Provenance-based Indexing Support in Micro-blog Platforms

Junjie Yao Bin Cui Zijun Xue Qingyun Liu
Department of Computer Science and Technology
Key Laboratory of High Confidence Software Technologies (Ministry of Education)
Peking University, P.R.China
{junjie.yao,bin.cui}@pku.edu.cn
{xuezijunpkueecs,liuqingyun.pku}@gmail.com

***Abstract*— Recently, lots of micro-blog message sharing applications have emerged on the web. Users can publish short messages freely and get notified by the subscriptions instantly. Prominent examples include Twitter, Facebook's statuses, and Sina Weibo in China. The Micro-blog platform becomes a useful service for real time information creation and propagation. However, these messages' short length and dynamic characters have posed great challenges for effective content understanding. Additionally, the noise and fragments make it difficult to discover the temporal propagation trail to explore development of micro-blog messages.**

In this paper, we propose a provenance model to capture connections between micro-blog messages. Provenance refers to data origin identification and transformation logging, demonstrating of great value in recent database and workflow systems. To cope with the real time micro-message deluge, we utilize a novel message grouping approach to encode and maintain the provenance information. Furthermore, we adopt a summary index and several adaptive pruning strategies to implement efficient provenance updating. Based on the index, our provenance solution can support rich query retrieval and intuitive message tracking for effective message organization. Experiments conducted on a real dataset verify the effectiveness and efficiency of our approach.

I. Introduction

With the development of social media sites and ubiquitous smart mobile devices, lots of micro-blog sharing applications have emerged on the web. For example, status features are common in social community sites(Facebook, Google Plus), and prominent micro-blog statuses are pervasive in Twitter, and Sina Weibo in China[1]. Micro-blog services enable users to publish and receive statuses in an intuitive and easy way. They can then comment or re-share these posts. Just at the age of five, Twitter now has 230 million tweets a day from more than 100 million users [1]. The massive social activities and user interests are well reflected in this torrent of messages. Recently, researchers have conducted some analytic work on a variety of directions in this social lens. For example, social science, stock market predication and user sentiment monitoring in [2], [3].

Micro-blog messages are short and temporal. In rapid changing scenarios, lots of events appear and soon are replaced by other newly emerging topics. The noise and short snippets are also inevitable. In Figure 1, we present a micro-blog message searching example, where several Twitter messages related to a baseball game between Yankee and Redsox [2] are returned. We can find some well written messages and also several emotional phrases and short noise. Users can re-share and comment the previous messages, i.e., last item (original previous message staring with 'RT').

Search

You searched for 'yankee redsox'

User	Post Time	Content
ariherzog_OLD	2009-09-30 02:28:22	Can't believe those #redsox. Argh!
dims	2009-09-30 01:18:11	#redsox sigh!
abcdude	2009-09-26 00:23:58	Classy. Way it should be RT @AmalieBenjamin: Lester getting an ovation from the #Yankee Stadium crowd as he gets to his feet. #redsox
wharman	2009-09-26 00:18:57	Lester down #redsox
BaldPunk	2009-09-17 03:44:20	#Redsox - glee ! - I put up awesome NY Yankee Stadium photos - Yankees - MLB - http://bit.ly/Uvcpr
dims	2009-09-17 03:19:03	unbelievable!! #redsox
stevebrownell	2009-09-17 02:56:26	ugh #redsox

Fig. 1. Common Micro-blog Message Search

The traditional keyword search approach may fail in micro-blog scenario due to several reasons. First, only 140 characters are allowed in each message, which sets obstacles for users to easily understand these short messages. More severe issue is that the instant publishing and open broadcasting characters of micro-blog make it vulnerable to noisy and low quality messages. Several short emotional messages appear in Figure 1. Second, micro-blog messages are easy to propagate. The re-sharing and diffusion behaviors are common on micro-blog services. Breaking news can reach a large number of audience in a short time. But at the same time, the development and

[1]http://facebook.com, http://plus.google.com, http://twitter.com, http://weibo.com

[2]http://t.pku.edu.cn/tweet/search/ or http://[2001:da8:201:1203::138]/tweet (IPV6)

changes during it may split. It becomes a difficult task for users to effectively understand micro-blog messages and grasp the context of their topical themes.

In a recent study, the authors found that breaking events and famous stars were popular on Twitter messages and users always monitor these events by repeated searches [4]. This differs from traditional web search behaviors and makes the micro-blog search more challenging. Better content organization of these short messages and more intuitive exploration methods are thus necessary.

