This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

Efficient Identification of Local Keyword Patterns in Microblogging Platforms

Xiaoyang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin

Abstract—Microblogging platforms, such as Twitter, serve as an important and efficient channel for sharing information. With the prevalence of geo-position enabled devices, a rapidly growing amount of microblogs are associated with geo-tags. Consequently, real-time analysis of the geo-tagged microblog stream has attracted great attentions. In this paper, we advocate the significance of keyword co-occurrence for geo-tagged microblogs analysis, which has been overlooked by existing studies. The co-occurrence of keywords is necessary to resolve the ambiguity in event analysis, especially when different events have overlapping descriptions. Given a geo-tagged microblog stream, we formally define the problem of identifying local (top-K) maximal frequent keyword co-occurrence patterns over geo-tagged microblog stream, namely LFP (LKFP) query. Given a query region, LFP query aims to retrieve the local maximal keyword patterns with frequency exceeding a given threshold; while LKFP query aims to identify K maximal keyword patterns with highest local frequency, in case users do not have a threshold in mind. To handle the high volume microblog stream and meet the requirement when large number of queries issued, we develop novel data structures to maintain the data stream, and propose efficient algorithms to process LFP and LKFP queries with theoretical underpinnings. The extensive empirical study on real dataset confirms the effectiveness and efficiency of our approaches.

Index Terms—Microblog stream, co-occurrence keywords, frequent pattern, top-K

1 Introduction

Nowadays, microblog, such as tweet, is playing an important role in online social media. Due to its concise feature, it can convey the information to people much faster than the traditional media. Through the microblog, users are accustomed to discuss realtime topics or events happening. These contents can serve as uncensored windows of current events.

With the proliferation of geo-position enabled devices, a large amount of microblogs are geo-tagged. For instance, recently it is reported[1] that there are about 30 millions people sending out tweets associated with geo-tags into the Twitterverse, and 2.2 percent of tweets (about 4.4 million tweets a day) provide location data together with the text of their posts. Thus, microblogs with geo-tags are studied as the human sensor of social events happening in the specific area [1], *i.e.,* local events. As a result, geographic factor is a big concern in the massive microblog stream analysis and there is an emerging call for effective and efficient data analysis techniques to make sense of the geo-tagged microblog stream. Usually, interesting/important events are associated with high frequent keywords. Thus, the problem of identifying the local frequent or bursting keywords over geo-tagged microblogs has received growing attentions in the literature [1], [2], [3], [4], [5], [6].

Motivation. There are two limitations in the existing work. Firstly, in most existing works (*e.g.,* [4], [5], [6]), only single bursting hashtags or frequent keywords are identified and returned as a description of local events. Nevertheless, these works overlook the co-occurrence information of keywords (*i.e.,* the keywords appeared in the same microblogs), in which they may fail to delivery interesting patterns or even mislead users. Secondly, some works (*e.g.,* [7], [8]) have considered to use keywords pattern for event description. They either try to group identified frequent/burst single keywords by re-checking the original microblogs or to cluster the keywords based on their spatial distribution. However the re-checking cost may be prohibitively expensive in the streaming environment. Clustering the keywords based on their spatial distribution may generate misleading results when events with similar keywords happened close to each others. Consequently, it will need further refinement to improve results.

Following are two motivating examples to illustrate the limitations in the existing works, and clarify the significance of keywords co-occurrence information.

Example 1. *Everyday, many people send microblogs about the local new events to inform or share with friends. Suppose there are two exciting events in the downtown on the same day, 1) a new Apple store opened and 2) a popular restaurant named "Sokyo" provides discount for customers. We may expect that there are lots of microblogs about these two local events, where "Apple", "Sokyo", and "discount" will become frequent keywords (i.e., frequently mentioned in the microblogs) in downtown area.*

So for the first class of work, single frequent keywords will be detected and returned. Nevertheless, without the co-occurrence knowledge of these keywords, customers may not be able to capture the important information (e.g., discount at Sokyo) or even be misled since customers may conclude that there is a promotion in the Apple store. For

- *Xiaoyang Wang, Ying Zhang are with University of Technology, Sydney NSW 2007, Australia. Email:{Xiaoyang.Wang, Ying.Zhang}@uts.edu.au*
- *Wenjie Zhang, and Xuemin Lin are with School of Computer Science and Engineering, The University of New South Wales, Sydney, NSW 2052, Australia. E-mail: {xiaoyangw, zhangw, lxue}@cse.unsw.edu.au*

1. http://www.futurity.org/tweets-give-info-location

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

the second class of work, if we group the keywords based on their spatial distribution, all of them will be grouped together (the three keywords are uniformly distributed in the area), so the problem is still not resolved. In addition, as the twitter stream is rapidly updated, we cannot maintain all of them in memory for the costly rechecking.

Therefore, besides the identification of individual frequent keywords in a particular region, it is critical to keep the frequent keywords co-occurrence information (e.g., "Sokyo" and "discount") so that users can obtain more abundant and accurate information.

Example 2. *Suppose a famous movie star Actor-A come to Sydney downtown and there is a serious car crash in the downtown in the same day. Then we may expect "Actor-A downtown" will appear frequently in the Actor-A's fans' tweets, while "car crash downtown" will be mentioned a lot in the local residents' tweets. If we simply return "Actor-A car crash downtown" as a result, it will become a disaster for the actor. Similar to Example 1, we cannot resolve the problem by grouping the keywords based on some similarity measurement. We must keep the co-occurrence of keywords to tell users that "Actor-A downtown" and "car crash downtown" are two different events.*

As shown in the examples, it is necessary to maintain the co-occurrence of keywords in order to provide a precise description of the local events/patterns. Even though the found pattern size in microblog may be small, it could lead to misunderstandings if we ignore the co-occurrence of the keywords.

Given a query region, *e.g.,* Sydney, we investigate the problem of identifying local maximal frequent keyword co-occurrence patterns over a geo-tagged microblog stream. We define the problem of finding interesting problems under two models: the first one is threshold based model, which is to retrieve the local patterns with frequency larger than a threshold; the second one is to find the top-k frequent patterns in the given query area. Since we cannot maintain all the data in main memory in the streaming context, we aim to solve these two problems by continually maintaining a summary of the microblog stream. The first model is more for the expert users who have the background knowledge about the data distribution. By setting a proper threshold, they can quickly find the interesting local events. The second model is more for the normal users, who do not have a proper threshold in mind.

Challenges. To the best of our knowledge, this is the first work to systematically investigate the problem of identifying local (top-K) maximal frequent keywords co-occurrence patterns over geo-tagged microblog stream using sketch techniques. The main challenges of the problem lie on two aspects.

The first one is that the data stream is large and rapidly updated. It is essential to continuously maintain a concise summary of the data stream to answer the two queries with high accuracy. Moreover, the summary should be able to support rapid update, and seamlessly capture the co-occurrence information of keywords. The second one is to meet the high-performance requirement, since there may be plenty of queries issued by different users. So the maintained summary should be well organized to support arbitrary query regions and thresholds. However, the traditional frequent pattern mining methods over data stream mainly focus on generating frequent patterns over the entire data stream by maintaining a global index structure [9], [10]. It is non-trivial to extend the existing techniques to deal with arbitrary query regions and thresholds.

A simple and intuitive solution is to continuously maintain a uniform sample of the data stream using spatial order (*e.g.,* Z-order value or grid index) and then apply the existing techniques (*e.g.,* [11]) on the samples within the query region to answer the queries. However, a subtle difficulty of our problems lies on the fact that the query region might be arbitrary, which indicates that we cannot continuously maintain the data structures utilized in existing works. Hence it may have to undertake the mining task from scratch for each query.

In this paper, we devise a novel sketch, inverted bottom-k sketch (IK sketch for short) to properly capture the spatial and textual information of geo-tagged microblog stream. In a nutshell, an IK sketch is a summarized inverted list structure in which only a limited number of objects are carefully and consistently maintained for each individual keyword in the data stream. Then the frequency of the pattern can be derived based on the estimation of the multiple sets intersection size using IK sketch. To accelerate the enumeration of all the frequent patterns and avoid generating too many candidates, we develop the sketch graph and enhanced sketch graph, which keep track of the co-occurrence of keywords in the sketch approximately. To meet the response requirement of many queries issued, we develop efficient algorithms based on the data structure proposed to answer the queries.

Contributions. Our principle contributions are summarized as follows.

- This is the first work that systematically studies the problem of local (top-K) maximal frequent keyword co-occurrence patterns identification in geo-tagged microblog stream using sketch techniques, which is an essential tool for microblog stream analysis in a wide spectrum of applications.
- We propose the inverted bottom-k sketch, sketch graph and enhanced sketch graph structure, which summarize the microblog stream and can properly capture both spatial and frequency information with limited space. Theoretical analysis is conducted for the correctness and accuracy guarantee of the techniques.
- Efficient algorithms are proposed to support LFP and LKFP queries based on the structures proposed.
- Extensive empirical study on real dataset demonstrates the efficiency and effectiveness of our techniques proposed in the paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

Roadmap. The rest of the paper is organized as follows. Section 2 presents the problem definition, and briefly surveys the most related works and necessary techniques used in this paper. Section 3 introduces IK sketch and an Apriori framework based method to support the LFP query. Section 4 introduces the enhanced sketch graph structure to further accelerate the LFP query processing. In Section 5, we introduce the algorithms for answer LKFP query based on the data structures proposed. The effectiveness and efficiency of the techniques are demonstrated over real Twitter dataset in Section 6. Finally, we concludes this paper in Section 7. In Appendix A, the correctness and accuracy analysis of the sketch based method is presented. In Appendix B, we present the algorithm for maintaining the samples in sliding window model.

