Topic Detection from Large Scale of Microblog Stream with High Utility Pattern Clustering

Jiajia Huang
Computer School
Wuhan University
Wuhan, 430072, China
huangjj@whu.edu.cn

Min Peng
Computer School
Wuhan University
Wuhan, 430072, China
pengm@whu.edu.cn

Hua Wang
Centre for Applied Informatics
Victoria University
Melbourne, Australia
hua.wang@vu.edu.au

ABSTRACT

With the popularity of social media, detecting topics from microblog streams have become an increasingly important task. However, it's a challenge due to microblog streams have the characteristics of high-dimension, short and noisy content, fast changing, huge volume and so on. In this paper, we propose a high utility pattern clustering (HUPC) framework over the microblog streams. This framework first extracts a group of representative patterns from the microblog streams, and then groups these patterns into topic clusters. This approach works well on large scale of microblog streams because it clusters the patterns that perform better in describing topics, rather than clustering noises and short microblogs directly. Furthermore, the proposed framework can detect coherent topics and new emerging topics simultaneously. Extensive experimental results on Twitter streams and Sina Weibo streams show that the developed method achieves better performance than other existing topic detection methods, leading to a desirable solution of detecting event from microblog streams.

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: General

Keywords

Topic Detection, High Utility Pattern, Clustering

1. INTRODUCTION

It is true that microblog platform has developed into one of the most popular social media that enables individuals, corporations and government organizations to stay informed of "what is happening now". As a result, the latest news often occurs on the microblog before appearing in traditional media outlets. It can yield unprecedentedly valuable information by monitoring and analyzing the rich and continuous flow of microblog streams, which, however, would not have been available from traditional media outlets [7]. Topic/Event detection is one of the most popular monitoring manners that has drawn huge attention from researchers in information sciences, policy and decision makers in governments and enterprises. Although various methods of event detection from conventional media sources have been addressed in Topic Detection and Tracking research program [9][12], they do not work well on microblog streams due to the big difference between microblog texts and formal texts (e.g., news articles, blogs, academic papers). We list the differences as follows:

- Microblog texts have the characteristics of short length, large number of spelling and grammatical errors and frequent use of informal and mixed language [7].
- Microblog texts are usually relevant to a wide variety of real-world events. Some of them might contain interesting and useful information, whereas others might provide little value.
- Microblog texts have the properties of very fast data arriving rate, unbounded size of data and inability to backtrack over previously arriving transactions.

Studying of topic detection on microblog (or streams) has become an active research area. Most researchers utilize probability topic models to mine topics over static Twitter feeds [4][15][23]. Furthermore, there are also some studies on detecting emerging topics from Twitter streams [16][19][20][22]. These studies discover topics either based on text clustering, or based on bursty keywords clustering. However, due to the characteristics of microblog texts discussed above, it is difficult to measure the similarity between a pair of microblog texts or to identify valuable keywords from a large scale of vocabulary effectively.

Although microblog texts are usually short and full of noises, there are still many co-occurrence terms that can perform well in representing a topic. For example, a topic about `presidential election` can be inferred from a frequent pattern [`Barack Obama`, `America`, `elect`, `win`]. Compared with a single term, a frequent pattern is more informative in expressing a topic. In fact, frequent patterns have been used in document clustering in previous studies [8][24]. In these studies, frequent patterns are mined from documents and then gathered into clusters. However, due to the *downward closure* property of the frequent pattern mining, there are many highly similar patterns in the generated result. Therefore, it is information redundancy and time-consuming if all of the mined patterns are used for clustering.

In this paper, we try to detect topics from microblog stream by proposing a high utility pattern clustering (HUPC)

PIKM'15, October 19, 2015, Melbourne, Australia.

DOI: http://dx.doi.org/10.1145/2809890.2809894.

framework. In this framework, we focus on tackling two problems: (1) how to mine representative patterns from microblog streams, and (2) how to effectively clustering these patterns to form meaningful topics.

To reduce the number of patterns used in pattern clustering process, we only mine a small number of representative patterns such that the entire streams can be covered as much as possible while the redundancies among the patterns are as less as possible. To this end, we apply a *utility* to evaluate the importance of a pattern. A pattern is identified as a *high utility pattern* (HUP) if it contains many important terms and appears in many texts. Otherwise, it's a low utility pattern. Furthermore, to reduce the redundant information among the mined HUPs, we apply an *overlap-degree* to evaluate the redundancy between a pair of patterns. If two patterns have a high overlap-degree, the one with the lower utility should be discarded.

Essentially, HUPs are short texts, which lack enough information to effectively measure the similarity between pair of them in pattern clustering process. But the semantic information of a pattern (or term) can be extended from external knowledge, such as Wikipedia [14] or WordNet [5] or from internal knowledge. However, it is hard to extend a emerging term that has not been covered by the external knowledge. In this paper, we use internal knowledge, i.e., pattern associated texts, to extend pattern's semantic information. Thus, the similarity between a pair of patterns can be measured by the similarity between their associated texts.

In summary, the HUPC framework works as follows.

