Detecting a Multi-Level Content Similarity from Microblogs Based on Community Structures and Named Entities

Swit Phuvipadawat, Tsuyoshi Murata
Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology, Japan
Email: swit.p@ai.cs.titech.ac.jp, murata@cs.titech.ac.jp

Abstract— **This paper presents a method for finding the content similarity for microblogs. In particular, we process data from Twitter for a breaking news detection and tracking application. The goal is to find a collection of similar messages. The method gives two levels of collections. In the first level, similarity is defined by TF-IDF. Since contents in microblogs have short lengths, we emphasize on specific terms called named entities. Message groups are obtained in the first level. In the second level, we construct a network from the message groups and named entities and perform a community detection. We evaluate and visualize the community results based on several community detection algorithms. We demonstrate that this method can be used to explore similar messages with results in both tightly and loosely coupled manners.**

Index Terms— **Twitter, Topic Detection and Tracking, Information Retrieval, Network Analysis**

I. Introduction

The emergence of microblogging has transformed the way people communicate and exchange information. A microblog is a minimal form of blog having two distinctive characteristics: brevity—contents are in short length and simultaneousness—contents are updated frequently. Examples of microblogging services are Twitter[1], Tumbler[2], Google Buzz[3], etc.

Twitter has emerged itself into a communication medium for hot topics and breaking news. In June 2009, Twitter has played an important role in delivering user-generated contents from the Iranian citizens in the Iran elections. We see that people with technology played a role of journalists in the situation where news reporting in a conventional way has been made difficult [1]. Anyone who are not associated to the media industry could also deliver news. Thus, Twitter presents a highly effective way to discover what is happening around the world.

The reason we emphasize Twitter in our microblogging study is that, Twitter does not only gain interests from the Internet users but also from research perspectives as well. From the Internet users perspective, as of April 2010, Twitter has over 105 million registered users with 300,000 new users per day [2]. From the research perspectives, we give the following examples. The study on the topological characteristics of Twitter found that 85% of trending topics in Twitter appear in headline news [3]. There is a proposed method of using Twitter data to improve web ranking [4]. It is interesting to see that microblogging data reveals fresh URLs not yet fetch by search engines. We can also use Twitter for event detections. Earthquakes and typhoons for examples, can be detected by treating Twitter users as sensors [5]. This demonstrate the realtime nature of Twitter data. There are also research on influential users and topics finding [6], [7].

Breaking news and top trending topics are posted in Twitter. However, they require an effort to discover it. Firstly, users often have problems of deciding which users to follow. That is, to find users with interesting tweets [8]. Secondly, users need to read through status updates and follow links to obtain further information. To ease these problems and to deliver breaking news effectively, we propose a method to collect, group, rank and track breaking news in Twitter. This work is a contribution to the area of Topic Detection and Tracking (TDT) [9]. The tasks we focus are first story detection, cluster detection, and tracking.

In our method, similar messages are grouped together based on cosine similarity of TF-IDF vectors. We tackled the problem of similarity comparison for short-length messages by carefully boosting scores for named entities or proper noun terms. The assumption is that named entities are crucial elements in messages as they provide identity information to the subjects. Named entities can be classified as person, organization or location using a name entity recognizer.

The grouping result from the above mentioned method gives message groups that contain messages similar to each other in terms of vocabulary. However, similar messages may be represented differently with different choices of vocabulary and thus similar messages by means of semantics might not be grouped together correctly. In this study, we will demonstrate the use of network

This paper is based on "Breaking News Detection and Tracking in Twitter" by S. Phuvipadawat and T. Murata, which appeared in the 2010 Workshop on the Intelligent Web Interaction (IWI-10), Toronto, Canada, August 2010. © 2010 IEEE.

[1]http://twitter.com
[2]http://tumblr.com
[3]http://google.com/buzz

doi:10.4304/jetwi.3.1.11-19

constructed from relationships between message groups and name entities to find communities of similar message groups. To this effect, we can find relevant communities of message groups based on named entities.

The rest of the paper is organized as follows: Section II describe the nature of contents in microblogs based on the breaking news detection and tracking application. Section III describes the method for collecting, grouping and ranking breaking news from microblogs. Section IV discusses the effects of term score boosting for named entities. Section V describes the message groups - named entities network, which is a result from the execution of method described in section III. Section VI describes the dataset used in the experiment. Section VII describes the community detection methods conducted on the message groups - named entities network. Section VI evaluates and shows the results of our experiment. Section IX discusses the experiment results and the possible application. Section X concludes our work.

II. Nature of Contents in Microblogs

This section describes the nature of contents in microblogs based on the breaking news detection and tracking application. We investigated both microscopic and macroscopic levels of the contents in microblogs which are identified as single message and timeline aspects.