2 BACKGROUND

In this section, we first formally define the problem of identifying local (top-K) maximal frequent keyword co-occurrence patterns over geo-tagged microblog stream. Then we introduce the related existing works as well as the key techniques employed in this paper. Table 1 summarizes the notations frequently used in this paper.

Notation	Meaning
d, d_i, o	spatio-textual object (i.e., geo-tagged microblog)
$\mathcal{D}$	a stream of spatio-textual objects
$\mathcal{V}$	the vocabulary of terms (keywords)
R	a query region
t, t_i, t_j	a term (keyword)
$d.\mathcal{V}, o.\mathcal{V}$	terms (keywords) of d and o
n	the number of objects in $\mathcal{D}$ or sliding window
$\mathcal{L}$	IK sketch
$\mathcal{L}(t)$	bottom-k sketch of keyword t inverted list
l	sample budget
P, P_i	a keyword pattern
K	the K in top-K
k	the k in bottom-k
h_R^t	the largest hash value of keyword t in R
h_S^k	the k-th smallest hash value of a set S of object
A, B, S	a set of objects

TABLE 1
The Summary of Notations

2.1 Problem Definition

A microblog stream consists of a set $\mathcal{D}$ of geo-tagged microblogs. Each geo-tagged microblog $d \in \mathcal{D}$ is associated with a spatial location $d.loc$ in a 2-dimensional space and a set of keywords $d.\mathcal{V}$ from a vocabulary $\mathcal{V}$. A keyword pattern P consists a set of keywords, *i.e.*, $P \subseteq \mathcal{V}$. Given a region R, $\mathcal{D}_R$ denotes the microblogs falling in R, that is $\mathcal{D}_R = \{d \mid d.loc \in R \ \wedge \ d \in \mathcal{D}\}$. In the paper hereafter, we abbreviate the microblog stream and the geo-tagged microblog as *data stream* and *object* respectively, if there is no ambiguity.

Definition 1. *(Pattern Local Frequency) Given a data steam $\mathcal{D}$, a keyword pattern P and a region R, the pattern local frequency $f(P, R)$ of P is the number of occurrence of P in $\mathcal{D}_R$, that is $f(P, R) = |\{d \mid d \in \mathcal{D}_R \wedge P \subseteq d.\mathcal{V}\}|$.*

Definition 2. *(Local Maximal Frequent Keyword Pattern, LFP). Given a query region R and a frequency threshold θ, a pattern is a local frequent pattern if its pattern local frequency is larger than θ, i.e., $f(P, R) \geq \theta$. A local frequent pattern is a LFP if there is no superset of the pattern that is also a local frequent pattern.*

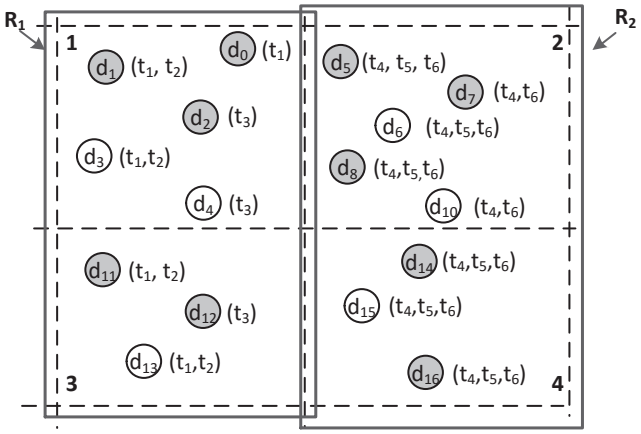


Fig. 1. Example of Geo-tagged Microblogs

For example, Figure 1 shows a set of geo-tagged microblogs in the space, where their associated keywords are listed besides. Given the query region R_1 and threshold $\theta = 3$, the patterns $\{t_1\}, \{t_2\}, \{t_3\}$ and $\{t_1, t_2\}$ are local frequent. However, only the patterns $\{t_1, t_2\}$ and $\{t_3\}$ are LFPs, because $\{t_1\}, \{t_2\}$ are subsets of $\{t_1, t_2\}$.

In some scenarios, normal users may not have the knowledge of data distribution in the local space and do not have a proper frequency threshold in mind. A small threshold may result a lot of output patterns and a large threshold may lead to few or even no results. In both cases, the found results will not be interesting to the users. So we also investigate the local frequent patterns in the top-K semantics, which is formally defined as follows.

Definition 3. *(Local top-K Maximal Frequent Pattern, LKFP) Given a query region R, let $\mathcal{P}_{max}^R = \{P_1, P_2, ...\}$ denotes the set of patterns appeared in R, and for each pattern $P_i \in \mathcal{P}_{max}^R$, there is no pattern $P_i' \supset P_i$ with $f(P_i', R) = f(P_i, R)$. The local top-K frequent patterns consist of K patterns in $\mathcal{P}_{max}^R$ with the highest frequency.*

For example, in Figure 1, the query region is R_1. When $K = 1$, pattern $\{t_1\}$ is the LKFP. When $K = 3$, patterns $\{t_1\}$, $\{t_3\}$ and $\{t_1, t_2\}$ are all LKFPs. As we notice, both $\{t_1\}$ and $\{t_1, t_2\}$ are included in the results when $K = 3$, this is because they have different local frequency even if $\{t_1\}$ is a subset of $\{t_1, t_2\}$.

Problem Statement. Given a data stream $\mathcal{D}$, an positive integer θ (K) and a query region R, the LFP (LKFP) query is to find all the LFPs (LKFPs). Since the data stream is usually quite large and frequently updated, we aim to carefully construct a small summary of the data stream $\mathcal{D}$ to answer the LFP query and LKFP query efficiently and accurately.

Remark 1. *Note that the constrains in LFP and LKFP have a bit difference. In LFP, we remove the patterns which have a frequent superset pattern from the results. While in*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

the LKFP, we remove the patterns which have a superset pattern with the same frequency. This is because in LFP query we try to reduce the number of patterns returned and make it easy for users to process the results. However, the algorithms proposed in this paper can be easily extended to support the same constrain as that in LKFP query.

2.2 Related Work

To the best of our knowledge, there is no existing work on identifying local (top-K) maximal frequent keyword co-occurunce patterns over geo-tagged microblog stream based on sketch techniques. The most related works are presented in the following three aspects.

Spatial frequent keyword identification. Recently, a large number of research efforts have been devoted to identify spatial frequent or burst keywords in geo-temporal microblog stream, which has been proven to be useful in various applications. Existing works try to find interesting keyword patterns by considering the spatial and temporal dimensions of microblogs, but the co-occurrence information of keywords is overlooked. For instance, the frequency of each individual keyword/hashtag is counted regarding particular regions in [4], [5], [6], [12], [13]. In [8], the correlation of keywords is considered in the EvenTweet system where keywords with close spatial proximity are grouped together. However, the co-occurrence of keywords is not kept, which is an important property as shown in our motivating example. In [7], authors find frequent/bursting keywords at first, and then checks the original microblogs to group related keywords together. However, it needs to cache the original microblogs for checking the grouping condition, which may not be applicable in data stream model. In addition, checking the co-occurrence of keywords in microblogs can be time consuming, as it needs to scan all the related microblogs and check different combinations each time. In [1], authors associate tweets with popular local to identify local events. It also utilizes the location in the tweets, like [14] to bust the analysis. Foteini et al. [2] consider the co-relation of pair tags to identify events in time. In [3], authors try find local burst keyword by extracting an accurate distribution of keywords. However, these works did not solve the keywords co-occurrence issue in our motivation examples.

Frequent pattern mining. On the other hand, approximate frequent pattern mining over data stream has attracted significant attention in the literature (See survey in [15]) where the co-occurrence of keywords in each transaction is considered by existing works. Nevertheless, the spatial location is not considered. Their techniques focus on developing efficient methods to mine frequent patterns over the whole data stream or dataset, and it cannot be trivially extended to support the problem studied in this paper since the query region is not known beforehand. There are some existing works on spatial co-location pattern mining (See survey in [16]). However, this problem is inherently different from ours, because it does not consider the co-occurrence of keywords in the same objects. Moreover, their techniques are not developed in the context of data streams.

Set intersection size estimation. If we store each keyword appeared in an inverted list, the frequency of a pattern equals to the intersection size of corresponding keywords' inverted lists. The problem of set intersection size estimation is to estimate the cardinality of set intersection with sketch/sampling methods, which has been extensively studied [17], [18], [19]. Recently, the problem of selectivity estimation for spatial keyword search is investigated in [20]. The research focus is to boost the estimation accuracy by exploiting local correlations when there are no sufficient samples in the query range (i.e., usually not the frequent pattern). The techniques are not suitable to our problems since we focus on frequent pattern mining where the boosting technique is not beneficial considering the boosting cost on frequent patterns. In addition, there are several works [21], [22] that tend to calculate an upper bound of set intersection size, which is orthogonal to our work. In this paper, we utilize bottom-k [19] sketch to estimate the intersection size of sets.

2.3 Preliminaries

In this section, we briefly introduce the bottom-k sketch [19], which is used in our IK sketch to capture the co-occurrence of keywords and estimate the cardinality of patterns. Bottom-k sketch is designed for estimating the number of distinct values in a multiset. Suppose N distinct points are uniformly distributed over (0, 1), then the expected distance between any two adjacent points is $\frac{1}{N+1} \approx \frac{1}{N}$. Given a multiset $S = \{v_1, v_2, \cdots, v_n\}$ and a truly random hash function h, each distinct value v_i in the set S is hashed to (0, 1) and $h(v_i) \neq h(v_j)$ for $i \neq j$. The bottom-k sketch consists of the k smallest hash values, i.e., $\mathcal{L}(S) = \{h(v_i) \mid h(v_i) \leq h_S^k \wedge v_i \in S\}$, where h_S^k is the k-th smallest hash value of the set. So the number of distinct value can be estimated with Equation (1).

$$\hat{D} = (k-1)/h_S^k \tag{1}$$

Based on the analysis in [23], the expected relative error of the estimator is $\sqrt{\frac{2}{\pi(k-2)}}$.