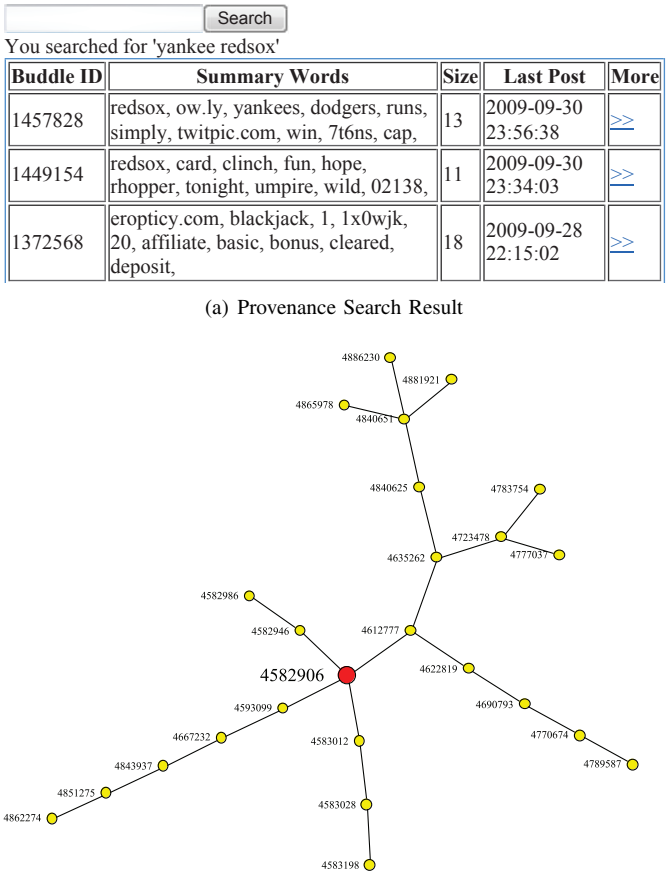
In this paper, we propose a provenance based indexing approach to explore and manage the micro-blog messages. Provenance discovery [5], [6] is an important technique to derive the source and transformation from large amounts of data. Provenance information describes the origin and the development of data in their life cycles. It has been demonstrated useful in many domains, such as business workflow, scientific processing and database query analysis. For example, information about provenance can serve as the basis of data results correctness and in turn, determine the quality of the final outputs. Here we introduce a provenance model over micro-blog messages to capture the messages' development. The development can be captured by means of temporal grouping and alignment of related messages, i.e., a kind of connection discovery to explore the temporal context of messages. Here, related messages are extracted and organized in an ordered structure. The latter incoming messages are connected with previous related ones.

Shown in Figure 2, we depict a search case of provenance model, where groups of relevant messages are returned as result items [3]. In each returned item, we have some related messages (top frequent words are shown as the summary) and especially preserve the connections between them (i.e., the lower part in this figure). These connections contain the topical or temporal relevance, such as re-sharing and common topics, which can reveal the development of dynamic micro-blog streams. Through this rich structure and the extracted summarization , the meaning of returned items are more vivid and the temporal development of each item is easy to grasp.

The provenance discovery can bring a variety of advantages to current micro-blog platforms:

- **Rich Context Summarization:** With the help of provenance discovery, we are able to extract the related messages and previous context. These grouped messages are usually of great use in extracting the context and underlying themes. This is beneficial for message understanding.
- **Development and Feedback Exploration:** The rapid developing events are significant parts in micro-blog messages. Usually users find it difficult to fully understand the background and development of events. Through provenance discovery, we can extract the information cascading path and temporal development trails, which are intuitive for storyline exploration and development visualization. The user comments in provenance paths are high quality materials to augment the understanding of ongoing micro-blog messages.
- **Quality Identification:** Noise exists in micro-blog services. Through the sources, developments and user feedbacks collected from provenance discovery, users can better distinguish the credibility of information. The collective intelligence existing in rich feedback and comments leverage the content quality assessment.

[3]http://t.pku.edu.cn/tweet/prov/



(a) Provenance Search Result

(b) Provenance Visualization, ID:1505316

Fig. 2. A Provenance Supported Micro-blog Message Search

Compared with previous provenance scenarios, the micro-blog messages have some characters, making it challenging to discover and maintain provenance information. First, the information travels from arbitrary messages to a large audience. Users can comment on or re-share the original messages. It is unfeasible to maintain the processing nodes explicitly, which is fairly larger than static or limited processing units in traditional provenance scenarios. Second, in a breaking event, multiple sources or recurring ones exist to discuss this topic. Many users publish messages simultaneously. It is therefore necessary to maintain the source identification. Third, operations of information diffusion in micro-blog scenario are complex. All the comment, re-share, feedbacks and others can diffuse and propagate messages. We need harness them to infer the

underlying provenance trails. At last, the messages in micro-blog services are always in a raging torrent. In order to cope with it, efficient processing methods with low overhead are required.

In this paper, following the proposed provenance concept model, we design a practical indexing scheme to extract and maintain the provenance information. The introduced indexing module acts as an additional engine besides the common micro-blog message retrieval counterpart. It receives incoming messages and updates their connections with prior items. We propose a message grouping strategy to preserve the provenance information. To efficiently cope with the message stream, we adopt several efficient pruning methods to reduce the maintenance overhead.

The contributions of this paper are listed as follows.

- We introduce a novel micro-blog provenance model. It embeds the rich interactions of micro-blog messages into a provenance discovery framework. The representation method and structure meaning of it are also concerned. This provides a new organization viewpoint of dynamic micro-blog messages.
- An efficient provenance indexing method for extraction and maintenance is utilized. We design a summary index to encode the provenance information. And we further adopt the provenance search method based on the proposed index. How new features and evidences from this kind of grouped messages will improve the micro-blog search is also discussed.
- Extensive experimental studies on real dataset are conducted. The results verify the effectiveness and efficiency over high volume message torrent. The applicability over other social media data is also promising.