A. Single message aspect

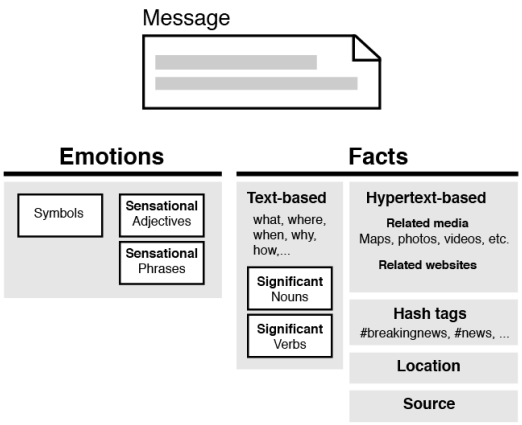

Figure 1. Emotions and facts in a message

There are two important elements in a message: emotions and facts as shown in Figure 1. The inclusion of emotions in the message makes news delivered in Twitter different from news delivered by professional journalists. Although there are cases where emotions are conveyed in conventional news, expression of emotions occurs much more often in Twitter messages. Emotions are expressed through the use of symbols (mainly the use of exclamation mark '!'), the use of sensational adjectives and phrases: crazy, amazing, great, terrible, wonderful, shocking, oh my god, etc.

Facts are provided in text-based, hypertext-based, and through location and source information of the message originator. Text-based information is highly significant as it helps interrogate the details of the news in terms of 'what', 'where', 'when', 'how', etc. We can identify keywords from facts that contribute to news stories. These keywords are identified as significant nouns and verbs. Significant nouns include keywords found in conventional news, names of famous places, people and events such as Japan, US president, emergency and airplane. Significant verbs are, for examples, fire, crash, bomb, survive, rescue, win, etc. Users often tag their message with a hash symbol (#) followed by keywords for examples, #breakingnews, #haiti, etc. as a mean to group together messages related to the keywords. Hypertext-based facts provide related information from external sources. To cope with the limitation of the message length, users often use a link shortening services like TinyURL[4] and bit.ly[5]. In addition to texts, users often include maps and pictures. Maps are provided by online services like Google Maps[6] and Yahoo Maps[7]. Pictures are hosted on services like TwitPic[8] , yfrog[9], etc.

B. Timeline aspect

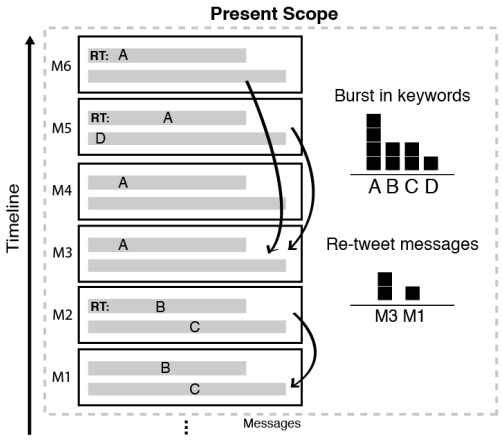

Figure 2. Burst in keywords and retweeted messages

From the timeline aspect shown in Figure 2, we can observe the burst in keywords and the number of retweeted messages through the passage of time. Interesting or important messages tend to be retweeted more than the others. For the case of breaking news we can see the development of news stories through the series of messages. We will use these facts to determine the rankings for a collection of messages. The method is described in section III-D.

[4]http://tinyurl.com
[5]http://bit.ly
[6]http://maps.google.com
[7]http://maps.yahoo.com
[8]http://twitpic.com
[9]http://www.yfrog.com

III. A METHOD FOR COLLECTING, GROUPING AND RANKING MESSAGES FROM MICROBLOGS

In this section, we describe the method prior to the community structure analysis. Tasks are divided into two stages: story finding and story development. The overview of the process is shown in Figure 3.

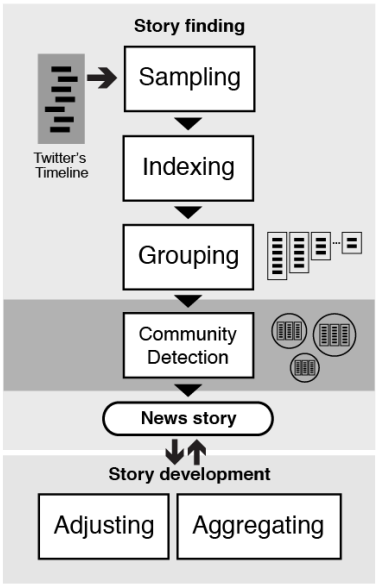

Figure 3. Two stages: story finding and story development

A. Story finding stage

In this stage, the tasks are presented in four steps: sampling, indexing, grouping and community detection.