Given a text stream collected from a microblog platform, the framework first mines a group of HUPs from the stream via a high utility pattern mining algorithm. This algorithm addresses the first problem by selecting a small number of patterns from all of the generated frequent patterns as HUPs, causing the pattern clustering process to be sped up significantly.

Then, the framework groups these patterns into different topics through designing a novel incremental pattern clustering processing, which uses associated texts that contain the pattern terms to compute the similarity between a pair of HUPs. In addition, this process is a combination of kNN classification and modularity-based clustering [17], which can identify new emerging topics and coherent topic simultaneously. Furthermore, the incremental clustering process is executed without specifying cluster number. It is a significant advantage for detecting topics from unstable microblog streams because users usually don't clear about how many topics are there in a large scale of stream before clustering.

Finally, a group of topical words are extracted from each pattern cluster to represent this topic in a more succinct manner.

The main contributions of this paper are summarized as follows:

First, we propose a HUPC framework, which tackles the topic detection problem as a pattern mining and clustering process. To the best of our knowledge, it is the first attempt to apply the frequent pattern clustering into topic detection from microblog streams. The designed pattern mining algorithm and incremental pattern clustering algorithm can achieve better performance in detecting meaningful topic from microblog streams.

Second, the incremental pattern clustering makes the proposed HUPC framework detect emerging topic and track the evolutionary trending of coherent topics at the same time.

Third, the HUPC framework is evaluated on both English texts stream and Chinese texts stream, which are collected from two popular microblog websites. We also employ some state-of-the-art methods for comparisons. Experimental results show that the framework achieves better and stable performance on different automatic evaluation metrics. We also present a case study on detected topic examples to illustrate the performance of the framework in identifying emerging topic from coherent ones.

The rest of the paper is organized as follows. Section 2 reviews the related work of topic detection. Section 3 describes the HUPC framework in detail. Section 4 presents datasets and experimental results. We conclude the paper in Section 5.

2. RELATED WORK

Methodologically, general-purposed topic detection methods largely fall in three classes: document-pivot method, probability topic model, and feature-pivot method. In the following, we present a short review of these three classes of topic detection approaches.

2.1 Document-pivot method

Document-pivot method groups together individual documents according to their similarity and represents a topic as sets of relevant documents. Incremental clustering is used in this type of approach, aiming at detecting emerging (or novel) topics by comparing each incoming text with all existing clusters [19][20][25]. The main difference among these methods is that the documents similarity is measured in different ways. For instance, Phuvipadawat et al. proposed a method to track breaking news, where an incoming tweet is clustered into a group by comparing its TF-IDF vector with the TF-IDF vector of the first tweet and the TF-IDF vector of the most common terms in this group [20]. Petrovic et.al. proposed a modification of Locality Sensitive Hashing (LSH) to retrieve the best match cluster for each incoming document [19] in a fast way. Zhou et.al. proposed a graphical model, called location-time constrained topic (LTT), to capture the content, time and location of social messages. Then, the similarity between two messages is measured by the distance between their distributions [25].

2.2 Probability topic model

Probability topic model treats the problem of topic detection as a probabilistic inference problem. It represents a topic as a distribution over the terms, and a document as a distribution over the topics. This model is widely used for mining topics from documents in recently years and has been improved from various aspects when applied on Twitter, including addressing the topic sparsity [15] and inferring the number of clusters automatically [23]. In addition, there are other studies on combining traditional topic models with other methods. For example, frequent pattern mining was used in conjunction with probabilistic topic model to enrich the representation of documents before they are processed by standard probabilistic topic models [13]. Knowledge of *must-link* and *cannot-link* was used in topic model inference to generate more coherent topics [4]. Burstiness features were combined with traditional topic model in a

Figure 1: HUPC Framework Overview

non-parametric model to identify both events and topics simultaneously [6]. However, these studies mainly focus on detecting topics from static texts, rather than from streaming texts.

2.3 Feature-pivot method

Feature-pivot method groups together terms to form topics, which mainly consists of two steps. First, a list of features (i.e., terms or phrases) are selected based on their frequency or burstiness, and then, some form of inter-feature similarity is used for clustering. These methods can typically be classified into two categories according to the feature granularities.

The first category focuses on selecting a group of bursty terms for topic detection. For example, TwitterMonitor [16] first identified bursty keywords that have a higher absolute frequency than usual and then used them as seeds to explore more bursty terms. EDCoW [22] applied a wavelet analysis on terms to model their frequencies to select bursty terms, and then clustered these terms based on a community detection method. The second category focuses on selecting co-occurrence features for topic detection, such as frequent patterns, text segments (i.e., n-grams), and *memes*. Besides being used in traditional text clustering [8][24], frequent patterns are also used for detecting topics from Twitter. Petkos et al. treated the topic detection problem as a frequent pattern mining process and proposed a soft frequent pattern mining algorithm, which represents a topic as a soft frequent pattern [18]. Guo et al. devised a frequent pattern stream mining method to detect hot topics from Twitter streams [10]. Besides frequent patterns, text segments are also used for detecting emerging events from tweets [14].

3. TOPIC DETECTION FROM MICROBLOG STREAM

In this section, we first formulate the problem of topic detection from microblog stream and then introduce the HUPC framework.