The remainder of this paper is organized as follows. In Section II, we review the related literature. Then we introduce the notions in Section III. In Section IV, we provide details of provenance discovery and index management solutions. The provenance maintenance and query support are revealed in Section V. Experiments are then reported in Section VI. At last, we conclude this paper.

II. Related Work

This work is broadly related to some research areas. Here we review them below:

Provenance Information Discovery: Provenance describes the origin and development of data in their processing circle. It has many applications in the data management tasks. The curated databases and workflow systems employ provenance to tract the source and transformation of data results [7], [5]. Recent years have witnessed the progress of different provenance concept models, extraction and maintenance methods [8], [9], [10], [11]. In social media domain, information streams have active participation and frequent updates. But the complex development and high volume characters prohibit traditional provenance methods [12], [13]. In this paper, we propose the use of micro-blog message's development trails and a well designed summary index structure to aid provenance discovery.

Dynamics and Diffusion in Social Media: The emerging of recent social media data changes the way of content creation and consumption. Lots of users generate and distribute the online content. Real world events are also remarkably reflected in user's online content generation. To analyze and understand them, burst event detection and information cascading mining have become hot topics recently [1], [14].

In [15], the authors investigated the phrase spreading over blog networks and found it had comparable scope and better quality over traditional news media agents. In [16], a two-phrase information spreading model was analyzed on a large scale Twitter dataset. A recent user case study found users in social media always kept re-search queries in Twitter to understand the development of news events [4]. However, most of these analysis and mining work are placed in an off-line mode. In this paper we focus on the practical indexing support for online dynamic monitoring and latter query/discovery tasks. The provenance information collection and indexing approach used in this paper could facilitate further dynamics analysis and content enrichment.

Micro-blog Data Management: Micro-blog messages are generally short and noisy, which is a remarkable character of recent social media content. In [17], Chen et al. presented a partial indexing design to immediately classify the incoming tweet content into high quality and noisy ones. The former category is indexed in real time and the latter one in a batch way. Other common approaches include [18], [19], where topic model methods are used to to extract theme concepts. The expertise of message authors, the feedbacks and broadcasting scope of a message are helpful to separate high quality content from tweet streams [20]. However, these methods have drawbacks. Context and evolution of dynamic messages are missing. The item based ranking or simple grouping methods are incapable of trail discovery. The provenance indexing proposed in this paper can explicitly represent the structure connections. We can better utilize them to organize the content and enrich the information discovery for latter analysis and querying support.

III. Preliminaries

In this section, we introduce the provenance concepts used in micro-blog messages.

Table I lists some messages talking about baseball game between "Yankee and Redsox", which has been mentioned in Figure 1 and 2. There are not only simple text messages, but also some ones associated with annotated indicants. For example, we find RT, $hashtag$ or URL. $Hashtag$ (started with '#') and URL point out the references or topics of the message. RT is the action of re-sharing a previous message. It outputs a new message based on the previous one by adding some comment or skipping comment with 'RT' ahead of the previous text. For example, message posted by user 'abcdude' is a re-sharing one , derived from user AmalieBenjamin's post.

TABLE I

EXAMPLES OF MICRO-BLOG MESSAGES

User	Date	Message Text
bren924	2009-09-26 01:06:11	WHEW!! RT @MLB: RT @IanMBrowne X-rays on Lester negative. Contusion of the right quad. Day to Day. #redsox
abcdude	2009-09-26 00:23:58	Classy. Way it should be RT @AmalieBenjamin: Lester getting an ovation from the #Yankee Stadium crowd as he gets to his feet. #redsox
TonyStarks40	2009-07-21 01:49:05	Yankee Magic, you can only find it at Yankee Stadium! THE YANKEEEEEEEESS WIN!!!

Continuing micro-blog messages form a stream. We define the concept of message stream as follows:

Definition 1 (Micro-blog Message Stream): A $Micro-blogMessageStream$ is a line of micro-blog messages t_1, t_2, ..., t_i, t_n ordered by the published date, in which each message t_i is represented as a multi-field tuple $[date, user, msg, urls, hashtags, rt]$. They are categorized as date, user message text(user, msg) and the annotated indicants(url, hashtag or RT) respectively.

Micro-blog messages inside the stream are not isolated. They connect with others through explicit RT relation or implicit topic related links. Given two micro-blog messages t_i and t_j, and t_j posted later than t_i, connection types between them can be categorized in Table II.

TABLE II

DIFFERENT CONNECTIONS BETWEEN MICRO-BLOG MESSAGES

Type	Condition
RT	t_j re-shares t_j
URL	$url(t_j) \bigcap url(t_i) \neq \emptyset$
hashtag	$hashtag(t_j) \bigcap hashtag(t_i) \neq \emptyset$
text	$text(t_j) \bigcap text(t_i) \neq \emptyset$

The connections between micro-blog messages represent a kind of continuing information development. RT either directly re-shares the previous message or at the same time, adds some comments. The common URLs, hashtags and words used can measure the similar topics they convey.