Union Operation. Consider two sets A and B, with corresponding bottom-k sketch $\mathcal{L}(A)$ and $\mathcal{L}(B)$ of size k_A and k_B. In this paper, we use $\cup_m(\cap_m)$ to denote the union (intersection) operation for multisets, and we use $\cup$ $(\cap)$ for simplicity when there is no ambiguity. An important property of bottom-k sketch is that it is closed under set union operation, and it utlizes the sketch information sufficiently. To obtain the sketch $\mathcal{L}(A \cup B)$ under union operation is shown as follows.

$$\mathcal{L}(A \cup B) = \{v | v \in \mathcal{L}(A) \cup \mathcal{L}(B) \ \wedge \ v < h_{A\cup B}^k\}$$

where $h_{A\cup B}^k = \min\{h_A^k, h_B^k\}$. To estimate the union size of sets A and B, *i.e.*, $|A \cup B|$, we have the following

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

equation:

$$\hat{D}_{\cup} = |\mathcal{L}(A \cup B)|/h^k_{A\cup B} \quad (2)$$

Intersection Operation. Similarly, the intersection size of sets A and B can be estimated using the equation as follow:

$$\hat{D}_{\cap} = |\mathcal{L}(A) \cap \mathcal{L}(B) \cap \mathcal{L}(A \cup B)|/h^k_{A\cup B} \quad (3)$$

Equation (3) is an unbiased estimation of set intersection size of A and B, which correctness will be proved latter in the paper. Note that the above equations for set union/intersection estimation can be immediately extended to support multiple sets operations.

3 IK SKETCH BASED APPROACH

In this section, we introduce the IK sketch and sketch graph, which can properly capture the co-occurrence of keywords in microblogs, as well as the algorithm which continuously maintains the proposed structures. The summary maintaining algorithm proposed in this section is for streaming model. In Appendix B, we will introduce how to maintain the summary over the sliding window model. Finally, we present the algorithm to answer the LFP query by using the data structure.

3.1 IK Sketch and Sketch Graph

In this section, we introduce the IK sketch and sketch graph, which is used to summarize the data stream information and answer the queries studied in this paper. Before describing the details of the data structures, we firstly introduce the general ideas and motivations for the data structures.

Motivation. There are three main requirements for the summary: 1) it can estimate the pattern local frequency accurately; 2) it is friendly for update; 3) it can be used to efficiently answer the queries. The IK sketch and sketch graph are designed according to these requirements:

- In general, IK sketch maintains the bottom-k sketch for keywords. Since bottom-k is effective for estimating the cardinality of sets intersection, we can estimate the pattern frequency based on the bottom-k sketch of keywords. In addition, bottom-k sketch is of low cost for updating in data stream.
- To answer the queries efficiently, a major problem to reduce the number of enumerated candidate patterns. The data structures used in the traditional frequent pattern mining algorithms are designed for the entire data and space expensive, while the query region in our problem may be arbitrary. Sketch graph is a light weight data structure that maintains the occurrence of keywords in coarse-grained and works in practice.

3.1.1 IK Sketch

The IK sketch is designed to estimate frequent of a pattern in an arbitrary query region accurately and efficiently.

IK sketch Data Structure. Given a sample size l and a data stream $\mathcal{D}$, IK sketch $\mathcal{L}$ continuously maintain a fixed size set with l objects in $\mathcal{D}$. For each coming object d, we generate its hash value $h(d)$, where $h(x)$ is a truly random hash function and $h(x) \in (0, 1)$. Then l objects with the smallest hash values in the data stream seen so far are kept. Specially, the l objects in IK sketch are organized in inverted lists format. That is, for each keyword t in the sampled objects, we build an inverted list, denoted as $\mathcal{L}(t) \in \mathcal{L}$, where $\mathcal{L}(t) = \{h(d) \mid h(d) \leq h^l_{\mathcal{D}} \text{ and } t \in d.\mathcal{V}\}$, where $h^l_{\mathcal{D}}$ is l-th smallest hash value in $\mathcal{D}$.

For a keyword t, $\mathcal{L}(t)$ maintains its sketch for the global space. In order to accelerate the search for local region, we maintains $\mathcal{L}(t)$ according to the spatial order (*e.g.,* Z-order [24] or grid index) of the objects. The spatial order can significantly reduce the cost of arbitrary spatial range search since the spatial proximity of objects are well preserved. We also maintain the keywords distribution with respect to the spatial order to fast retrieve the keywords falling in a query region.

Example 3. *Suppose we maintain 10 samples in the IK sketch $\mathcal{L}$ for the objects in Figure 1, where the dark nodes in Figure 1 are samples. The corresponding IK sketch sketch is shown in Figure 2(a), and the value besides each object d_i is its hash value. The spatial order applied are based on the grids in the space. For example, keyword t_1 appears in 3 samples on the left two grids, thus $\mathcal{L}(t_1)$ is partitioned into 2 parts which is illustrated with two underlines for t_1. In the same grids, sample are sorted by their hash value. Given IK sketch and the query region R_1, we can efficiently estimate the local frequency of t_1 as $f(t_1, R_1) = (3-1)/0.4 = 5$. Similarly, $f(\{t_1, t_2\}, R_1) = 1/0.25 = 4$.*

Maintain IK sketch. In the IK sketch, as the objects (*i.e.,* microblogs) come, we continually keeps l samples with the smallest hash value. For each new coming object d, we directly add it to $\mathcal{L}$ if the current sample size $|\mathcal{L}|$ is smaller than l. Otherwise, we compare its hash value $h(d)$ with $h^l_{\mathcal{D}}$ to decide if we add d into the samples or not.

In Algorithm 1, we present the details of maintaining IK sketch, *i.e.,* the procedure when a new object comes. Let τ be the largest hash value in the current samples. Note that when the sample size $|\mathcal{L}|$ equals l, $\tau = h^l_{\mathcal{D}}$. In Line 2, we check if object d should be added to the sample. If the sample size is already l and $h(d) \geq \tau$, we just discard it directly. Otherwise, we add d into the samples, and for each keyword in $d.\mathcal{V}$, we update the corresponding inverted bottom-k sketch from Line 3 to Line 5. In Line 6, we check if we have more than l samples. This happens when the original sample size is l and the new object's hash value is smaller than $h^l_{\mathcal{D}}$. In this case, we remove the sample with largest hash value to keep the sample size unchanged.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

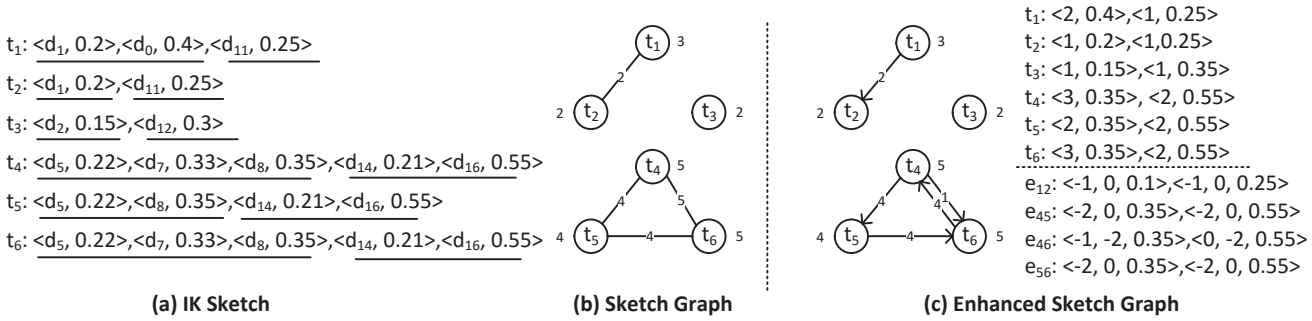


Fig. 2. IK Sketch, Sketch Graph and Enhanced Sketch Graph Corresponding to the Example in Figure 1

Algorithm 1: Maintain IK Sketch

Input : d : a new arriving object; l : sample size.
Output: Updated IK sketch $\mathcal{L}$

```
τ ← largest hash value in L ;
if h(d) < τ or |L| < l then
    L ← L ∪ {d} ;
    for each t_i ∈ d.V do
        L(t_i) ← L(t_i) ∪ {d} ;
if |L| > l then
    o ← sample in L with largest hash value ;
    L ← L \ {o} ;
    for each t_i ∈ o.V do
        L(t_i) ← L(t_i) \ {o};
```

Example 4. *Suppose we have a new object d comes in Figure 1 and $l = 10$. Thus, $\tau = h^l_{\mathcal{D}} = h(d_{16}) = 0.55$. If $h(d) > 0.55$, we discard it directly. Otherwise, we delete d_{11} from the IK sketch and add d.*

Discussion. Suppose there are n ($n \gg |l|$) objects seen as far in the data stream. Given a truly random hash function $h(x)$, each object is selected with probability l/n. Consequently, the expected size of $\mathcal{L}(t_i)$ is $\sum_{d\in\mathcal{D}} Pr(t_i \in d.\mathcal{V}) \times l/n = freq(t_i) \times l/n$, where $freq(t_i)$ is the frequency of keyword t_i in the data stream. It indicates that the resource allocation among inverted bottom-k sketch of the keywords is proportional to their frequencies.