Set a microblog stream D as the texts generated in a stream up to a given timestamp L. Set all the latest texts arriving in a fixed time interval (e.g., one hour) as a current text batch D_L, and all the texts arriving the previous time interval as a previous text batch D_{L-1}. Each text d_i in a batch $D_l(l = L, or L-1)$ is represented as *a-bag-of-words* after stopwords removed. We model the text batch D_L as a mixture of multiple topics $T_L = \{T_1, T_2, .., T_{L^N}\}$, where $T_i(i = 1, 2, ..., L^N)$ denotes the topic detected from D_L. This topic is either a coherent topic appeared in the previous batch D_{L-1}, or a new emerging topic. Therefore, in this paper, our topic detection task is to detect topics $\{T_L\}$ from the latest text batch D_L and to identify them as emerging or coherent topics.

The HUPC framework consists of three components: *top-K HUP mining*, *HUP clustering* and *post-processing*, as shown in Figure 1. After receiving a text batch, such as all texts of one hour, the top-K HUP mining component digs out the top-K most representative HUPs. Then the HUP clustering component detects topics from the batch through patterns clustering and identifies the types (i.e., emerging or coherent) of these topics at the same time. At last, in the post-processing component, topical words are selected from each pattern cluster to express the topic in a more succinct way. In the rest of this section, we will describe each component in detail following the order of their usage in our framework.

3.1 High Utility Pattern Mining

High utility pattern mining is a primary and important component of the framework, because the quality of the detected topics depends on the quality of the mined patterns. In this part, we first give relevant definitions of high utility pattern and then present the top-K high utility pattern mining (top-K HUP mining) algorithm.

Let $D_L = \{d_1, d_2, ..., d_{L^m}\}$ be the latest text batch (i.e., transaction set) that contains L^m texts and $I = \{i_1, i_2, ..., i_{L^n}\}$ be a set of terms (i.e., items) in this batch.

Definition 1.The utility of a pattern p in a text d_j is denoted as $u(p, d_j)$ and is defined as $\Sigma_{i \in p \wedge p \in d_j} w_i$, where w_i is the utility of term i in this batch [1].

Definition 2.The utility of a pattern p in the text batch D_L is denoted as $u(p)$ and is defined as $\Sigma_{p \in d_j \wedge d_j \in D_L} u(p, d_j)$.

Although the high utility pattern mining based on above definitions does not have the downward closure property, it still may generate a lot of patterns that share many common terms (i.e., [`Barack Obama`, `people`, `guess`, `elect`, `win`], and [`people`, `Obama`, `guess`, `elect`]). These similar patterns, especially long ones, usually express similar semantic information, which in fact is a kind of information redundancy for topic detection. In this paper, we hope to reduce this redundancy as much as possible. To this end, we measure the redundancy between two patterns by defining an *overlap-degree* metric as follows:

Definition 3. The utility of the overlap pattern $p \cap q$ between pattern p and pattern q in D_L is denoted as $u(p \cap q)$ and is defined as $\Sigma_{p \cap q \in d_j \wedge d_j \in D_L} u(p \cap q, d_j)$.

Definition 4. The overlap-degree between pattern p and pattern q in D_L is denoted as $o(p \cap q)$ and is defined as $u(p \cap q)/(u(p) + u(q) - u(p \cap q))$.

Thus, the problem of HUP mining can be stated formally as follows:

Problem 1. HUP mining is to find out a group of patterns P from batch D_L such that the sum of their utilities is maximized and the overlap-degree between each pair of patterns is no more than a given threshold. The problem thus can be formulated as an optimization problem with following objective function:

$$\arg\max_{p \in P} \sum U(p), \qquad s.t., \qquad \forall p, q \in P, o(p, q) < \delta \qquad (1)$$

[1]In this paper, the utility of an item is set to the frequency of it appearing in a text batch.

where δ is the maximum overlap-degree threshold between a pair of HUPs.

Based on above definitions, it is easy to find out a group of HUPs by executing a greedy algorithm with following two steps: (1) Sort all the patterns generated by Apriori [1] or FP-Growth [11] with utilities in descending order. (2) Select HUPs by scanning each pattern and judge whether its overlap-degrees with all the patterns in the HUP set are lower than δ. Add it into the HUP set if it does.

However, there are two challenges in mining HUPs from text stream based on above steps: (1) It is difficult to set a proper minimum support threshold for a microblog stream due to the dynamic change of the stream. (2) It is memory-consuming in storing all the patterns in step (1), because it may dig out millions and even more patterns when the minimum support is set too small.

We design a top-K HUP miming algorithm to meet the above challenges. This algorithm speeds up the HUP mining process by executing the HUP selection along with the pattern generation. Furthermore, as it is easier and more intuitive for users to indicate how many patterns they would like to see than specify a minimum support threshold, the algorithm turns to mining top-K HUPs to satisfy formula (1).