1) *Sampling*: In this experiment, messages are fetched through the Twitter streaming API [10] using pre-defined search queries to get near real-time public statuses. Pre-defined search queries are, for example, hash tags users often use to annotate breaking news e.g. #breakingnews and "breaking news" keyword.
2) *Indexing*: To accommodate the process of grouping similar messages, an index based on the content of messages is constructed. In this experiment, we use Apache Lucene [11].
3) *Grouping*: Messages that are similar to each other are grouped together to form a news story. Similarity between messages is compared using TF-IDF [12], [13] with term score boosting for named entities.
4) *Community Detection*: To further explore the similarity of contents between message groups, an analysis on the community structure is performed. The details for this task is explained in section VII.

B. Message-level similarity

The similarity between two messages in the grouping task is defined as:

$$\text{sim}(m_1, m_2) = \sum_{t \in m_1} [\text{tf}(t, m_2) \times \text{idf}(t) \times \text{boost}(t)] \quad (1)$$

$$\text{tf}(t, m) = \frac{\text{count}(t \text{ in } m)}{\text{size}(m)} \quad (2)$$

$$\text{idf}(t) = 1 + \log \left[\frac{N}{\text{count}(m \text{ has } t)} \right] \quad (3)$$

boost(t) is raised for name entities e.g. China, England, Eurostar, Haiti and Twitters artifacts like hash tags and usernames to improve the score on identifiable keywords. We use the Stanford Named Entity Recognizer (NER) [14] for the classification of named entities. NER provides a general implementation of linear chain Conditional Random Field (CRF) sequence models, coupled with well-engineered feature extractors for Named Entity Recognition. In our experiment it gives an accuracy of 87% using the the CRF classifier.

C. Group Assignment

The algorithm for a message-group assignment is shown below.

Algorithm 1 Assign message m into a group in G

```
for g in G do
    Score[g] ← Sim(m, g.firstDoc, g.topTerms)
end for
if Max(Score) > MergeThreashold then
    Assign(m, Max(Score).groupId)
else
    groupId ← Group.create()
    Assign(m, groupId)
end if
return G
```

To ensure that messages in the groups are related to the first story and to allow further messages to develop upon previous messages, we will compare a message with the first message in a group and the top k terms in that group. In this experiment, we set k to 10. This is the average number of words in a Twitter message. If we do not place a limit to the number of terms to compare, it is possible that the topic of the message group may drift to irrelevant topics away from it's original content. We then assign a message to a group if the score exceeds a pre-defined threshold called $MergeThreshold$. The results to this point are groups of messages or news stories.

D. Group Ranking

The score for each group is computed as follows:

$$S = w_1 \sum_{u \in g_i} No.Follower(u_i) + w_2 No.Retweet(g_i) \quad (4)$$

$$Score(g_i) = \frac{1}{Z} \sum_{n=1}^{l} \left[\frac{S}{\log(\Delta_n + 2)} \right] \quad (5)$$

$$\Delta_n = t_{current} - t_n \quad (6)$$

A raw group score S in (4) is based on reliability and popularity factors. Reliability is determined from the numbers of followers from all the users who posted messages in the group. Popularity is determined from the numbers of retweet within the group. The final group score in (5), is adjusted based on the freshness of messages. To this effect a group that has newer messages receives a higher score than a group with older messages. The computation is done on l last messages in a group. Δ_n in (6) is the difference between the current time and the time where a message is created. Z is the normalizing factor.

E. Story development stage

In the subsequent stage, each news story is adjusted with appropriate ranking through a period of time. In addition new findings from external source (outside of Twitter) such as news articles of reliable news source or media for examples, photos and video footage can be aggregated to existing news stories.

IV. Effects of Term Score Boosting for Named Entities

We show the result when similar news containing messages are grouped together. The sample messages have been collected in February 2010. Each message is given number and group label. For the purpose of demonstration, we selected 10 messages $m0_{(label)} - m9_{(label)}$ and give 5 labels as follows: (1) for Toyota's brake problems. (2) for Michael Jackson's Doctor. (3) for heavy snow storm in the U.S. (4) for U.S. military base issues in Okinawa, Japan. (5) for escaped prisoners in Haiti. In this experiment, we show that raising the importance of named entities can improve the grouping result.

We employ the grouping method described in section III. Table I shows results from 4 configurations: No boost for named entities, with boost for name entities by raising the term score by the factor of 1.5, 1.7 and 2 respectively.