Harnessing the connections between these messages can benefit in the discussed context understanding and development trail extraction. Recently there have been some research work in message connection behavior extraction [16], [21], however, these previous studies are isolated and rather limited. Here we combine multiple connection indicants and propose a general provenance model to represent these connections.

Data Provenance has two broad categories of view, i.e., the source provenance for data origin finding and transformation provenance for development trail monitoring [5]. We use the latter line to model message connections and temporal interactions in micro-blog services.

Definition 2 (Micro-blog Message Provenance): In micro-blog message streams, we connect new messages with prior ones through some defined connections. Connection scores between two messages t_i and t_j are derived from RT, hashtag , url or word similarities. Provenance preserves the message development trail in this way.

To extract the provenance information, most of the existing techniques are classified by their choices to provenance recording. One line focuses on computing provenance information when data are created (*annotation based, eager*), while the other does so when the provenance information is requested (*non-annotation based, lazy*) [6]. Distinct characters of micro-blog message provenance are the high volume of stream and the variety of connection types. We regard each new incoming message as a fresh unit and thus we have no pre-defined processing nodes. The non-annotation approach is intolerable due to the high volume. On the other hand, if we maintain each message's former message connections, the complete annotation approach is also costly. An efficient method is required to connect new messages with prior ones.

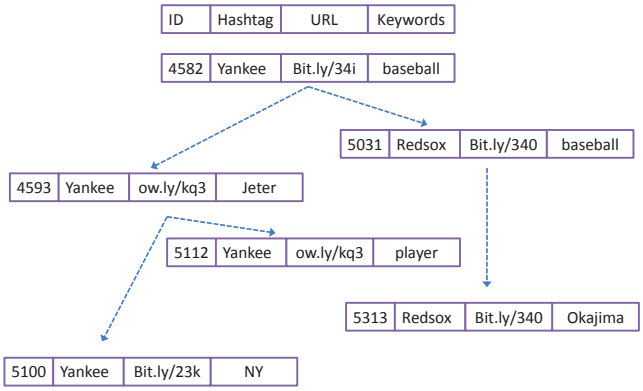


Fig. 3. Provenance Bundle of Micro-blog Messages

To speed up the micro-blog message provenance discovery, we resort to selecting appropriate granularity levels to represent the provenance information. We propose two strategies: first, the provenance information is represented in some local message groups; second, one message only retains a maximum scored connection with its prior similar one. That is to say, we form the messages in a directed graph similar to Figure 2. We group related messages together to get a set of non-overlapping groups and only retain the message connection inside the bundle. Message connections beyond the bundle are filtered. The provenance bundle definition is introduced as follows:

Definition 3 (Provenance Bundle): Bundle is a group of messages $T = \{t_1, t_2, \ldots, t_i, \ldots, t_n\}$, where message connections within group are preserved and intra-group connections are skipped. The connection between t_i and t_j is aggregated from the previous mentioned connection types shown in Table II.

The bundle is a kind of coarse representation for prove-

nance information collection and discovery. The provenance information is preserved within the bundle, in the form of message connections. We also extract the summary of bundles to facilitate the new message insertion. In Figure 3, we display a case of provenance bundle. Messages are connected with previous ones and placed in a compact tree structure. As shown in this figure, we can find the extracted hashtags and web page URLs for the $Yankee\&Redsox$ related message group. They are extracted to support the connection finding and the bundle summarization.

To implement provenance support in micro-blog messages, the most important component is effective finding and preserving provenance bundles from incoming micro-blog message streams. At the same time, an efficient solution is required to cope with high volume challenge. In the next two subsequent sections, we will provide details of provenance indexing maintenance and manipulating operations. The introduced bundle is also used as a basic retrieval unit for micro-blog message search. The relatedness and ranking features to evaluate the similarity between messages will be discussed later.

IV. Provenance-based Indexing

This section presents methods used in provenance information discovery. We first outline the framework for provenance summary index construction and bundle information updating. Then we present a indexing structure to support efficient bundle updating and later retrieval operations. The detailed algorithms for bundle processing in new message routing and allocation are discussed.

A. System Framework

Given the aforementioned concepts, we depict our provenance-based indexing framework in Figure 4. It is divided into two parts, i.e., the on-disk storage back-end and in-memory processing unit. The in-memory unit has a summary index to manage bundle collections and a bundle pool to maintain fresh bundles. The back-end bundle storage is used to keep finished bundles that no longer receive updates. The provenance indexing framework acts as an additional processing engine, independent of common text based micro-blog indexing and retrieval systems.

In a practical micro-blog provenance discovery system. Incoming message torrents and a variety of connection types make it challenging. Here the bundle summary index retains enough indicants to aid the new messages routing and efficiently placing them into appropriate bundles. If found, the new messages are inserted into the most appropriate one. If no such bundle exist, new bundle will be created. Bundles are retained in memory for fast match and insertion. After some iterations of new message processing, some old bundles will be eliminated or flushed back into disk.

B. Provenance Summary Index

Here we discuss the details of summary index in provenance discovery. Discussed in Section III, each bundle extracts some connection indicants for new message match and insertion.