The entire process is given in Algorithm 1. Here, we maintain a global HUP set P and a corresponding pattern utility set PU. Line 1 runs the FP-Growth to generate candidate pattern q one by one, where the minimum support is set to 0 or a very small value. Then, it compares pattern q with each pattern p in the HUP set P (line 2-21) until there are K patterns in P. In the comparison process, pattern q is added into P (line 15-17) if it is judged as a HUP (line 3-14). At last, remove all the redundant patterns in *delete_pattern_set* from P and their corresponding pattern utilities from PU (line 18-19).

In this algorithm, the runtime is majorly used for calculating each candidate pattern's utility. However, the computation time can be reduced by constructing a term-text index, where the pattern utility can be calculated by indexing its relevant terms from the index. Note that the algorithm needs to scan the entire transaction set in FP-Growth process, whose time complexity is $O(L^m)$, where L^m denotes the total number of texts in D_L. Thus, the time complexity of the algorithm is $O(K+L^m)$, where K denotes the number of mined pattern. In fact, although there may ten thousands of texts in a batch, hundreds of patterns containing a few thousands of terms are enough to represent the entire batch.

Algorithm 1: Top-K HUP Mining

Data: Text batch D_L; Maximum number of patterns K; Maximum overlap-degree threshold δ.
Result: High utility pattern set P.
Initialize: $P \leftarrow \varnothing$, $PU \leftarrow \varnothing$, $delete\_pattern\_set \leftarrow \varnothing$;

```
1  for pattern q generated by FP-Growth do
2      if |P| ≤ K then
3          Calculate the pattern utility u(q) by Def. 2;
4          Sort the pattern in P with utility in descending order;
5          Count = 0;
6          for each pattern p in P do
7              Calculate overlap-degree o(p, q) by Def. 4;
               if o(p, q) > δ then
8                  if u(q) > u(p) then
9                      Add p into delete_pattern_set;
10                     Count+ = 1;
11                 else
12                     break;
13             else
14                 Count+ = 1;
15         if Count == |P| then
16             Add pattern q into P;
17             Add pattern utility u(q) into PU;
18         Remove patterns in delete_pattern_set from P
           and its corresponding pattern utilities from PU;
19         Set delete_pattern_set ← ∅;
20     else
21         break;
```

3.2 High Utility Patterns Clustering

Generally, it need to cluster the mined patterns to form more complete topics, because a pattern in the HUP set P usually only expresses a topic to a certain extent. For example, both pattern [`Barack Obama`, `win`] and pattern [`vote`, `president Barack Obama`] refer to a part of information about the topic `Obama win presidential election`. In this part, we devise an incremental pattern clustering method, a combination of kNN classification and modularity-based partition to cluster the HUP set P into groups, where each group corresponds to a possible realistic event. The kNN is used to classify HUPs into existing topics clusters based on its k nearest neighbors (i.e., similar patterns in previous text batch) if they belong to a coherent topic, and the modularity-based partition is used to deduce new emerging topics from the rest of the HUPs.

3.2.1 Pattern similarity

Note that it does not work by measuring the pattern similarity based on the common words they shared because HUP is also short text. In this peper, we measure the similarity between two patterns based on their associated texts in the batch.

Let $P_L = \{p_1, p_2, ..., p_{L^M}\}$ and $P_{L-1} = \{p_1, p_2, ..., p_{(L-1)^M}\}$ be two pattern sets mined from current text batch D_L and previous text batch D_{L-1} respectively. Let $RD_L = \{rd_1, rd_2, ..., rd_{L^M}\}$ and $RD_{L-1} = \{rd_1, rd_2, ..., rd_{(L-1)^M}\}$ be relevant text sets, where each element in each set consists of the associated texts of a pattern. Here, we employ 1) cosine similarity to measure the similarity between a current pattern and a historical pattern and 2) Jaccard coefficient to measure the similarity between two current patterns as follows:

$$sim(p_i, p_j) = 1 - cos(rd_i, rd_j), rd_i \in RD_L, rd_j \in RD_{L-1} \tag{2}$$

$$Jac(p_i, p_j) = \frac{|rd_i \cap rd_j|}{|rd_i \cup rd_j|}, rd_i, rd_j \in RD_L \tag{3}$$

where all texts in set rd_i (or rd_j) are regarded as one document.

Here, Jaccard coefficient is used to measure the similarity between current patterns is because experimental result

shows that this metric is more effective in measuring pattern similarity. But when two patterns come from different text batches and don't share common texts, cosine similarity is used. It has been reported that the cosine similarity outperforms KL divergence, weighted sum, and language models as distance function in the first story detection task [2].

3.2.2 Pattern clustering

As discussed above, a topic deduced from a newly arriving text batch is either an emerging topic that never appeared before or a coherent topic that has appeared in previous batch. Thus, we can utilize the top-k most similar patterns in previous topic clusters to infer the topics from the newly mined patterns, and then deduce emerging topics from the rest of the patterns that cannot be classified into any previous topics.