TABLE I.
Grouping results based on different term score boosting values

(a) No boost NMI = 0.929	
g0	$m3_{(2)}$ $m4_{(2)}$ $m5_{(2)}$
g1	$m7_{(4)}$ $m8_{(4)}$
g2	$m0_{(1)}$ $m1_{(1)}$
g3	$m2_{(2)}$
g4	$m6_{(3)}$
g5	$m9_{(5)}$

(b) boost$(t) = 1.5$ NMI = 1.0	
g0	$m2_{(2)}$ $m3_{(2)}$ $m4_{(2)}$ $m5_{(2)}$
g1	$m7_{(4)}$ $m8_{(4)}$
g2	$m0_{(1)}$ $m1_{(1)}$
g3	$m6_{(3)}$
g4	$m9_{(5)}$

(c) boost$(t) = 1.7$ NMI = 0.896	
g0	$m0_{(1)}$ $m1_{(1)}$ $m7_{(4)}$ $m8_{(4)}$
g1	$m2_{(2)}$ $m3_{(2)}$ $m4_{(2)}$ $m5_{(2)}$
g2	$m6_{(3)}$
g3	$m9_{(5)}$

(d) boost$(t) = 2$ NMI = 0.782	
g0	$m2_{(2)}$ $m3_{(2)}$ $m4_{(2)}$ $m5_{(2)}$ $m9_{(5)}$
g1	$m0_{(1)}$ $m1_{(1)}$ $m7_{(4)}$ $m8_{(4)}$
g2	$m6_{(3)}$

It is necessary to raise an importance of named entities because the length of messages in Twitter is short, and may not have enough information for comparison. For example a web page usually contains a larger set of terms when compared with Twitter messages. The average term size after the removal of stop words for messages in Twitter according to our data set is 7. If we do not place the importance on particular terms then we cannot find related messages that have weak similarity based on a traditional TF-IDF scheme. One such example is shown in table I(a). $m2$ does not have a similarity score high enough to be grouped with $m3$, $m4$ and $m5$. However if we raise the score for name entities grouping results can be adjusted. By raising boost(t) to 1.5, we can achieve the correct grouping result. Table I(c) and (d) show results when the grouping result based on name entities is too sensitive. That is the case when named entities have a large impact over the grouping result. It is important to choose the appropriate value for boost(t), as it defines how sensitive the grouping is. In this experiment we choose the value that maximizes the normalized mutual information (NMI). NMI is explained in section VIII.

V. Message Groups - Named Entities Network

From the grouping result in section III, we can construct a message groups - named entities network (GE network) by linking each message groups to the named entities it refers to. In this network there are two types of vertices: message groups (*Gr*) and named entities (*En*). An example of such network is shown in Figure 4. We can see that each of *Gr* refers to one-to-many of *En*. From this network we want to find a community of *Gr* through the correspondence with *En*. The assumption is that two or more message groups which refer to common named entities have similarity with each other.

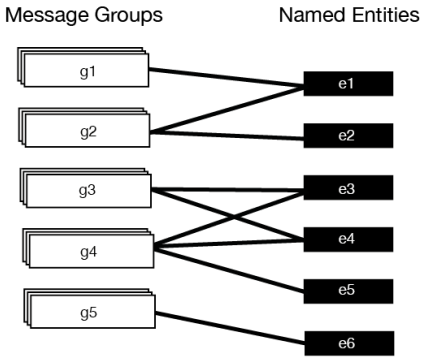

Figure 4. Message groups-named entities network

A. Message Groups

A messages group is a collection of similar messages put together using TF-IDF with named entity term boosting. Therefore we have $g_i = \{m_{i1}, m_{i2}, ...m_{in_i}\}$.

Messages m contain within a message group g_i are strictly similar to each other in terms of vocabulary. By using Twitter as a dataset, the grouping method can mostly detect retweeted messages or messages with similar word arrangements.

B. Named Entities

Named entities are special terms classified by NER into predefined categories such as Person, Organization, Location and Miscellaneous. The assumption is that named entities are important elements in a short-length message as they give clues to identifiable facts. There exist an edge from a message group to a named entity vertex when such message group refers to such named entity.

VI. Dataset

The GE network dataset is collected from Twitter on July 21st, 2010 for the purpose of this study. The data is processed based the method proposed in section II. The properties of the GE network are shown below.