3.1.2 Sketch Graph

Limitation of IK Sketch. Given the IK sketch, we are able to estimate the frequency of any pattern in a query region. Thus, we can answer the LFP query and LKFP query with IK sketch only. For example, to answer the LFP query, we first retrieve the sketch within the query region. Then, following the Apriori framework, we firstly find the local frequent pattern of size 1. In the following each iteration, we expand the pattern with one more frequent keyword, until we find all the local frequent patterns.

The bottleneck of this method is that it will enumerate many non-frequent patterns. For example, in Figure 1, given the query region R_1 and the frequency threshold 1, we firstly find the local frequent keywords t_1, t_2, t_3. In next iteration, we need to check if $\{t_1, t_2\}$, $\{t_2, t_3\}$ and $\{t_1, t_3\}$ are frequent or not by doing the sketch intersection. However, $\{t_2, t_3\}$ and $\{t_1, t_3\}$ have no co-occurrence in local, so they cannot be frequent and the checking time for these patterns is a waste. When the number of local frequent keywords becomes larger, the number of redundant checks will grow fast.

Data Structure. Since we want to answer the query efficiently and accurately with limited space, we aims to design a light-weight data structure to facilitate the pattern enumeration. A sketch graph is continuously maintained for the keywords in IK sketch to approximatively describe the co-occurrence information among keywords.

Formally, a sketch graph is an undirected weighted graph $G = (V, E)$, where V and E denote the set of nodes and edges in G respectively. A node $v_i \in V$ corresponds to a keyword t_i in IK sketch, and there is an edge e_{ij} between two nodes (keywords) v_i and v_j if there is a co-occurrence of keywords t_i and t_j in a sample $d \in \mathcal{L}$. The weight of v_i is the frequency of the keyword in $\mathcal{L}$, and the weight of an edge is the number of co-occurrences of two keywords in $\mathcal{L}$. *Hereafter, we use keyword t_i to denote node v_i when there is no ambiguity.*

Example 5. *As shown in Figure 2(b), it is a sketch graph corresponding to the IK sketch in Figure 2(a). The weight of node t_1 is 3, since t_1 has appeared 3 times in the sketch. Given the sketch graph, we can directly tell that $\{t_2, t_3\}$ and $\{t_1, t_3\}$ will not be frequent patterns, since there is no edge between the keywords.*

Maintain Sketch Graph. To maintain the sketch graph, we continually check each added sample object d. Naively, for each new keyword in the sample, we create a new node in the sketch graph with initial weight 1; if it already exists, we increase its node weight by 1. Similar operation is applied for each pair of keywords, *i.e.*, edges. The weight of a node (edge) will be decreased if the related samples are removed, and the node or edge will be obliterated if its weight

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

becomes zero.

Discussion. We remark that only the keywords appeared in the sketch $\mathcal{L}$ may have corresponding nodes in the sketch graph. We observe that the number of nodes in the graph is much smaller than the number of distinct keywords in dataset. This is because in real datasets or natural languages, the keyword follows the Zipf's distribution [25]. As in the experiment, the sketch graph only takes up to 4.8% of the space used by samples. If we only maintain the promising edges and nodes by setting a threshold, the space is down to 1.1% of the space used by samples. The percentage of space used by sketch graph becomes even smaller under the sliding window model.

3.2 IK Sketch based LFP Identification

In this section, we introduce the algorithm for answering LFP query based on IK sketch and sketch graph.

Pruning Rules. Naively following the Apriori framework, the number of enumerated candidate patterns will be large. Fortunately, given the sketch graph, we can use the following rules to reduce the number of of enumerated patterns.

- Each pattern should form a clique on the sketch graph, *i.e.,* each pair of keywords in the pattern have co-occurrence.
- The upper bound of the pattern $P's$ frequency should be larger than the given threshold θ.
- Any subset of P should fulfil the pervious two rules.

Based on the first rule, pattern $\{t_1, t_3\}$ will not be enumerated in Figure 1, since it is not a clique. The upper bound of a pattern P local frequency in R can be calculated as I_R^{up}/h_R^P. I_R^{up} is the upper bound of sketch intersection size among the keywords of P in region R, *i.e.,* $|\cap \mathcal{L}(t_i, R)|$ for $t_i \in P$, where $\mathcal{L}(t_i, R)$ is the sketch of t_i in region R. If we have not calculate $|\cap \mathcal{L}(t_i, R)|$, we use the minimum intersection size of any subsets of P and edge weights as I_R^{up}. h_R^P is the minimum hash value of $h_R^{t_i}$ for $t_i \in P$. For example, in Figure 1, given the query region R_1, we want to calculate the upper bound of pattern $\{t_1, t_2\}$ local frequency. Since we have not calculated the intersection size between $\mathcal{L}(t_1, R_1)$ and $\mathcal{L}(t_2, R_1)$, we use the edge weight $w(e_{12}) = 2$ as I_R^{up}. The maximum hash value of t_1, t_2 in the region R is 0.4 and 0.25 respectively, thus $h_R^P = \min\{0.4, 0.25\} = 0.25$. Thus the upper bound of $f(\{t_1, t_2\}, R_1)$ can be calculated as $2/0.25 = 8$.

Algorithm. Algorithm 2 illustrates the details of LFP query processing using IK sketch and sketch graph. It follows the Apriori framework and expands the patterns following the edges in the sketch graph, and it utilizes pruning rules to reduce the number of patterns enumerated. We maintain a maximum heap for the objects in $\mathcal{L}$, thus we can quickly find the sample with largest hash value.

In Line 2, all keywords in the $\mathcal{L}$ fallen in R, *i.e.,* $\mathcal{V}(R)$, is firstly retrieved. Since each $\mathcal{L}$ is organized based on spatial order, we can quickly retrieve the sketch in the query area. Then we estimate the local frequency for each keywords and keep the frequent one in S_1. Then start from $k = 2$, we expand the patterns in S_{k-1} with one keywords each time, and the enumerated patterns should fulfil the pruning rules above. For the patterns pass the pruning rules, we estimate its local frequency in Line 10 to decide if we will keep this pattern or not. The processing continues until we find all the local frequent patterns.

Example 6. *For example, in Figure 1, given the query region R_1 and the frequency threshold 3. We first retrieve the keywords fall in R_1, i.e., t_1, t_2, t_3. Based on their sketch, we can estimate the local frequency of them as $5, 4, 3.3$ respectively, so $S_1 = \{t_1, t_2, t_3\}$. Then we expand the patterns in S_1, only pattern $\{t_1, t_2\}$ pass the pruning rules and is added into C_2. t_3 fails to expand since it has no neighbours in the sketch graph. After estimating the frequency of patterns in C_k, $\{t_1, t_2\}$ is added to S_2. Then the processing is terminated, since no patterns in S_2 can be further expanded.*

Algorithm 2: IK sketch based LFP Identification

Input : R : query region, $\mathcal{L}$: IK sketch of $\mathcal{D}$, G : sketch graph, θ : frequency threshold.
Output: LFP in R

1 $\{S_i\} \leftarrow \phi$;
2 **for each** $t_i \in \mathcal{V}(R)$ **do**
3 $\quad \hat{D}_{t_i} \leftarrow$ estimate the size of $t_i \in R$ based on Equation 1 ;
4 $\quad$ **if** $\hat{D}_{t_i} \geq \theta$ **then**
5 $\quad\quad S_1 \leftarrow S_1 \cup \{t_i\}$;
6 $k \leftarrow 2$;
7 **while** $S_{k-1} \neq \phi$ **do**
8 $\quad C_k \leftarrow$ expand the patterns in S_{k-1} based on sketch graph with one keywords ;
9 $\quad$ **for each** $P \in C_k$ **do**
10 $\quad\quad \hat{D}_c \leftarrow$ estimate the size of P based on Equation 3 ;
11 $\quad\quad$ **if** $\hat{D}_c \geq \theta$ **then**
12 $\quad\quad\quad S_k \leftarrow S_k \cup P$;
13 $\quad k \leftarrow k + 1$;
14 **return** $\cup S_i$

Discussion. In worst case, the sketch graph will offer no pruning power. In this case, we may need to enumerate all the local patterns and check their local frequency by doing sketch intersection. Suppose, there are a local frequent keywords, the average sketch length in the query region is b, and average frequent pattern size is c. Then the time complexity is bounded by $O(bc \cdot 2^a)$ in the worst case, since we need to estimate $O(2^a)$ patterns frequency and each cost is bc.

4 Enhanced Sketch Graph based Approach

Based on IK sketch and sketch graph, we can answer the LFP query efficiently and accurately. However,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

there are still some limitations in the efficiency of the query processing. In this section, we firstly introduce the enhanced sketch graph, which store more information in the sketch graph to further reduce the pattern enumeration and verification cost. Secondly, based on the new data structure, we propose a depth-first order based search algorithm to reduce the number of intermediate patterns enumerated in the LFP query.

The accuracy of answering the LFP query only depends on IK sketch, while sketch graph only accelerate the query processing. Similarly, the enhanced sketch graph also only accelerate the query processing.

4.1 Motivation

In this section, we introduce the motivations of designing enhanced sketch graph and the depth-first order based algorithm.

In Algorithm 2, to answer LFP query, we many need to enumerate many candidate patterns and check the co-occurrence of keywords in the sketch. As we explore the real twitter cleaned dataset (the dataset cleaning procedure can be referred to the experiment section dataset description), we found the frequent local patterns can be classified into two classes based on the pattern size.

- In the first class, the pattern size is usually very short, *e.g.*, less than four keywords. This is quite natural to understand that for most events, the core event can be described in very few words, such as "NBA final", "open week", "Australia national museum", etc. While people tend to use different words to describe the events. Consequently, only the core events keywords will become local frequent.
- In the second class, the pattern size can be very long. This usually happens in some national news, weather report, advertisements, etc. In this class, the tweets are related to some ground truths or spams. For example, at the begin of the day, many official news account we sends tweets about the local weather and some people may retweets these tweets as a reminder. Most twitter accounts are using the same words to describe these events, like weather report which may only have slightly different in temperature. In the experiments, we list examples of found pattern for weather report and advertisement.