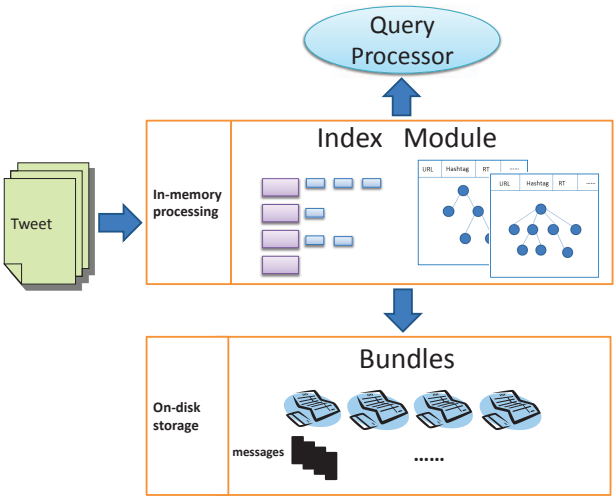


Fig. 4. Framework of Provenance-based Indexing

The summary index is collected from indicant summaries of bundles.

Figure 5 illustrates a diagram of summary index. In the shown example, some keywords and hashtags, i.e., “redsox”, “yankee” and URL are extracted and inserted in the summary index. The summary index has top level keys with the bundle indicants, such as hashtags, URLs and keywords. More system specific fields can also be included, like the RT information or manually defined message groups. The indexed items are associated with underlying bundles. In each key’s index field, bundles having the indicants are enumerated.

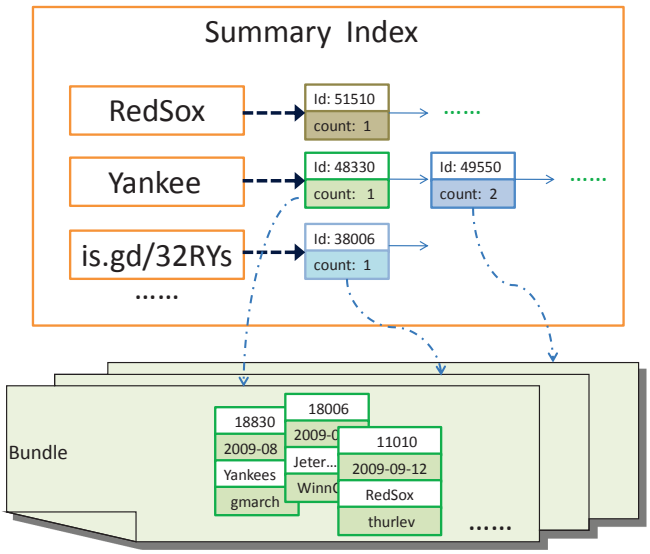


Fig. 5. Summary Index of Micro-blog Bundles

C. Message Match with Bundles

Given a new micro-blog message, the summary index first extract the connection indicants from new message, i.e., hash-

tag, URL, keywords or RT. Then the indicants' corresponding bundles in the summary index are selected. For each hashtag, URL or keyword found from the new message, we fetch an associated candidate bundle list from summary index. We then measure the bundle's relevance based on connection similarity and other factors, such as bundle size and bundle timeliness.

Algorithm 1 New Message Match with Bundle

Input: new incoming message t
Summary Index I

1: {Step 1. fetch candidate bundles }
2: **for** $indicant \in t$ **do**
3: **for** $item \in indicant$ **do**
4: $bundle\_list \leftarrow I_{indicant}[item]$
5: **for** $id \in bundle\_list.keys()$ **do**
6: $candidate\_list.append(id)$
7: **end for**
8: **end for**
9: **end for**
10: {Step 2. select bundle, insert message}
11: $bundle$ = select_max_score($candidate\_list, t$)
12: **if** $bundle$ is $null$ **then**
13: $bundle \leftarrow$ new bundle.
14: $bundle \leftarrow t$
15: update index.
16: **else**
17: $bundle \leftarrow t$
18: **end if**
19: {Step 3. update summary index }
20: **for** $indicant \in t$ **do**
21: **for** $item \in indicant$ **do**
22: $I_{indicant}[item] \leftarrow t$
23: **end for**
24: **end for**
25: **return**

The message match process is sketched in Algorithm 1. This process is divided into three stages. They are, candidate bundles fetching (Line 1-9), bundle scoring and insertion(Line $11-18$), and the final index updating (Line $20-24$). When a new message is inserted into a bundle, the affected indexed items in the summary index will be updated. If no such item exists in the summary index, a new index item will be added.

In the above Algorithm 1, we wrap a scoring function, $select\_max\_score$, which is used to rank the best suitable bundle for a new message. It evaluates the relevance between the new coming message and a candidate bundle. Equation 1 shows how some factors are combined in this function.

$$S(t,B) = \alpha|url(t) \cap url(B)| + \beta|tag(t) \cap tag(B)| + \gamma|date(t) - date(B)| + \dots \quad (1)$$

where B represents a candidate bundle and t is the incoming micro-blog message. The function aggregates some indicant closeness measuring functions. Here we only present the URL, hashtag and date as examples. We combine the overlap of URLs and hashtags of the new message and the candidate bundle. The time difference between them is also taken into consideration. The intuition behind is, under similar overlapping conditions of URLs and hashtags, a fresh bundle is more suitable to match with. The α, β and γ are parameters to tune the weight, which can be manually set to reflect system requirements.