TABLE II.
NETWORK PROPERTIES

Network type	undirected
Numbers of vertices (*Gr*, *En*)	200, 218
Total numbers of edges	1109
Mean degree	5.639
Number of connected components	49
Fraction of vertices in the largest component	0.629

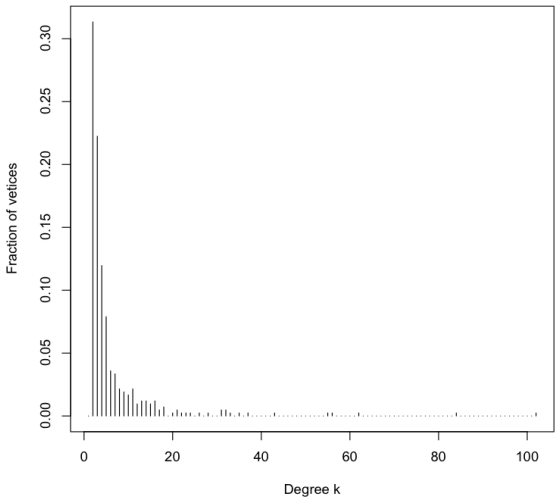

Figure 5. The degree distribution of the GE network

Figure 5 shows the histograms of the distributions of the vertices in the network. We can see that most of the vertices in the network have low degree. Vertices with degree 1,2 and 3 are approximately accounted for 31%, 12% and 7% respectively. The average mean degree of message groups is 5.09. This means that on average, each messages group refers to around 5 named entities.

Figure 6 shows a partial network structure of the dataset with 45 message groups (circled nodes) and 24 named entities (squared nodes). There are two leading news stories: First, the news stories related to BP, a UK-based global oil and gas company which refer to named entities, for examples, BP, Tony Hayward (BP's CEO at that time), London, Greenpeace, etc. Second, the news stories related to the Australian election which refer to named

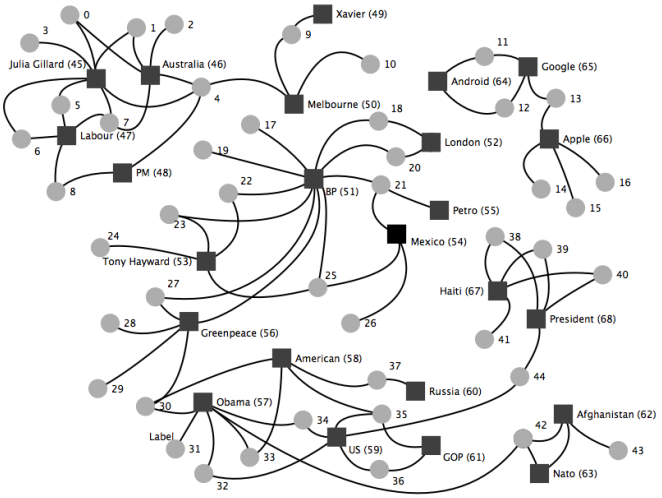

Figure 6. A partial network structure of our dataset

entities, for examples, Julia Gillard (27th Prime Minister of Australia), Labor, Melbourne, etc.

VII. Detecting Similarity using Community Detection Methods

In this section, we apply community detection methods to the GE network described in previous section. The purpose is to demonstrate that named entities can be used to determine clusters of similar message groups and to evaluate each community detection method for the appropriateness of our approach. In this experiment we choose four community detection methods, edge-betweenness algorithm, fast greedy algorithm, label propagation algorithm and walktrap algorithm. We use *R* [15], a software environment for statistical computing and graphics with *igraph* [16], a software library for creating, manipulating graphs and network analysis.

Edge betweenness

This algorithm was developed by Girvan and Newman [17]. Instead of trying to construct a measure to determine the edge centrality with respect to communities, it focuses on edges that are least central, or the edges that are most "between" communities. The algorithm works in the following steps:

1) Calculate the betweenness using Newman's Fast algorithm [18] for all edges in the network.
2) Remove the edge with the highest betweenness.
3) Recalculate betweennesses for all edges affected by the removal.
4) Repeat from step 2 until no edges remain.

Fast greedy

This algorithm was developed by Clauset *et al.* [19] It tries to find dense subgraph, also called communities in graphs by directly optimizing a modularity score. Let A_{ij} be an element of the adjacency matrix of the network

where k_i is the degree of vertex i. Then, the modularity, denoted by Q is given by

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where c_i is the group to which vertex i belongs and $\delta(c_i, c_j) = 1$ if $c_i = c_j$, otherwise 0.

The advantage of this algorithm is that it has a linear running time of $O(n \log^2 n)$ whereas the naive implementation of edge betweenness algorithm runs at $O(n^2)$ on a sparse graph.

Label propagation

This algorithm was developed by Raghavan *et al.* [20] This is a near linear time algorithm where each iteration of execution takes a linear time of $O(m)$ where m is the number of edges. Initially, each node is labeled with a unique label. Then, it iteratively updates the labels by majority voting in the neighborhood of the vertex. The process continues until the condition of labels is stable.

Walktrap algorithm

This algorithm was developed by Pons and Latapy [21]. This hierarchical agglomerative algorithm tries to find densely connected subgraphs in a graph by using a random walk process. This algorithm runs at $O(n^2 \log n)$ in most cases.

VIII. Evaluation and Results

A. Evaluation methods

We use *purity* and *normalized mutual information* (*NMI*) [12] to evaluate the community division results. *Purity* is computed by summing the number of items of a majority class in each cluster.