The enhanced sketch graph is motivated by the property of microblogs in the first class. If we can accelerate the verification or pruning of patterns with size less then four with small cost, the efficiency can be greatly improved. When there are long frequent patterns, the depth-first search will reduce the number of intermediate results enumerated.

4.2 Enhanced Sketch Graph

In the enhanced sketch graph, we utilize the first property, we add more information to maintain the co-occurrence of patterns within size three. In addition, we maintain more spatial information in the new data structure to reduce the verification cost. We modified the sketch graph in two aspects: 1) pattern co-occurrence information; 2) spatial information. As shown in Figure 2(c) is the enhanced sketch graph corresponding to the example in Figure 1.

Data Structure: Pattern Co-occurrence Information. As motivated that there are large number of frequent patterns are of short size, we change the sketch graph into a directed graph to approximately maintain the occurrence of patterns with size less or equal than three.

In an enhanced sketch graph $G_e = (V, E_e)$, the node set V is the same as that in the sketch graph, denoting each keyword appeared in the IK sketch. Suppose there is a global order among all the keywords (nodes). Given two nodes $v_i, v_j \in V$, we have $v_i < v_j$ if $i < j$, similarly for t_i, t_j. An edge $e_{ij} \in E_e$ is a forward edge, if $i < j$, otherwise e_{ij} is a backward edge.

Definition 4. *(Successive Keywords) Given a sampled object d and two keywords $t_i, t_j \in d.\mathcal{V}$, we say t_i and t_j are successive in d if $t_i < t_j$ and there is no keyword $t_k \in d.\mathcal{V}$ which satisfies $t_i < t_k < t_j$.*

Given two keyword t_i, t_j and $t_i < t_j$, then weight $w(e_{ij})$ of edge e_{ij} is the number of co-occurrence when the two keywords are successive in the sampled objects; the weight $w(e_{ji})$ of edge e_{ji} is the number of co-occurrence when the two keywords are not successive. Note that the sum of the edges' weight between two nodes, *i.e.*, $w(e_{ij}) + w(e_{ji})$, equals the number of co-occurrence of the two nodes in the IK sketch.

Example 7. *As shown in Figure 1, t_4 and t_5 are successive keywords in all the samples, thus $w(e_{45}) = 4$ and $w(e_{54}) = 0$. t_4 and t_6 are successive keywords in sample d_7, and not successive in samples d_5, d_8, d_{14}, d_{16}, in hence, $w(e_{46}) = 1$ and $w(e_{64}) = 4$. $w(e_{46}) + w(e_{64}) = 5$ is the number of co-occurrence of t_4, t_6 in the samples.*

Given the enhanced sketch graph and a pattern $\{t_i, t_j, t_k\}$ and $i < k < j$. We do not need to check the pattern if $w(e_{ki}) = 0$ or the estimated number of co-occurrence is less than the threshold when t_i and t_k are not successive. For a pattern with size larger than three, any subset of size three should fulfil the backward edge requirement.

Data Structure: Pattern Spatial Information. Changing the edges in sketch graph into directed edges, we store more keywords co-occurrence information. However, these co-occurrence information is for global space. When querying a part of the keywords, we still needs to do the sketch intersection. For instance, in Figure 1, when the query region is grid 3 and the threshold is 4, we still needs to estimate the pattern local frequency by using the sketch.

This motivates us to store more information for the local region to speedup the verification. Based on the spatial order, the space is partitioned into a set of small grids $g = \{g_1, g_2, ...\}$. The general idea

is to store the weight for nodes and edges in each local grid where they appear . For each node t_i, we store a set of tuples $\langle w(t_i, g_i), h_{g_i}^{t_i} \rangle$, where $w(t_i, g_i)$ is the number of appearances for t_i in g_i, and $h_{g_i}^{t_i}$ is the largest hash value for t_i in g_i. For the edges between two nodes t_i, t_j, we store a set of tuples $\langle s(e_{ij}, g_i) \cdot w(e_{ij}, g_i), s(e_{ji}, g_i) \cdot w(e_{ij}, g_i), h_{g_i}^{e_{ij}} \rangle$, where $w(e_{ij}, g_i)$ is the number of co-occurrence of t_i, t_j in g_i when they are successive keywords, $h_{g_i}^{e_{ij}}$ equals $\min\{h_{g_i}^{t_i}, h_{g_i}^{t_j}\}$ and $s(e_{ij}, g_i)$ equals -1 if $h_{g_i}^{e_{ij}}$ appears in both $\mathcal{L}(t_i)$ and $\mathcal{L}(t_j)$, otherwise $s(e_{ij}, g_i)$ equals 1. Similar definition goes for $w(e_{ji}, g_i)$ and $s(e_{ji}, g_i)$.

Example 8. *As shown in Figure 1, for keyword t_1, we store $\langle 2, 0.2 \rangle, \langle 1, 0.25 \rangle$ since t_1 appears in two grids with weight 2 and 1; for the edge between t_1 and t_2, we store $\langle$-1, 0, 0.2$\rangle, \langle$-1, 0, 0.25$\rangle$, since t_1 and t_2 are successive keywords and have one co-occurrence in both grids. In addition, 0.2 and 0.25 appear in both keywords' sketch, thus $s(e_{12}, g_i)$ equals -1. Then to find the intersection size of two keywords, we can directly check the edges information.*

Lower Bound and Upper Bound of Patterns. Given a query region R, let $g_{up}(R)$ be the set of minimum number of grids that covers R, and let $g_{low}(R)$ be the set of maximum number of grids than covered by R. Then we merge the statistics in $g_{up}(R)$ and $g_{low}(R)$ to estimate the lower bound and upper bound of a pattern's frequency in the query region. The estimation method is similar to that we estimate the pattern frequency upper bound using the sketch graph. Then two pruning and verification rules come directly: 1) if the lower bound of the pattern frequency is larger than the given threshold, we can claim it is frequent without further verification, 2) if the upper bound of the pattern frequency is less than the threshold, we can discard the pattern immediately.

Example 9. *The query region is R_2 in Figure 1. Then $g_{low}(R_2) = \{g_2, g_4\}$. Suppose we want to estimate the frequency lower bound of pattern $\{t_4, t_5, t_6\}$. The sum of the weight of edge e_{64} is 4 in $g_{low}(R_2)$, and t_5 is the only nodes which ranks between t_4 and t_6. Thus the pattern appears in the samples at less 4 times in $g_{low}(R_2)$. Based on the hash value stored, we can estimate the lower bound as $(4-1)/0.55 = 5.5$.*

Discussion. We discuss some details in the real implementation for enhanced sketch graph. In the enhanced sketch graph, for each edge, we may need to twice space to store the co-occurrence information, since there can be two edges between a pair of keywords at most. For the spatial information, we do not need to store additional information for nodes, since in IK sketch, we have index each list based on the spatial order, thus we can quickly retrieve the information of nodes, *i.e.,* the weight and hash value in each grid. For edges, we only store the spatial information, when the two keywords are frequent than a predefined threshold. Because when then keywords are not frequent, we can get the spatial information by quick sketch intersection. Moreover, since many co-occurrence keywords have the local property, such as "Opera House", that is, it will only have co-occurrence in limited number of grids. Consequently, the number of tuples maintained for each edge is not large and it will not result large cost in space.

4.3 Enhanced Sketch Graph based LFP Identification

In this section, we introduce a depth-first order based algorithm for processing LFP query. As we have stated, there are some patterns of large size in the microblogs. In this case, the Apriori based method will enumerate many intermediate patterns.

Example 10. *Suppose we have a local frequent pattern $\{t_1, t_2, t_3, t_4\}$. Based on the Apriori framework, we first find $\{t_1\}, \{t_2\}, \{t_3\}, \{t_4\}$ are frequent; next we find all the patterns of size two are frequent; then we find all the patterns of size three are frequent and finally we identify $\{t_1, t_2, t_3, t_4\}$ is frequent. Thus a lot of intermediate results are enumerated. Following the depth-first order, we will find patterns $\{t_1\}$, $\{t_1, t_2\}$, $\{t_1, t_2, t_3\}$ and $\{t_1, t_2, t_3, t_4\}$ are frequent in order. Even though we will enumerate some other patterns in the follows, since we have already find $\{t_1, t_2, t_3, t_4\}$ is frequent, these patterns will have no checking cost. Thus, the depth-first order based search can save a lot of cost when the pattern size is large.*

Suppose there is a global order among all the keywords based on the keyword subscripts. Given a keyword t_i and a pattern P, we have $t_i < P$ if $\exists t_j \in P$ that $i < j$, and we have $t_i > P$ if $i > j$ for each $t_j \in P$. The general idea of the algorithm is that for a pattern P, we maintain two lists I, X for it. I, X store all the keywords t_i for P that P is still local frequent if we add t_i into P. The difference is that for $t_i \in I$ we have $t_i > P$; for $t_j \in X$ we have $t_i < P$. Each time we select a node from I added to P and update I, X. When both lists are empty, we get a LFP; if X is not empty, then P must be a sub-pattern in the previous found ones. Algorithm 3 introduces the details of the algorithm

Algorithm 3: LFP Identification based on Enhanced Sketch Graph

Input : G_e : enhanced sketch graph; $\mathcal{L}$: IK sketch; θ : frequency threshold; R : query region.
Output: LFPs in R.
1 $I \leftarrow$ frequent keywords in R ;
2 $P, X \leftarrow \phi$;
3 sort I based on keyword order ;
4 **Enum**(P, I, X) ;

In Algorithm 4 Line 4 and 5, there are two functions $GetNext_I(I, v)$ and $GetNext_X(X, v)$. These two functions are used to refine the list I and X when we add node v to the current pattern. In the refinement, it keeps I and X with the satisfied nodes, and it uses the enhanced sketch graph to accelerate the pruning of non-qualified nodes. Since we always add nodes to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

Algorithm 4: Enum(P, I, X)

Input : P : a frequent pattern; I, X : candidate nodes list.