Let $T_{L-1} = \{T_1, T_2, ..., T_{(L-1)^N}\}$ be the $(L-1)^N$ topics detected from D_{L-1}, where each topic cluster consists of patterns mined from D_{L-1}. For a new pattern $p_i \in P_L$, it may be classified into a previous topic cluster T_l according to its k most similar previous patterns set $N_k(p_i)$ with following formula:

$$f(p_i) = \arg\max_{l} \sum_{p_j \in N_k(p_i)} I(f(p_j) = l) sim(p_i, p_j)$$
$$p_j \in P_{L-1}, l \in \{1, 2, ..., (L-1)^N\}, \cos(p_i, p_j) > \sigma \quad (4)$$

where $f(p_i)$ denotes the topic id of pattern p_i, I is an indicator function. $I = 1$ if $f(p_i) = l$, or $I = 0$ otherwise.

Formula (4) illustrates that a new pattern can be classified into a certain coherent topic cluster if the similarities between it and its k most similar previous patterns are higher than a given threshold σ. Otherwise, this pattern is regarded as an element of an emerging topic cluster, and won't be classified into any previous topics.

Next, we discuss how to detect emerging topics from the unclassified patterns. In this paper, we employ a modularity-based graph partitioning for pattern clustering. Let $G = (V, E, \mathbf{W})$ be an undirected weighted graph, where vertex set V contains all of the unclassified patterns $P_L^* = \{p_1, p_2, ..., p_{L^{M^*}}\}$, and the edge is $E = V \times V$. There is an edge between vertices v_i and v_j if $Jac(p_i, p_j) > \rho$, and then set $w_{ij} = Jac(p_i, p_j)$.

Newman proposed a metric called *modularity* to measure the quality of such graph partitioning as follows [17]:

$$\mathcal{Q} = \frac{1}{4m} \sum_{ij} (w_{ij} - \frac{k_i k_j}{2m}) \delta_{s_i s_j} \quad (5)$$

where k_i and k_j are the degrees of vertices v_i and v_j, and $m = \frac{1}{2}\sum_i k_i$ is the total number of edges in the network. s_i and s_j are the indexes of the subgraph that nodes v_i and v_j belongs to respectively, and $\delta_{s_i s_j}$ is the Kronecker delta. $\delta_{s_i s_j}$ =1 if $s_i = s_j$, or $\delta_{s_i s_j}$ =0 otherwise.

The goal is to partition G such that $\mathcal{Q}$ is maximized, which turns into an optimization problem of maximizing the term involving the leading eigenvalue and completely ignoring all the others [17]. Then, the graph can be divided into two groups according to the signs of corresponding elements in the leading eigenvector. This method is further recursively applied into each of two subgraphs to divide them into smaller ones until no more subgraph can be constructed (i.e., $\mathcal{Q} < 0$).

With such a graph partition, each subgraph corresponds to a topic. A main advantage of employing the modularity-based clustering is that it doesn't need to specify cluster number, which is important for detecting topic from text streams.

3.3 Post-Processing

For the topics $T_L = \{T_1, T_2, ..., T_{L^N}\}$ detected from the text batch D_L, where each one consists of a group of HUPs, this part discusses how to select r representative topical words $W_l = \{w_{l1}, w_{l2}, ..., w_{lr}\}$ to explain each topic T_l $(l = 1, 2, ..., L^N)$. We think a good topical word should have the abilities of (1) good readability, (2) high descriptiveness of the topic and (3) good discrimination performance among different topics. Therefore, we extract topical words from each topic cluster by considering the following factors:

- POS (part-of-speech): each term in a cluster is given a weight based on its POS. For example, *noun* has the highest weight (e.g., 1.0), *verb* comes second (e.g., 0.85), while *adjective* has a lower weight (e.g., 0.5).
- Frequency of a term in a batch: A term's importance is in proportion with the frequency of it appearing in the batch.
- Support of a term in a cluster: A term's importance is in proportion with the number of patterns that contain it in the cluster.
- Distinction among clusters: A term's importance is inversely proportional with the number of different clusters contain it.

Thus, the importance of term w_{li} in cluster T_l can be calculated as follows:

$$score(w_{li}) = w\_pos(w_{li}) \times \log(tf_{li})$$
$$\times sup_{li} \times \log(\frac{|P_L|}{pt} \times \frac{|T_L|}{|Tp|} + 1) \quad (6)$$

where $w\_pos(w_{li})$ denotes the POS weight of word w_{li}, tf_{li} counts of the frequency of w_{li} appearing in the batch D_L, sup_{li} describes the support of w_{li} in the cluster T_l, $|pt|$ is the number of patterns containing word w_{li}, and $|Tp|$ is the number of clusters containing word w_{li}.

Based on the importance scoring of all terms in all topics, we can extract a group of most representative terms to explain each of the detected topics in a brief manner.

4. EXPERIMENTAL RESULT

4.1 Dataset and Experimental Setting

We have created two large scale of datasets to test the performance of the HUPC framework. The first dataset is a tweets collection downloaded from Twitter (in English) (called TwitterSet), consisting of 2,068,721 tweets from a time period spanning October, 2012 to March, 2014 [26]. The second data set is a microblog texts collection downloaded from Sina Weibo (in Chinese) (called SinaSet), consisting of 280,287 texts from a time period spanning January to June of 2014. This dataset is downloaded through the Sina Weibo API by inputting event-oriented keywords. Both datasets are mixing text streams that consists of several event-relevant sub-collections, such as *iPhone5* and *Obama*. The time interval of a text batch is set to one day, because most topic trendings evolving on this interval are more significant than on the intervals of hours. Finally,

we get 38 batches from TwitterSet and 102 batches from SinaSet.