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j|$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the set of clusters and $\mathbb{C} = \{c_1, c_2, ..., c_j\}$ is the set of classes.

The range of purity is between 0 and 1. A perfect clustering has a purity of 1 whereas bad clusterings have purity values close to 0.

NMI is computed as follows:

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega; \mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2}$$

where $I(\Omega; \mathbb{C})$ measures the mutual information and H is the entropy function:

$$I(\Omega; \mathbb{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k) P(c_j)}$$

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k)$$

The range of NMI is between 0 and 1. If the community divisions are approximately the same for Ω and $\mathbb{C}$, then NMI has a value closes 1. If the two have or no relationships, then NMI becomes 0.

B. Community detection evaluation

We evaluate four methods of community detection: edge betweenness (EB), fast greedy (FG), label propagation (LP) and Walktrap (WT) algorithms conducted on our GE network. For each method, we select 105 message groups or roughly 25% of all the nodes in our network for evaluation. The results are evaluated based on the standard classes which are judged by human. The metrics for evaluation are modularity, community size, purity and NMI.

TABLE III.
Performance and Statistics

Metrics	EB	FG	LP	WT
Modularity	0.854	0.850	0.817	0.823
No. of Community	68	59	99	113
Purity	0.67	0.625	0.707	0.78
NMI	0.34	0.312	0.401	0.567

From Table III, we can see a comparison of the four community detection methods. In our dataset, the Walktrap algorithm gives the highest purity and NMI values. For this size of network (as described in Table II), where, the number of connected components is 49 and the fraction of vertices in the largest component is 0.629, it is rather not a dense graph. Therefore a community detection method which approximate a large number of community is likely to work well with this network since there exist disconnected components and these components are themselves form communities. In real life, there are some news stories, usually one or two leading stories in Twitter, which are widely discussed within a certain period of time. These popular news stories constitute the portion of the network where connections between similar vertices are denser than the rest of the network. There are also news which are not so popular in a global scale or not related to the leading stories. These constitute the disconnected components in the network. In the event where a breaking news which is highly popular and widely discussed happens the appearance of the network changes compared to the event where not so many popular news stories appeared in the timeline.

C. Community visualization

Figure 7 and 8 show the finding of communities. We use *igraph* as a tool for visualization. For simplicity, we visualize the partial structure of the network as discussed in section VI. In this network there are 8 leading news stories: (1) BP, (2) Australian election, (3) the U.S. President, (4) smartphone, (5) NATO, (6) Greenpeace, (7) Haiti president, (8) Australia's school. Circled nodes represent message groups whereas squared nodes represent named entities.

IX. Discussions

We see the possibility of using named entities to define the similarity between groups of messages using the community detection methods. Message groups which

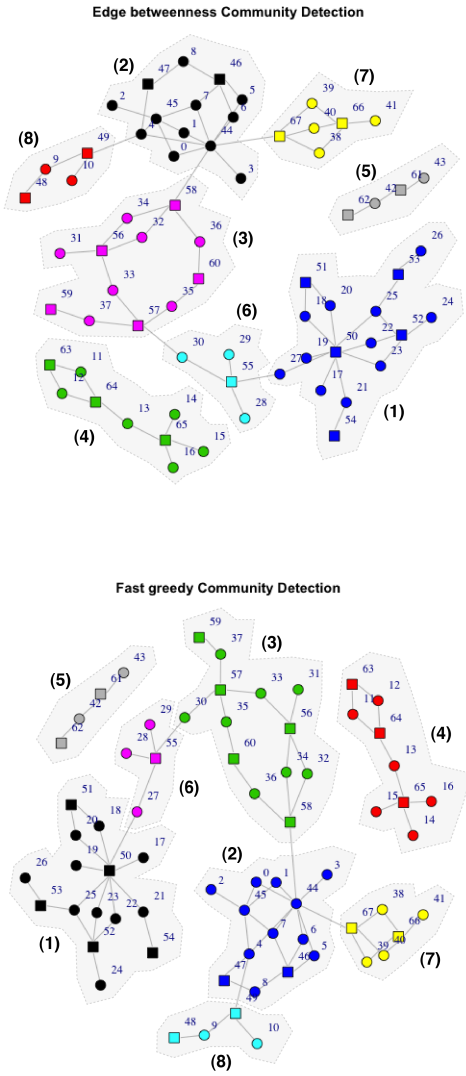

Figure 7. Memberships of vertices in the GE network based on edge betweenness and fast greedy algorithms