1 **if** $I = \phi$ *and* $X = \phi$ **then**
2 **return** P
3 **for each** $v \in I$ **do**
4 $I' \leftarrow GetNext_I(I, v)$;
5 $X' \leftarrow GetNext_X(X, v)$;
6 **Enum**$(P \cup \{v\}, I', X')$;
7 $I \leftarrow I \setminus \{v\}$;

the pattern with larger order, if in the final I is empty and X is not empty, it means the found pattern must be a sub-pattern in the previous found patterns. Thus, we only return a pattern when I and X are empty as shown in Algorithm 4 Line 1.

Example 11. *Following Example 10, we start the enumeration from keyword t_1. When current pattern is $\{t_1\}$, $I = \{t_2, t_3, t_4\}$ and $X = \{\}$. After we get the pattern $\{t_1, t_2, t_3, t_4\}$, I and X are both empty, so the pattern is returned. Then we back track to the pattern $\{t_1, t_2\}$, and $I = \{t_3, t_4\}$ and $X = \{\}$. After adding t_4 into the pattern, we have $I = \{\}$ and $X = \{t_3\}$, because $t_3 < \{t_1, t_2, t_4\}$. Then the pattern will not be returned.*

Discussion. For Algorithm 3 in the worst case, we need to enumerate all the LFP. Thus, the worst case performance is the same as the IK sketch based approach.

5 LKFP Identification

To answer LKFP query, naively, we can maintain a top-K min-heap and follow the Apriori framework utilizing IK sketch. We continuously update the frequency threshold during enumeration process until we find top-K local maximal patterns. This algorithm, denoted as IKK, serves as the baseline method for answering LKFP query based on IK sketch. However, this approach will enumerate many intermediate patterns that may not be included in final top-K results. So in this section, we develop an efficient algorithm to answer LKFP query based on the property of top-K query using IK sketch and enhanced sketch graph.

Motivation. To reduce the number of patterns enumerated, a critical point is to reduce the keyword space for enumeration and set a proper stop condition. Let f_{dk}^R denote the k-th distinct frequency of keywords in the query region R. For example, suppose the frequency of keywords in region R is $\{t_1 : 5,\ t_2 : 5,\ t_3 : 3,\ t_4 : 2\}$, then $f_{d2}^R = 3$, since t_1 and t_2 have the same frequency. Given a region R, the set of *top-dK* frequent keywords $\mathcal{V}_{dK}^R$ in R is defined as $\{t \mid t \in R \ \wedge \ f(t, R) \leq f_{dK}^R\}$. In order to answer LKFP query, Theorem 1 suggests that we only need to enumerate over the *top-dK* frequent keywords in the query region. Consequently, we can gradually expand over the *top-dK* keywords for enumeration. Following the previous example, the *top-$d2$* frequent keywords are $\{t_1, t_2, t_3\}$. To find *top*-2 local maximal patterns, we only need to enumerate over $\{t_1, t_2, t_3\}$, since the final *top*-2 results may be $\{t_1, t_2\}$ and $\{t_3\}$. Algorithm 5 describes the details of LKFP identification algorithm.

Theorem 1. *Given a query region R and the set of top-dK frequent keywords in the query region $\mathcal{V}_{dK}^R$. To answer LKFP query, we only need to enumerate patterns over $\mathcal{V}_{dK}^R$.*

Proof: We prove the correctness by contradiction. Given a query region R and top-K K, suppose we find the result set of LKFP query, and one of the result patterns P includes a keyword t_1 which is not in *top-dK* frequent keywords, i.e., $t_1 \in P$ and $t_1 \notin \mathcal{V}_{dK}^R$. Then the frequency of this pattern must be smaller than the frequency of pattern t_1 based on the Apriori property, i.e., $f(P, R) \leq f(t_1, R)$. It means that there must be a keyword t_2 belongs to the *top-dK*, but it is not included in any of result patterns. However, t_2 must belongs to a maximal pattern which frequency is not smaller than $f(t_2, R)$, otherwise t_2 itself will become a maximal pattern according to the maximal definition. We have $f(t_2, R) > f(P, R)$, hence pattern $\{t_2\}$ must be returned instead of P. This contradicts our assumption. □

Algorithm 5: LKFP Identification

Input : G : enhanced sketch graph, $\mathcal{L}$: IK sketch, K : top-K, R : query region.
Output: LKFP

1 **for** $i \in [1, K]$ **do** $I_i \leftarrow \mathcal{V}_{di}^R / \mathcal{V}_{d(i-1)}^R$;
2 $H, C, X, I \leftarrow \phi$;
3 **for each** $i \in [1, K]$ **do**
4 $I \leftarrow \bigcup_1^i I_j$;
5 **for** $t \in I_i$ **do**
6 $S_P \leftarrow Enum'(t, I, X)$;
7 $I \leftarrow I \setminus \{t\}$;
8 insert (P_i, I, X) into C, H **for** $P_i \in S_p$;
9 **while** $C \neq \phi$ and $C.top \geq H.top$ **do**
10 $(P, I, X) \leftarrow C.pop$;
11 **for each** $t \in I \cap N(P)$ **do**
12 $P_t \leftarrow P \cup t$;
13 $S_P \leftarrow Enum'(P_t, I \cap N(P_t), X \cap N(P_t))$;
14 update C, H ;
15 **return** H

Algorithm. Algorithm 5 follows the best first framework to find the top-K results. H is a min heap which stores the current top-K results, and C is a max heap which stores the candidate patterns. Obviously, if $C = \phi$ or $C.top < H.top$, we can terminate the algorithm and return H as shown in Line 9. In Line 1, the keywords with frequency f_{di}^R are firstly identified and stored in I_i, with $\mathcal{V}_{d0}^R = \phi$. By setting each keyword $t \in I_i$ as the initial enumerated keyword, we find a set of maximal patterns with frequency equals $f(t_i, R)$ in Line 3 to 8. According to Lemma 1, to generate a maximal pattern with the first keyword as $t \in I_i$, we only to enumerate over $\bigcup_1^i I_j$. The found patterns S_P in Line 6 are inserted into H as initial top-K results, and they are also inserted into

candidate heap C. The function $Enum'$ is similar to Algorithm 4. The difference is that in the algorithm $GetNext_I$ and $GetNext_X$ return the set of candidate keywords which will not decrease the frequency of current pattern; it will enumerate only if the pattern frequency is not smaller than $H.top$, when $|H| = K$. The keywords are ordered based on the inverse order of frequency, which means we only attempt to add a more frequent keywords into current pattern. In the second step, for each candidate pattern P in C, we enlarge it with 1 candidate keyword in Line 12, and continue the enumeration and find a maximal pattern. If the found patterns frequency are smaller than $H.top$ they will be discard directly; otherwise the found patterns are used to update the heap H and inserted into candidate heap C.

Example 12. *For the simplicity of explaining the algorithm, suppose we sample all the data. Thus the estimate frequency is the exact frequency. Given $K = 3$, the local data are shown like this: $t_1 : 10$, $t_2 : 10$, $\{t_1, t_2\} : 5$, $t_3 : 4$, $t_4 : 3$. Thus $I_1 = \{t_1, t_2\}$, $I_2 = \{t_3\}$ and $I_3 = \{t_4\}$. Based on the algorithm, we first enumerate a set of patterns from I_i with frequency not less than itself. From I_1 we will enumerate patterns $\{t_1\}$ and $\{t_2\}$ with I equals $\{t_2\}$ and $\{t_1\}$ respectively. From I_2, I_3, we get patterns $\{t_3\}$, $\{t_4\}$. So in heap H, we have $\{t_1\}$, $\{t_2\}$, $\{t_3\}$, and in candidate heap C we have all the four patterns. In the next round, we pop $\{t_1\}$. After expanding the pattern, we get $\{t_1, t_2\}$. Since the frequency of $\{t_1, t_2\}$ is larger than t_3, we update H and insert it into C. Then, we will pop $\{t_2\}$ and $\{t_1, t_2\}$ for pattern expansion. Since they do not need to expand anymore, we just discard it. Then the top in C is $\{t_3\}$ which frequency is less than that of $\{t_1, t_2\}$ (top in H). Then we stop the algorithm.*

Discussion. Suppose the average sketch length in local is b. For the keywords in I_i, it will try to enumerate the pattern with keywords in $\cup_1^i I_i$. Thus in the worst case, there will be $2^{|\cup_1^K I_i|}$ patterns enumerated. Then the time complexity is bounded by $O(bK2^{|\cup_1^K I_i|})$.

6 Experiment

In this section, we present the results of a comprehensive performance study to evaluate the efficiency and effectiveness of the proposed techniques.

6.1 Experiment Setup

Algorithms. As there is no previous work for the problem of local (top-K) frequent patterns identification over geo-tagged microblog stream, we use a uniform sampling based approach as the baseline method. The algorithms investigated in the experiments are listed as follows.

- **UNF/UNK.** Algorithms to answer LFP/LKFP query based on uniform sample method and Apriori framework.
- **IKF/IKK.** Algorithms to answer LFP/LKFP query based on IK sketch and sketch graph proposed in this paper.
- **IKGF/IKGK.** Algorithms for LFP/LKFP query based on IK sketch and enhanced sketch graph proposed in this paper.
- **IKGFS/IKGKS.** Algorithms for LFP/LKFP query based on IK sketch and enhanced sketch graph proposed in this paper and enforce the space to be the same as IKF/IKK by reducing the sample size.