In the pre-processing, first, punctuations, stopwords and URL in each dataset are removed. Then, for TwitterSet, (1) convert letters into lowercase; (2) perform stemming for terms with the WordNet Lemmatizer of NLTK[2]; (3) select the top 3000 frequency terms to represent each tweet in this batch. For the SinaSet, (1) convert text string into terms list through segmentation tool[3], (2) select the top 800 frequency terms to represent each message in this batch.

The dataset is a mixed stream of multi-events, where every text is labeled with an event id. Thus, the topic detection performance can be evaluated from the results of classifying these texts to its relevant topic clusters. To this end, we employ RI (Rand-Index), TPR (true positive rate), FPR (false positive rate)[21] and F-measure [8] as metrics.

We compare the performance of the HUPC framework against online-LDA [3], SFPM [18], and a variant of our method, called MP. SFPM is a type of feature-pivot methods that uses each mined soft frequent patterns to represent a topic. MP is implemented by clustering all the patterns in a batch through modularity-based partitioning.

4.2 Overall Performance Comparison

For Online-LDA, the the topic number is set to 15, which makes it performance best on both datasets. For SFPM, the parameters in *sigmoid* function are set to the same values as reference [18], i.e., $b = 5$, $c = 2$. In addition, to obtain results through this method in a tolerable time, we only select the top 200 terms for soft frequent pattern mining. For both our method and MP, the maximum number of patterns K is set to 500, the maximum overlap-degree threshold δ is set to 0.4. Both the minimum Jaccard similarity threshold ρ and the minimum cosine similarity σ are set to 0.5, the number of neighbors k in kNN is set to 10.

Table 1 shows the evaluation results of HUPC, Online-LDA, SFPM and MP tested on two datasets. F-measure and RI evaluation results show that HUPC and SFPM always perform better. However, from TPR results, we find that HUPC still has satisfactory result while the SFRM's performance is poor. It means that the HUPC works well on both clustering the texts that belong to same class into one cluster, and separating the texts that belong to different classes into different clusters. The reason is that HUPC can generate more meaningful topics within more distinctiveness words to express them. In contrast, SFPM inclines to sperate the texts that belong to different classes into different clusters. Further reason is that too much topics are generates, where each topic only contains a few terms and some of the topics even consist of non-sense terms. Therefore, although SFPM achieves high values in F-measure and RI, the generated topics are not conformed to people's knowledge so much. In addition, MP method always has a comparable performance to ours, and sometimes even better. It is because the modularity-based partition is able to infer a proper number of clusters in the clustering process. The reason why Online-LDA method always has the worst performance is that it needs to specify topic numbers. But when detecting topic from streams, different batches may discuss different numbers of topics, thus a fixed value of the topic number for the entire stream is not a good choice.

Furthermore, we find that HUPC has more stable performance than Online-LDA and SFPM, which are more sensitive to their parameters. Generally, Online-LDA can result in a higher F-measure when setting a lower value of topic number. Take the Twitterset as example, Online-LDA achieved a F-measure of 0.41 when the topic number is set to 15, while only obtained a F-measure of 0.32 when topic number is 50. However, the experimental results shows that HUPC always has stable performances on all metrics when its parameters are set to different values. Parts of related experimental results are presented in Section 4.3.

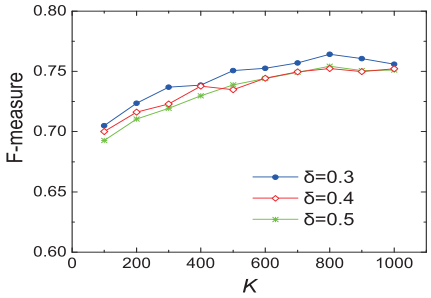

Figure 2: F-measure performance of different values for δ and K

4.3 Parameter Tuning

In this part, we investigate the effect of different parameters in HUPC on the final results. Owing to space limitations, we only demonstrate the F-measure result tested on the SinaSet. The framework achieves similar results when tested on other metrics and on TwitterSet.

First, we test the effect of parameters in pattern mining on the F-measure. Figure 2 shows the F-measure across different values of maximum overlap-degree δ and maximum pattern numbers K, where the minimum similarity ρ and σ are set to 0.5, and the number of nearest neighbors k is set to 10. From Figure 2, we make two observations. On one hand, the F-measure grows when enlarging K, and can finally tends toward stability when K is enlarged to a certain number (e.g, 600). The reason is more number of patterns usually provides more information about the topics in a batch, causing the detected topics to be more similar to the real events. But when the number of mined patterns increases to a certain threshold (i.e., 600), it has provided enough information for inferring topics from the stream. Thus, more patterns become a kind of redundancy. On the other hand, the F-measure grows when decreasing the overlap-degree δ. A low value of δ indicates less similarity between two patterns, leading to a lower information redundancy among the high utility patterns. Therefore, a lower value of δ usually achieves a high F-measure when K is fixed.