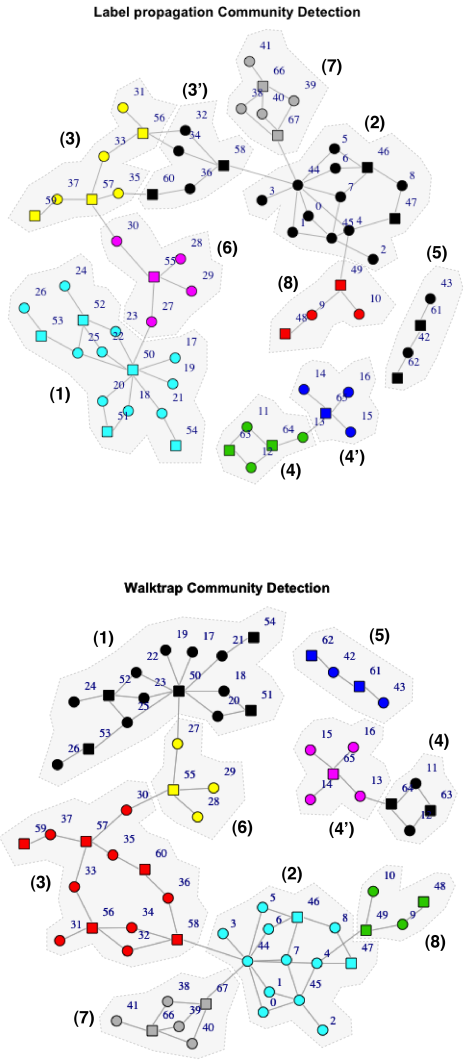

Figure 8. Memberships of vertices in the GE network based on label propagation and walktrap algorithms

refer to many of the same named entities are more densely connected than the others. By applying the community detection we can divide our network into communities of message groups having dense connections within communities. The benefit from this approach is that we can explore related message groups. In the application of breaking news detection from Twitter, it can help us explore stories in a broader sense. The limitation is that this approach makes sense only within a certain scope of time. That is a named entity appeared in news mostly refers to one exact concept at a certain point of time. In addition, named entities that are general and commonly referred in many contexts may pull all message groups together in one group, thus reducing the purity metric.

There is a challenge for natural language processing tasks in microblogs because the style of texts are often considered unconventional [6]. That is a sentence may not be grammatically correct, contain spelling mistakes (either intended or not intended), and slangs. While style of writing may be different according to geographic regions, person and time, however named entities are often preserved in a uniform manner.

We can extend beyond the use of named entities to the use of significant entities. A significant entity is any entity that we considered important in a message. It can be named entities, tags, urls, application-specific terminologies, and etc. The idea is that messages group that refer to these entities may relate to each other.

In terms of applications, this approach can be used to find similar news story from microblogs. We prototyped a web application based on the method described in section III called Hotstream. At the present stage, the implementation for similarity detection is at the grouping level. The purpose is to construct a real-time news portal featuring breaking news and popular stories from Twitter. Figure 9 (a) shows the front page of Hotstream accessed

on February 5th, 2010. The page contains the list of top stories within 24 hours. The ranking for top stories is based on the method described in section III-D.

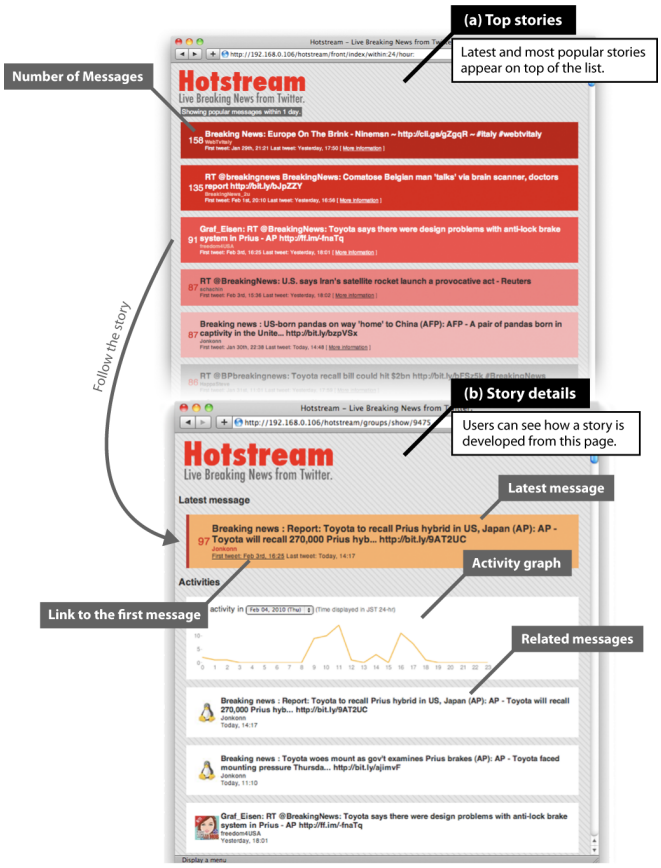

Figure 9. Hotstream, "Live breaking news from Twitter", application screenshots. Top (a) front page showing top stories, bottom (b) story details.