Dataset. One real Twitter dataset from [30] is employed in the empirical study, which is collected from September 2012 to February 2013. For the effectiveness of the query processing, we clean the dataset by removing the stop words, links, special symbols, usernames and words with length less or equal than two. We also remove the tweets without words after cleaning. Finally, we have more than 300 million tweets with average tweets length equals to 5.7. The tweets are collected using Twitter API from the whole world, so the coordinator are from -90 to 90 and -180 to 180 corresponding to the latitude and longitude respectively.

Workload. In the experiment, we only evaluate the sliding window model which is more interesting to users in real applications. The slide window size n varies from 1 million to 9 million, with 1 million as default value. The query region is a rectangle whose center is randomly selected from the locations of objects underlying the whole space. The query region size varies from 1% to 15% in term to the ratio of whole space with 5% as default value. The sample size l is defined in term of ratio of sliding windows size, which varies from 0.1% to 10% with 5% as default value. For LFP query, the query frequency threshold varies from 1000 to 5000 with 2000 as default value. For LKFP query, the query K varies from 1 to 100 with 10 as default value. For each problem, we generate the 500 queries, and the average performance is reported. Since the query regions are randomly selected, they may have overlapping in space.

To evaluate the effectiveness of proposed techniques, precision and recall are reported for LFP queries. The precision is calculated as the number of true LFPs in the returned results dividing the number of patterns returned. The recall is calculated as the number of true LFPs in the returned results dividing the total number of true LFPs. For LKFP queries, recall is reported by varying different parameters. The recall for LKFP queries is calculated as the number of true LKFPs in the returned results dividing K. To measure the efficiency of the algorithm, the response time and sketch update time is reported.

All algorithms are implemented in C++ with GNU GCC 4.8.2 with -O3 flag. Experiments are conducted on a PC with Intel Xeon 3.4GHz CPU and 32G memory using Redhat Linux.

6.2 Effectiveness Evaluation

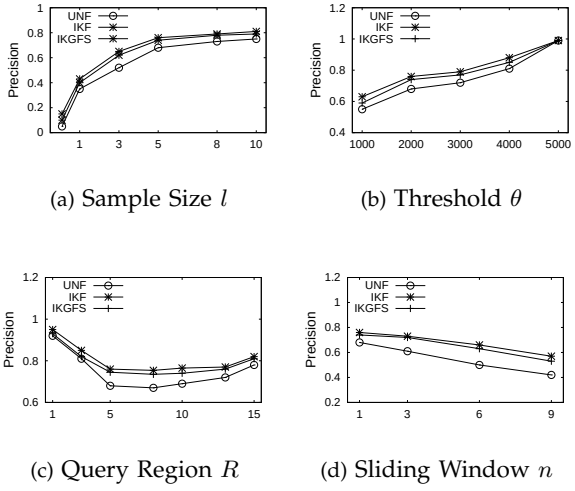
In this section, we present the effectiveness evaluation of the algorithms.

Effectiveness Examples. We list five patterns we found in the dataset as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering

- Pattern "high school" in Raleigh. It is the first day of a local high. Some students sent tweets about the day, and some people sent tweets about seeing many high school students.
- Pattern "just became mayor" in Bandung. The pattern is frequent because of the popularity of Foursquare app. In the app, if you checkin in a venue more frequent than others, you just became the mayor of the venue. In addition, Bandung is a tourism city, which will attract more people to checkin.
- Pattern "international airport" in Bali, where many people leave and arrive Bali international airport as a tourism. For such a beautiful place, many people sent tweets about the sadness of leaving and happiness of arriving.
- Pattern "falling temperature rain today humidity" in Auckland, which is a news about the changing weather. Thus many accounts and people sent tweets or retweeted about the news.
- Pattern "civil structural engineer jobs" in Colorado, which is an advertisement or spam about the new job.

As we can see, the found patterns fulfil the classification as we mentioned. For local interesting news, the pattern size tends to be small, since the core event can be described in a few words. When it comes to some national news, like the weather report, people just retweets or send tweets with the same content as the official account, since it is hard to change the content. For advertisements or spams, it also tends to be of long size, since the tweets may be just sent by some robot accounts.



(a) Sample Size l (b) Threshold θ

(c) Query Region R (d) Sliding Window n

Fig. 3. Precision of LFP Query

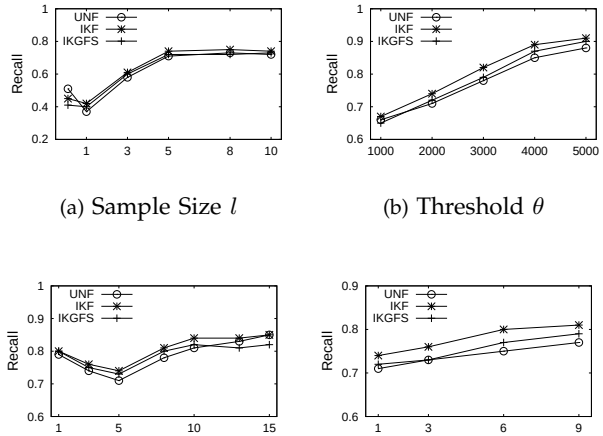
Effectiveness of LFP Query. Since enhanced sketch graph only improves the efficiency for query processing, we report the performance of UNF, IKF and IKGFS. To evaluate the quality of our approximate approaches, Figure 3 and Figure 4 depicts the precision and recall for answering LFP queries by varying different parameters. As we can see, the IKGFS method have only slight drop in the precision and recall compared with IKF method. This is because, in the enhanced sketch graph, we only allocate more information for the frequent edges in the real implementation, thus the overhead is not much and IKF and IKGFS are close in sample size. Compared with the uniform sampling approach, the effectiveness of proposed methods are better. This is because bottom-k is more suitable for set intersection size estimation. In Figure 3(d), when the sliding window size increases, the precision drops. This is because the frequency threshold is not changed, then the number of distinct keywords increases with the sliding window size, and the number of patterns increases too. Based on union bound, we may need more samples to get a more accurate estimation. However, when the query size enlarge, the trend is not the same as sliding window as shown in Figure 3(c). This is because, the enlarge of spatial region may not include more data or distinct keywords. For example, there are much more tweets in downtown than in suburbs. Even if we have enlarge the space to include the suburbs, the data size will not change a lot. But by increasing the sliding window size, the number of distinct keywords grows much faster.

Effectiveness of LKFP query. Similarly, we report the performance of IKK, UNK and IKGKS. In Figure 5, we report the performance by varying different parameters. As we can see, the performance of IKGKS slightly drops compared with IKK. By varying the sample size l and top-K K, the performance of proposed methods are close to UNK, however, they significantly outperform UNK in efficiency as shown later. As shown in Figure 5(a), by varying the sample size, similar trends can be observed as in LFP query evaluation. Varying the top-K K, we report the performance in in Figure 5(b). As shown, the recall decreases with the increase of K, because for larger K, we need more samples in order to bound the correct order of patterns. In Figure 5(c), the recall of all methods increases by varying the region size from 1% to 15%. For LKFP query, the performance is not that sensitive to sliding window size as LFP query. This is because, the enumeration space is much smaller than that in LFP query.

6.3 Efficiency Evaluation

To evaluate the efficiency of proposed sketch structure and different algorithms, we examine the query response time and the sketch update time. In Figure 6 and 7, we report the query response time for LFP and LKFP queries by varying different parameters. By increasing the sample size, query region size and sliding window size, all the methods' response time increases, since we have more data or samples to process. The proposed methods significantly outperform the baseline approach, because of the novel data structure introduced. As shown in Figure 6(d), the sliding window size has great impact on the response time for LFP query, this is because, the number of data grows quickly with the sliding window size,

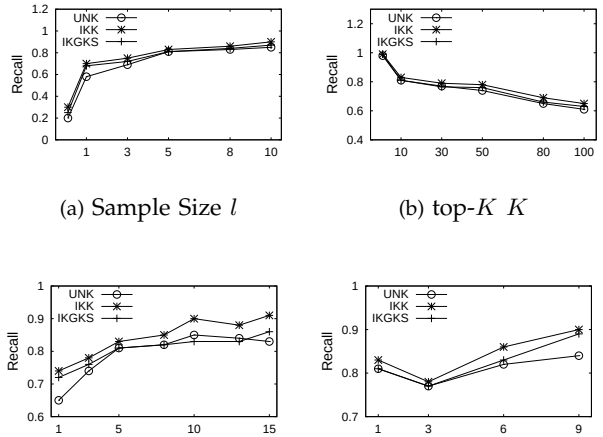
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2578330, IEEE Transactions on Knowledge and Data Engineering