Next, we test the effect of minimum similarity ρ and σ, as well as the number of nearest neighbors k on the F-measure. Here, $\delta = 0.4$ and $K = 500$. Figure 3 indicates that different numbers of k don't have significant impact on the results, while the minimum similarity (i.e., ρ and σ) parameters impacts the result observably. It is because only the nearest neighbor that the similarity between it and a new pattern is higher than the threshold σ is qualified to participate in the

[2]http://www.nltk.org

[3]In this paper, we use jieba word segmentation toolkit for Chinese microblog segmentation. It is downloaded from http://www.oschina.net/p/jieba

Table 1: The comparison among different topic detection methods on two datasets.

	TwitterSet				SinaSet			
	HUPC-Stream	MP	On-LDA	SFPM	HUPC-Stream	MP	On-LDA	SFPM
F-measure	0.616	0.573	0.418	0.639	0.747	0.730	0.451	0.667
RI	0.780	0.702	0.222	0.837	0.856	0.832	0.713	0.799
TPR	0.101	0.134	0.040	0.081	0.253	0.235	0.066	0.141
FPR	0.099	0.211	0.038	0.022	0.076	0.082	0.032	0.021

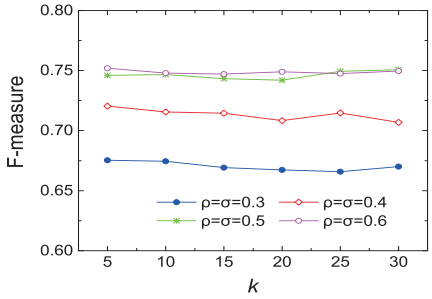

Figure 3: F-measure performance of different values for ρ, σ and k

new pattern's classification process. Most of these nearest neighbors come from the same topic cluster and thus can give a more consistent guidance. In addition, F-measure increases by enlarging the similarity threshold, and then tends to stable. It is because a lower value of similarity threshold makes two patterns that not such similar be gathered into one cluster, resulting in a wrong clustering.

4.4 Case study

In this part, we test the performance of HUPC in identifying emerging topics from coherent ones. Here, we show the result tested on TwitterSet. There are 10 event collections in TwitterSet, where each event lasts for several days. Table 2 shows the topical words of detected emerging topics. These topics came from the batch where some texts were labeled with an event id that didn't appear in previous batch. The first column shows the batch id of appearing new events, the second column shows the emerging event id, the third column shows the top-5 event-relevant words, and the last column shows the detected topical words. Each group of topical words are selected from the detected topic that is identified as emerging topic and its topical words contains event-relevant words. The event-relevant words are selected from the texts that have the same event id in the batch.

Table 2 shows that HUPC can detect most of the emerging events, especially for the ones that consist of a list of coherent terms. For example, three events `NBA`, `Ipad mini`, and `Mitt Romney` are identified by HUPC in batch 19, and most of the event-relevant words also appear in the topical words list. In addition, note that there are more than one emerging topics that relate to the same emerging event. It is because each event lasts for a long time, making all the tweets that contain the keyword (i.e., `obama`) be downloaded to fill the event-relevant text sub-collection. Therefore, it is inevitable that this sub-collection contains more than one event-relevant topics. At last, we find that HUPC sometimes fails in detecting some events, such as the event `Chanel` appearing in batch 16 and `Iphone` appearing in batch 18. Two reasons can explain the situation: 1) only a few of tweets in the batch refer to this event, or 2) the event-relevant tweets are noises. As a result, it doesn't exist a topic such that all of its topical words are so coherent with each other that can be regarded as a meaningful topic.

Table 3: Coherent topics detected from five sequential segments

Batch id	Topical words	
34	iphon, ipad, mini, win, barack obama	throne, game, iphon, mini
35	iphon, ipad, mini, win, follow	throne, game, season, watch, follow
36	iphon, chanel, ipad, game, buy	throne, game, watch, appl, wont, season
37	iphon, ipad, mini, romney obama	throne, game, watch, time, ipad
38	iphon, ipad, mini, Obama, barack obama	throne, game, watch, win, mini

Next, we demonstrate the performance of HUPC in tracking the evolution of coherent topic. Table 3 presents the topical words of two topics that both last from batch 34 to batch 38. From Table 3, we find that HUCP achieves good performance in tracking coherent topics. It is because the kNN classification is able to catch the topic that doesn't have significant changes in a long time period.

5. CONCLUSIONS

In this paper, we propose a HUPC framework for detecting topics from large scale of microblog streams. In this framework, we design a high utility pattern mining algorithm, which can use a small number of mined patterns to represent the entrie texts stream. These patterns are more powerful in expressing complete semantic information. Furthermore, we also design an incremental pattern clustering method to detect coherent and emerging topics from these patterns. Experimental study shows that the HUPC framework can achieve better performance than other baseline approaches, and is able to identify emerging topics from coherent ones. As a part of our future work, we will investigate more efficient pattern mining algorithms, where high utility patterns are not selected from the generated frequent patterns, but mined by building a novel pattern mining tree. Another task is to study how to select representative patterns rather than terms to express the detected topics.