Our index structure differs from traditional inverted index in the following aspects. First, we have feature selection flexibility and weighting variety. Besides the common hashtag and URL, other closeness indicants can also be freely incorporated into this summary index for versatile match implementation. For example, the selected keywords from user or automatic assignment and other message grouping methods. Second, bundle's timeliness has been considered. As a non-indexed factor, this can be incorporated after the candidate bundle fetching.

D. Message Allocation within the Bundle

After selecting a bundle to match the new message, inserting the message into the suitable position inside this bundle is also not trivial. Since there are usually several message nodes inside a bundle, choosing the suitable one to align the new message requires similarity and other metrics.

Shown in Figure 3, a new message needs to compare its content and date closeness with existing members of the bundle. Here we design a solution with multiple steps: collecting each message's match information; calculating the similarity in the message candidate set; selecting the most similar one and finally placing the new message. The operations are summarized in Algorithm 2.

Algorithm 2 Message Allocation inside the Bundle

Input: chosen bundle B
new message to insert t

1: **for** $msg \in B.msg\_list$ **do**
2: **if** $t \bigcap msg \neq \emptyset$ **then**
3: $candidate\_list \leftarrow msg$
4: **end if**
5: **end for**
6: $top\_msg = get\_max(candidate\_list)$
7: connect top_msg with t
8: **if** $t.date > B.end\_time$ **then**
9: $B.end\_time$ = $t.date$
10: **end if**
11: **if** $t.date < B.start\_time$ **then**
12: $B.start\_time$ = $t.date$
13: **end if**
14: **return**

In the placing operations, we measure the similarity between messages with the intersection operation on the connection indicators (i.e., hashtag and URLs). The calculation procedure is a weighted combination of similarities from different indicators. In the Equation 5, we aggregate the similarities from the URL and hashtag set scores.

We use function $U(t_i, t_j)$ to measure the percentage of shared URLs between message t_i and t_j.

$$U(t_i, t_j) = \frac{|url(t_i) \cap url(t_j)|}{|url(t_i)|} \tag{2}$$

Similarly, function $H(t_i, t_j)$ to indicate the percentage of shared hashtags between message t_i and t_j;

$$H(t_i, t_j) = \frac{|tag(t_i) \cap tag(t_j)|}{|tag(t_i)|} \tag{3}$$

Another factor considered is the time closeness of two messages. It is intuitive to match the new message with fresh messages. We use a function T to measure this factor. A simple strategy is the inversion of time span between two messages, shown below:

$$T(t_i, t_j) = \frac{1}{|date(t_i) - date(t_j)| + 1} \tag{4}$$

where $date(t_i)$ represents the post time of message t_i.

We then integrate the above factors with weighting parameters into a equation to measure the similarity of two messages.

$$S(t_i, t_j) = \alpha U(t_i, t_j) + \beta H(t_i, t_j) + \gamma T(t_i, t_j) \tag{5}$$

where α, β, γ are tuning parameters to weight the importance of different factors.

V. Provenance Maintenance and Retrieval Support

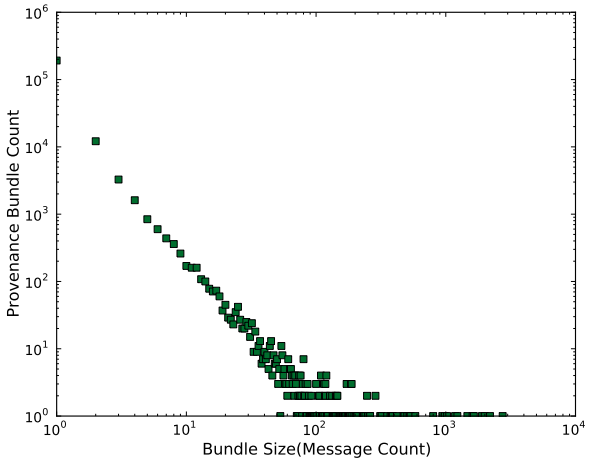
Given more and more incoming micro-blog messages, we generate a growing number of provenance bundles in memory. Feasible cleaning and backup is necessary to meet the stability requirement of provenance discovery. In this section, we come up with several maintenance solutions of summary index module. These extensions are build upon the aforementioned provenance discovery module.

We first discuss a statistics study on the provenance bundles and then introduce some maintenance approaches. Based on the summary index, the query support is also touched on at the end of this section.

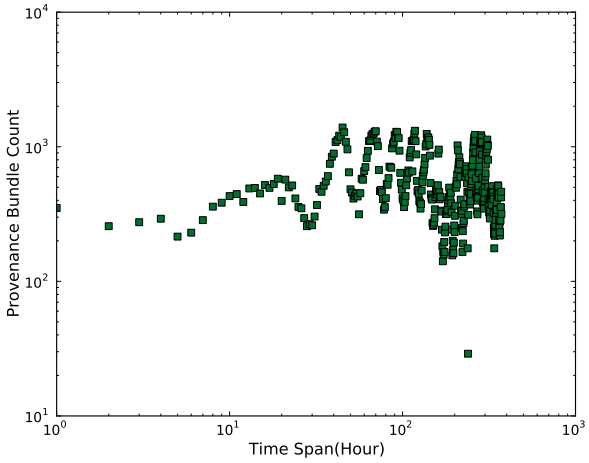
A. Characters of Provenance Bundle

After the processing of bundle discovery and message match/allocation algorithms discussed in Section IV, we analyze the size and time span of the generated bundles. In a simulation experiment on some real micro-blog messages, we bulk about 700k messages into the memory and compare the provenance bundles. The experimental setup and dataset description are described Section VI. We do not set any restriction of the bundle size and message match here. From this stream of micro-blog messages, we found about 30k bundles.

