods. The experiment results have shown the effectiveness of our proposed method. With the topics being analyzed, the discussions in online communities can be categorized and organized at a higher level, making the searching and visualization on the discussion data much easier. However, the performance of TDT on discussion data is still not as good as those of news stories, partly because of the very nature of discussion threads. In our future work, we will keep looking for methods to further improve the performance of the TDT system for online communities.

6. Acknowledgment

This work is partly supported by NSFC (Grant No.60672040, 60705003) and the National 863 High-Tech R&D Program of China (Grant No. 2006AA01Z453).

7. References

[1] J. Allan, J. Carbonell, G. Doddington, J. Yamron and Y. Yang. "Topic detection and tracking pilot study: Final report". In *Proc. of DARPA Broadcast News Transcription and Understanding Workshop*, 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*, Tucson, Arizona, 2004, pp. 266-277.

[3] T. Brants, F. Chen and A. Farahat. "A System for New Event Detection". In *Proc. of ACM SIGIR'03*, 2003, 330-337.

[4] C.-C. Chang and C.-J. Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] http://www.newsmth.net/

[6] J. W. Kim, K. S. Candan and M. E. Dönderler, "Topic segmentation of message hierarchies for indexing and navigation support", in *Proc. of WWW'05*, 2005, 322-331.

[7] G. Kumaran and J. Allan. "Text Classification and Named Entities for New Event Detection". In *Proc. of ACM SIGIR04*. 2004, 297-304.

[8] J. Makkonen, H. Ahonen-Myka and M. Salmenkivi, "Simple Semantics in Topic Detection and Tracking", *Information Retrieval*, Springer, 2004, 347-368.

[9] N. Matsumura, D. E. Goldberg and X. Llorà, "Mining directed social network from message board", in *Proc. of WWW'05*, 2005, 1092-1093.

[10] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola and R. C. Williamson. "Estimating the support of a high-dimensional distribution". *Neural Computation*, 2001, 13(7), 1443-1471.

[11] M. Steyvers, P. Smyth, M. Rosen-Zvi and T. Griffiths, "Probabilistic author-topic models for information discovery", in *Proc. of ACM SIGKDD'04*, 2004, 306-315.

[12] N. Stokes and J. Carthy. "Combining Semantic and Syntactic Document Classifiers to Improve First Story Detection". In *Proc. ACM SIGIR'01*. 2001, 424-425.

[13] V. H Tuulos and H. Tirri, "Combining Topic Models and Social Networks for Chat Data Mining", In *Proc. of WI'04*, 2004, 206-213.

[14] Y. Yang, T. Pierce and J. Carbonell. "A Study of Retrospective and On-line Event Detection". In *Proc. of ACM SIGIR'98,* 1998, 28-36.

[15] K. Zhang, J. Li and G. Wu. "New Event Detection Based on Indexing-tree and Named Entity". In *Proc. of ACM SIGIR'07,* 2007, 215-222.