Users can click on each story to find more details. Figure 9 (b) shows a timeline of a story. We show the number of messages along with the first message in the story. The top message is the latest message in the story. In addition, the activity graph which is based on the number of messages in the story with respect to time is displayed.

In the case when we want to find strictly similar messages, for example, retweeted messages in Twitter, we may use the message groups. While in the case where we want to explore similar groups of messages based on named entities, we can find clusters based on the community structure.

X. Conclusion

In this approach, we define two levels of similarity for microblogs. In the first level, we find message groups. A message group is a collection of messages bundled together based on TF-IDF with term score boosting for named entities. The similarity at the first level is strictly based on terms within a message. In the second level, we find communities of the message groups along with named entities to further discover related messages from a different group. At this level, the similarity is governed by named entities. We show that this approach is suitable for a microblogging service like Twitter in a breaking news detection and tracking application.

References

[1] E. Morozov, "Iran Elections: A twitter Revolution?" *The Washington Post*, June 17, 2009, http://www.washingtonpost.com/wp-dyn/content/discussion/2009/06/17/DI2009061702232.html.

[2] D. Gross, "Twitter claims 105 million registered users," http://scitech.blogs.cnn.com/2010/04/14/twitter-claims-105-million-registered-users/, Accessed August 1, 2010.

[3] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *WWW '10: Proceedings of the 19th International Conference on World Wide Web*. New York, NY, USA: ACM, 2010, pp. 591–600.

[4] A. Dong, R. Zhang, P. Kolari, J. Bai, F. Diaz, Y. Chang, Z. Zheng, and H. Zha, "Time is of the essence: improving recency ranking using twitter data," in *WWW '10: Proceedings of the 19th International Conference on World Wide Web*. New York, NY, USA: ACM, 2010, pp. 331–340.

[5] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *WWW '10: Proceedings of the 19th International Conference on World Wide Web*. New York, NY, USA: ACM, 2010, pp. 851–860.

[6] S. D. D. Ramage and D. Liebling, "Characterizing microblogs with topic models," in *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2010.

[7] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*. New York, NY, USA: ACM, 2010, pp. 261–270.

[8] R. Mateosian, "Micro Review: Twitter," *IEEE Micro*, vol. 29, Issue 4, pp. 87–88, July-August 2009.

[9] J. Allen, *Topic Detection and Tracking*. Norwell, Massachusetts: Kluwer Academic, 2002, pp. 17–30.

[10] "Twitter Streaming API," http://apiwiki.twitter.com/Streaming-API-Documentation, Accessed February 1, 2010.

[11] "Apache Lucene," http://lucene.apache.org, accessed February 1, 2010.

[12] C. D. Manning, P. Raghavan, and H. Schütze, *Introdution to Information Retrieval*. New York: Cambridge University Press, 2008, pp. 108–115, 356–358.

[13] "Similarity (Lucene 3.0.0 API)," http://lucene.apache.org/java/3_0_0/api/all/org/apache/lucene/search/Similarity.html, accessed February 1, 2010.

[14] J. Finkel, T. Grenager, and C. Manning, "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling," in *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, 2005, pp. 363–370.

[15] "The R Project for Statistical Computing," http://www.r-project.org/.

[16] "The igraph library for complex network research," http://igraph.sourceforge.net.

[17] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review*, vol. E 69, no. 026113, 2004.

[18] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review*, vol. E 69, no. 066133, 2004.

[19] C. M. Aaron Clauset, M. E. J. Newman, "Finding community structure in very large networks," *Physical Review*, vol. E 70, no. 066111, 2004.
[20] R. A. Usha Nandini Raghavan and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review*, vol. E 76, no. 03610, 2007.
[21] M. L. Pascal Pons, "Computing communities in large networks using random walks," *Journal of Graph Algorithms and Applications*, vol. 10, no. 2, pp. 191–218, 2008.

Swit Phuvipadawat is a master degree student in the Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology. He received his B.Sc. in Information Technology from Sirindhorn International Institute of Technology, Thailand in 2008. At Tokyo Institute of Technology, he conducts research on Twitter, topic detection and tracking and social network analysis.

Tsuyoshi Murata, Dr. is an associate professor in the Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology. He obtained his doctors degree in Computer Science at Tokyo Institute of Technology in 1997, on the topic of Machine Discovery of Geometrical Theorems. At Tokyo Institute of Technology, he conducts research on Web mining, artificial intelligence, and social network analysis.