(a) Sample Size l (b) Threshold θ

(c) Query Region R (d) Sliding Window n

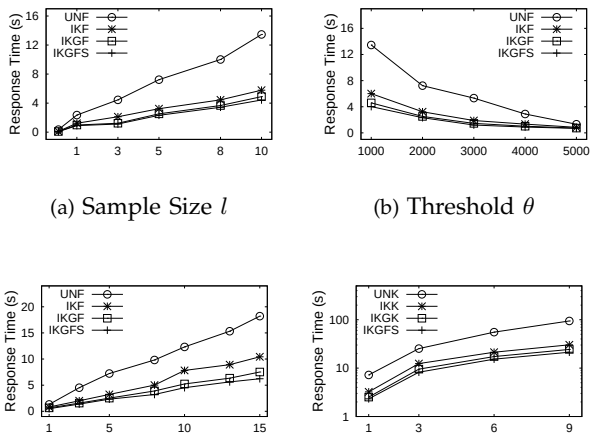
Fig. 4. Recall of LFP Query



(a) Sample Size l (b) top-K K

(c) Query Region R (d) Sliding Window n

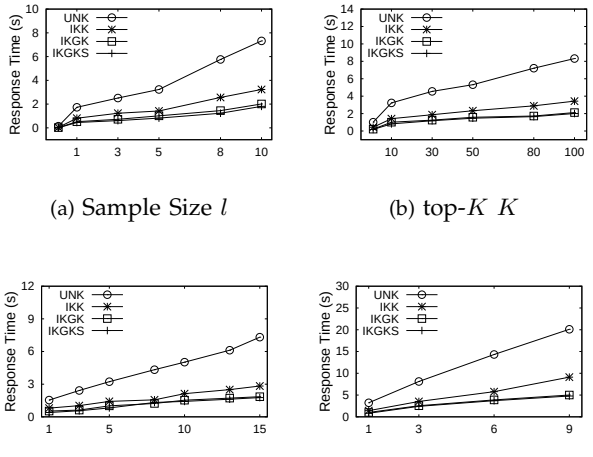
Fig. 5. Recall of LKFP Query



(a) Sample Size l (b) Threshold θ

(c) Query Region R (d) Sliding Window n

Fig. 6. Response Time of LFP Query



(a) Sample Size l (b) top-K K

(c) Query Region R (d) Sliding Window n

Fig. 7. Response Time of LKFP Query

and we do not change the frequency threshold θ, which will result a large number of patterns. Most of these patterns have no help for event analysis, such as "right now". This is the case that we have not select a good threshold, which leads to long processing time and many patterns of no help. It is also the reason we introduce top-K query in this paper.

We also evaluate the update time of IK sketch based approach and enhanced sketch graph based approach by changing the space budget. It takes $42.6ms$ and $113.5ms$ when sample ratio varies from 0.1% to 10% for IK sketch based approach. It takes $63.2ms$ and $151.9ms$ respectively for enhanced sketch graph based approach, since it needs to maintain the information in grid.

7 Conclusion

In this paper, we investigate the problem of identifying local maximal frequent keyword co-occurrence patterns over geo-tagged microblog streams. Specifically, we consider two models, threshold based frequent pattern (LFP) model and top-K based frequent pattern (LKFP) model. Novel sketch techniques (IK sketch and enhance sketch graph) are proposed to summarize the most recent geo-tagged microblog stream. Efficient pattern identification algorithms are proposed. Moveover, theoretical analysis is presented for the space requirement to answer the problems accurately with summary. In the final, we conduct an empirical study to demonstrate the efficiency and effectiveness of our approaches.

References

[1] K. Watanabe, M. Ochi, M. Okabe, and R. Onai, "Jasmine: A real-time local-event detection system based on geolocation information propagated to microblogs," in *CIKM*, 2011, pp. 2541–2544.

[2] F. Alvanaki, S. Michel, K. Ramamritham, and G. Weikum, "See what's enblogue: Real-time emergent topic identification in social media," in *EDBT*, 2012, pp. 336–347.

[3] H. Abdelhaq and M. Gertz, "On the locality of keywords in twitter streams," in *IWGS*, 2014, pp. 12–20.

[4] T. Lappas, M. R. Vieira, D. Gunopulos, and V. J. Tsotras, "On the spatiotemporal burstiness of terms," *PVLDB*, vol. 5, no. 9, 2012.

[5] C. Budak, T. Georgiou, D. Agrawal, and A. El Abbadi, "Geoscope: Online detection of geo-correlated information trends in social networks," *PVLDB*, vol. 7, no. 4, 2013.

[6] A. Skovsgaard, D. Sidlauskas, and C. S. Jensen, "Scalable top-k spatio-temporal term querying," in *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, 2014, pp. 148–159.

[7] M. Mathioudakis and N. Koudas, "Twittermonitor: trend detection over the twitter stream," in *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, 2010, pp. 1155–1158.

[8] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: Online localized event detection from twitter," *PVLDB*, vol. 6, no. 12, 2013.

[9] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, "Moment: Maintaining closed frequent itemsets over a stream sliding window," in *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK*, 2004, pp. 59–66.

[10] R. C. Wong and A. W. Fu, "Mining top-*K* frequent itemsets from data streams," *Data Min. Knowl. Discov.*, vol. 13, no. 2, pp. 193–217, 2006.

[11] V. T. Chakaravarthy, V. Pandit, and Y. Sabharwal, "Analysis of sampling techniques for association rule mining," in *Database Theory - ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, 2009, pp. 276–283.

[12] K. Watanabe, M. Ochi, M. Okabe, and R. Onai, "Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs," in *CIKM*, 2011.

[13] E. Schubert, M. Weiler, and H. Kriegel, "Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds," in *KDD*, 2014, pp. 871–880.

[14] S. Unankard, X. Li, and M. A. Sharaf, "Emerging event detection in social networks with location sensitivity," *World Wide Web*, vol. 18, no. 5, pp. 1393–1417, 2015.

[15] H. Liu, Y. Lin, and J. Han, "Methods for mining frequent items in data streams: an overview," *Knowl. Inf. Syst.*, vol. 26, no. 1, 2011.

[16] J. Han, H. Cheng, D. Xin, and X. Yan, "Frequent pattern mining: current status and future directions," *Data Min. Knowl. Discov.*, vol. 15, no. 1, 2007.

[17] E. Cohen, "Size-estimation framework with applications to transitive closure and reachability," *J. Comput. Syst. Sci.*, vol. 55, no. 3, 1997.

[18] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla, "On synopses for distinct-value estimation under multiset operations," in *SIGMOD Conference*, 2007.

[19] E. Cohen and H. Kaplan, "Summarizing data using bottom-k sketches," in *PODC*, 2007.

[20] X. Wang, Y. Zhang, W. Zhang, X. Lin, and W. Wang, "Selectivity estimation on streaming spatio-textual data using local correlations," *PVLDB*, vol. 8, no. 2, pp. 101–112, 2014.

[21] D. Takuma and H. Yanagisawa, "Faster upper bounding of intersection sizes," in *SIGIR*, 2013.

[22] D. Okanohara and Y. Yoshida, "Conjunctive filter: Breaking the entropy barrier," in *ALENEX*, 2010.

[23] K. S. Beyer, P. J. Haas, B. Reinwald, Y. Sismanis, and R. Gemulla, "On synopses for distinct-value estimation under multiset operations," in *SIGMOD Conference*, 2007.

[24] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," *Technical Report, Ottawa, Canada: IBM Ltd*, 1966.

[25] W. Li, "Random texts exhibit zipf's-law-like word frequency distribution," *IEEE Transactions on Information Theory*, vol. 38, no. 6, pp. 1842–1845, 1992.

[26] E. Cohen and H. Kaplan, "Tighter estimation using bottom k sketches," *PVLDB*, vol. 1, no. 1, 2008.

[27] D. Dubhashi and D. Ranjan, "Balls and bins: A study in negative dependence," *Random Structures and Algorithms*, vol. 13, pp. 99–124, 1996.

[28] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava, "Hashed samples: selectivity estimators for set similarity selection queries," *PVLDB*, vol. 1, no. 1, 2008.

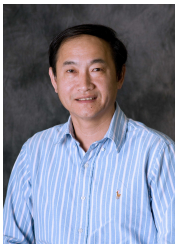
[29] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 41–82, 2005.

[30] J. Jiang, H. Lu, B. Yang, and B. Cui, "Finding top-k local users in geo-tagged social media data," in *ICDE*, 2015, pp. 267–278.

Xiaoyang Wang is a research associate in the University of Technology, Sydney. He received his BSc and MSc degrees in Computer Science from Northeastern University, China in 2010 and 2012 respectively, and PhD degree from the University of New South Wales, Australia in 2016. His research interests include query processing on massive spaital data stream.

Ying Zhang is a senior lectuer and ARC DECRA research fellow (2014-2016) at QCIS, the University of Technology, Sydney (UTS). He received his BSc and MSc degrees in Computer Science from Peking University, and PhD in Computer Science from the University of New South Wales. His research interests include query processing on data stream, uncertain data and graphs. He was an Australian Research Council Australian Postdoctoral Fellowship (ARC APD) holder (2010-2013).

Wenjie Zhang is currently a senior lecturer and ARC DECRA (Australian Research Council Discovery Early Career Researcher Award) Fellow in School of Computer Science and Engineering, the University of New South Wales, Australia. She received PhD in computer science and engineering in 2010 from the University of New South Wales. Since 2008, she has published more than 20 papers in SIGMOD, VLDB, ICDE, TODS, TKDE and VLDBJ. She is the recipient of Best (Student) Paper Award of National DataBase Conference of China 2006, APWebWAIM 2009, Australasian Database Conference 2010 and DASFAA 2012, and also co-authored one of the best papers in ICDE2010, ICDE 2012 and DASFAA 2012. In 2011, she received the Australian Research Council Discovery Early Career Researcher Award.



Xuemin Lin is a Scientia Professor in the School of Computer Science and Engineering, the University of New South Wales. He has been the head of database research group at UNSW since 2002. He is a concurrent professor in the School of Software, East China Normal University. Before joining UNSW, Xuemin held various academic positions at the University of Queensland and the University of Western Australia. Dr. Lin got his PhD in Computer Science from the University of Queensland in 1992 and his BSc in Applied Math from Fudan University in 1984. During 1984-1988, he studied for Ph.D. in Applied Math at Fudan University. He currently is an associate editor of ACM Transactions on Database Systems. He is a senior member of IEEE. His current research interests lie in data streams, approximate query processing, spatial data analysis, and graph visualization.