Figure 6 depicts the bundle size distribution and the time span of these bundles. The bundle size is measured by the number of messages inside each bundle. And the time span indicates last update time of each bundle. We find that: a remarkable proportion of the bundle sets are in small size,



(a) Bundle Size without Any Limitation



(b) Distribution of Time Span

Fig. 6. Provenance Bundle Characters

and also a significant number has median count of messages. Only a small proportion of these bundles are large. Most of the bundles no longer get updating after some time.

These characters of the provenance bundle probably reveal the temporal dynamics of micro-blog messages. Besides some breaking or hot events, represented by large bundles, most of bundles are about common temporal topics and in a controllable size. We can utilize them to design our optimization strategies in provenance maintenance. We select old bundles from the bundle pool in memory and backup them onto disk. At the same time, we refine the bundles in memory through filtering small and old ones.

These operations can reduce the size of summary index overhead and at the same time avoid the severe decline of provenance discovery accuracy.

B. Bundle Pool Constraint

Here we describe the details of our optimization methods. We periodically check the memory status to monitor the provenance bundle consumption. In these checks, when the bundle number in the memory exceeds a pre-defined threshold, an refinement procedure will be invoked. It scans the bundles and eliminates those with less chance to be updated, i.e., receive new messages. The corresponding index items will also be updated. In these eliminated bundles, tiny old ones are directed deleted and those median bundles are backup onto disk.

We set a refining condition to achieve this goal. Bundles with messages published earlier than a predefined time span, and at the same time the bundle size smaller than a threshold, will be recognized as aging tiny one and be eliminated directly. After that, if the size of bundle pool is still at a high level, we will rank the remaining bundles and continue the elimination until the bundle pool size is smaller than a defined bound.

The metrics of this ranking combines both the bundles' time and size information. To rank the updating probability of bundles, we define a function G as follows:

$$G(B) = curr - date(B) + 1/|B| \tag{6}$$

where B is the bundle evaluated by function G. $date(B)$ is latest updating time of messages in bundle B. $curr$ is the current time. $|B|$ represents the bundle's size. For any bundle B, the higher function $G(B)$ scores, the more likely it will not be updated in the future and the more possible we eliminate it in a refinement operation. The remaining bundles will be sorted by the ranking score in a descending order. Then we fetch bundles from the top and delete them until the bundle pool size is smaller than a defined bound.

Besides these two maintenance rules, we also adopt the bundle size constraint to avoid huge group of messages. In provenance bundle pools, sometimes the incoming messages talking a hot event can make a bundle grow into a huge group. The materialization and analysis will stumble on these big bundles. For bundles exceeding a size threshold, we do not insert new messages into them and mark the closed state of them. We control the size of our bundle by set a maximum bundle size. If a bundle reaches the threshold, we will make it an isolated one. In our system, we set a $closed$ tag for such bundles to indicate whether new tweets can be inserted in. In next bundle pool scan, they will be backup onto disk.

We summarize the above bundle pool maintenance operations in Algorithm 3. The refining process can be divided into two stages. The first stage(Line $1-11$) is eliminating the aging and tiny bundles. The bundle which is both aging and tiny will be deleted directly and the deleted bundle counter $count$ will be updated. Other bundles will be inserted into a candidate queue and evaluated by function G. In the second stage(Line $12-17$), the candidate queue is sorted based on the items' scores of function G in descending order. Then the lower bound of eliminating number will be checked. If it is not fulfilled, the item at the head of the queue will be eliminated until the lower bound is reached.

Algorithm 3 Refinement Process for Bundle Pool

Input: Bundle Pool P
Limitation of Bundle Pool Size M
Current Date $curr$
Bundle Refining Size R
Bundle Refine Time T

1: $count = 0$
2: **for** $b \in P$ **do**
3: **if** $curr - date(b) > T \wedge |b| < R$ **then**
4: $B.delete\_index(b)$
5: $B.delete(b)$
6: $count+ = 1$
7: **else if** $curr - date(b) > T \wedge status(b) == closed$ **then**
8: dump b to disk
9: **else**
10: $score = G(b)$
11: $refine\_waiting\_list \leftarrow ((b, score))$
12: **end if**
13: **end for**
14: $refine\_waiting\_list\_list.sort()$
15: **while** $count < refine\_lower\_limit$ **do**
16: $b \leftarrow$ pop $refine\_waiting\_list$
17: $P.delete\_index(b)$
18: $P.delete(b)$
19: $count + = 1$
20: **end while**
21: **return**

In order to assure the efficiency of our provenance index system, the bundle pool refinement will not be carried out until there are enough bundles. We set a lower bound for the number of bundles to invoke the checking procedure. This mechanism avoids the problem of frequent bundle scanning operation.