6. ACKNOWLEDGMENTS

The work was supported by the National Science Foundation of China (NSFC, No. 61472291 and No. 61303115).

7. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

Table 2: The comparison between detected emerging topical words and standard emerging event words.

Batch id	Emerging Event id	Event- relevant words	Detected Topical words
16	2	chanel, love, don't, people, time	mitt romney, twitter, amp
	10	mitt romney, barack obama, obama, win	elect, president barack obama, barack
			retweet, win, presid, follow, obama
18	8	iphon, relationship, don't, appl, win	
19	4	facebook, twitter, follow, like, post	facebook, post, obama, barack, like
	6	Ipad, mini, win, appl, enter	Ipad, mini, win, enter, follow
	10	romney, mitt romney, mitt, obama, win	Barack Obama, realdatil, people, photo
26	5	game, throne, peopleschoic, thrones,tv	game, nba, chanel, fine, throne
			game, throne, peopleschoic, walk, dead
	11	nba, game, losangel, fine, win	nba, win, fine, thanksgiv, day, live
			nba, appl, giveaway, losangel, play
	12	walk, dead, peopleschoic, cabletvdrama	peopleschoic, cabletvdrama, throne, dead
27	10	Mitt, mitt Romney, obama, romney, peopl	mitt romney, mitt

[2] J. Allan, V. Lavrenko, D. Malin, and R. Swan. Detections, bounds, and timelines: Umass and tdt-3. In *Proceedings of topic detection and tracking workshop*, pages 167–174, 2000.

[3] L. AlSumait, D. Barbara, and C. Domeniconi. On-line lda: adaptive topic models for mining text streams with applications to topic detection and tracking. In *ICDM*, pages 3–12, 2008.

[4] Z. Chen and B. Liu. Mining topics in documents: standing on the shoulders of big data. In *SIGKDD*, pages 1116–1125, 2014.

[5] Z. Chen, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R. Ghosh. Discovering coherent topics using general knowledge. In *CIKM*, pages 209–218, 2013.

[6] Q. Diao and J. Jiang. Recurrent chinese restaurant process with a duration-based discount for event identification from twitter. In *SDM*, pages 388–397, 2014.

[7] A. Farzindar and W. Khreich. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31:132–164, 2015.

[8] B. Fung, K. Wang, and M. Ester. Hierarchical document clustering using frequent itemsets. In *SDM*, pages 59–70, 2003.

[9] G. Fung, J. Yu, H. Liu, and P. Yu. Time-dependent event hierarchy construction. In *SIGKDD*, pages 300–309, 2007.

[10] J. Guo, P. Zhang, J. Tanb, and L. Guo. Mining hot topics from twitter streams. *Procedia Computer Science*, 9(2012):2008 –2011, 2012.

[11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *SIGMOD Record*, pages 402–406, 2000.

[12] Q. He, K. Chang, and E. Lim. Analyzing feature trajectories for event detection. In *SIGIR*, pages 207–214, 2007.

[13] H. Kim, D. Park, Y. Lu, and C. Zhai. Enriching text representation with frequent pattern mining for probabilistic topic modeling. *Proceedings of the American Society for Information Science and Technology*, 49(1):1–10, 2012.

[14] C. Li, A. Sun, and A. Datta. Twevent: segment-based event detection from tweets. In *CIKM*, pages 155–164, 2012.

[15] T. Lin, W. Tian, Q. Mei, and H. Cheng. The dual-sparse topic model: mining focused topics and focused terms in short text. In *WWW*, pages 539–550, 2014.

[16] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In *SIGMOD*, pages 1155–1158, 2010.

[17] M. Newman. Modularity and community structure in networks. *Proceeding of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[18] G. Petkos, S. Papadopoulos, L. Aiello, R. Skraba, and Y. Kompatsiaris. A soft frequent pattern mining approach for textual topic detection. In *WIMS*, pages 25:1–25:10, 2014.

[19] S. Petrovic, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *HLT-NAACL*, pages 181–189, 2010.

[20] S. Phuvipadawat and T. Murata. Breaking news detection and tracking in twitter. In *WI-IAT*, pages 120–123, 2010.

[21] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(366):846–850, 1971.

[22] J. Weng and B. Lee. Event detection in twitter. In *ICWSM*, pages 401–408, 2011.

[23] J. Yin and J. Wang. A dirichlet multinomial mixture model-based approach for short text clustering. In *SIGKDD*, pages 233–242, 2014.

[24] W. Zhang, T. Yoshida, X. Tang, and Q. Wang. Text clustering using frequent itemsets. *Knowledge-Based Systems*, 23(5):379–388, 2010.

[25] X. Zhou and L.Chen. Event detection over twitter social media streams. *VLDB Journal*, 23(3):381–400, 2014.

[26] X. Zhu, Z. Ming, Y. Hao, X. Zhu, and T. Chua. Customized organization of social media contents using focused topic hierarchy. In *CIKM*, pages 1509–1518, 2014.