C. Bundle Retrieval

Important usage of summary index support is the bundle based micro-blog message search. Given a user input query q, summary index returns a list of relevant bundles as the search result, instead of previous simple message list.

We can represent a bundle with a group of micro-blog messages and compare its textual relevance with input query. Further more, the rich structure and indicants preserved in provenance bundle can also enrich the retrieval experience and ranking quality. In Equation 7, the overall relevance function between input query r and candidate bundle B, $r(q, B)$ can be aggregated from several aspects.

$$r(q, B) = \alpha * s(q, B) + \beta * i(q, B) + (1 - \alpha - \beta) * t(B) \tag{7}$$

where α, β are tuning parameters between 0 and 1.

Here, we combine the common textual similarity used in retrieval (the function $s(*, *)$) and the indicant closeness from summary index (the function $i(*, *)$). The third function $t(*)$

represents the freshness, which also attributes to the evaluation function.

To present the performance of bundle based query, we set up a demo site [4] to display the bundles found. The input query will return a list of bundles and each bundle shows the top indicators(Hashtags, URLs). The visualized tree structure of provenance bundle can also be found. Snapshots are demonstrated in Figure 2 and cases in experimental study.

VI. Experimental Study

In this section, we report the empirical study conducted on real micro-blog dataset. We test the performance of bundle index and the viability of provenance discovery.

A. Dataset and Experiment Setup

According to a recent report of the latest micro-blog platform development, in Oct 2011, more than 230 million messages are posted on Twitter every day [1]. Here in the experiment, we use a real dataset to simulate this message stream. The collecting process had been proceeded before October 2009 through Twitter's API and has more than 25 million messages [22], [17]. We select a two-month period of messages in this dataset, August and September in 2009. Each day has about 70k messages. Without explicit mention, all the following experiments are conducted on this subset.

We conduct the experiments on an x64 Linux server with eight core CPUs and 32G memory. Codes are developed with Python and Django [5]. We implement the query support using Lucene [6].

The simulation experiment runs as follows. We import the micro-blog messages into the system in a temporally ordered sequence. The latest message's date is simulated as the system's current date. To test the performance of provenance extraction methods, we compare three different implementations:

- **Full Index:** We do not limit summary index and bundle pool size. This method is a baseline for provenance discovery. Derived from our proposed provenance model, the message connections extracted in this method are treated as the ground truth for further evaluation of the approximation methods we used.
- **Partial Index:** Here we use the index refinement algorithm discussed in Section V-B without the bundle size constraint. We filter out the bundles from the memory, based on each bundle's freshness and size.
- **Partial Index, Bundle Limit:** Besides the bundle pool limitation caused by partial index, here we also restrict the bundle size. Later in the experiment figures, we use its abbreviation as 'Bundle Limit'.

[4]http://t.pku.edu.cn/tweet/prov/ or http://[2001:da8:201:1203::138]/tweet/prov/ (IPV6)

[5]http://www.python.org, http://www.djangoproject.com

[6]http://lucene.apache.org

B. Performance of Provenance Discovery

Figure 7 displays the growth of in-memory bundles under different approaches. The baseline method has a linear growth. In the other two partial index approaches, the bundle pool limitation used is 10k. With the restriction of bundle pool size and the freshness, we find a sharp decline of bundle pool size at the initial stage. After then, the periodical refinement operations restrain the bundle count at a low level. In the third method, the additional provenance bundle size limitation leads to a slight increase of bundle count in memory.

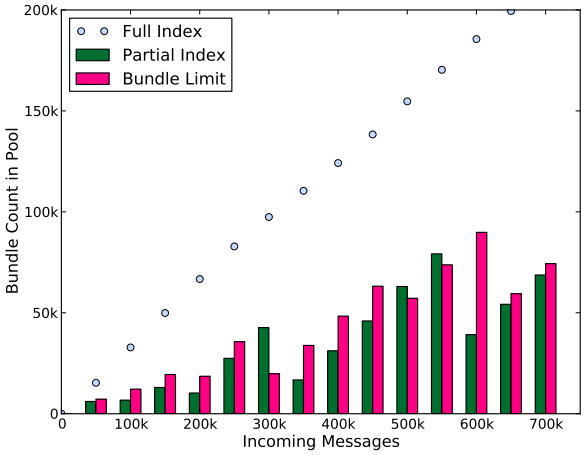


Fig. 7. Provenance Bundle Growth under Different Approaches

The basic operation in provenance discovery is the connection alignment. Given an incoming micro-blog message, we find a suitable bundle, insert the message into this bundle and then align the message with a priori message. To test the accuracy of provenance discovery, we compare the outputs of message connections from each approach. In the baseline full index method, provenance information used is complete and the provenance bundle discovered is valid. We use its output as ground truth and compare the performance of the two other methods.

In the experiment, at each date check point, we collect all message connections from each method's bundles as their corresponding output results. E_0, E_1, E_2 represent these message edge set respectively. Take the example of partial index approach(method 2), the provenance discovery performance can be measured as:

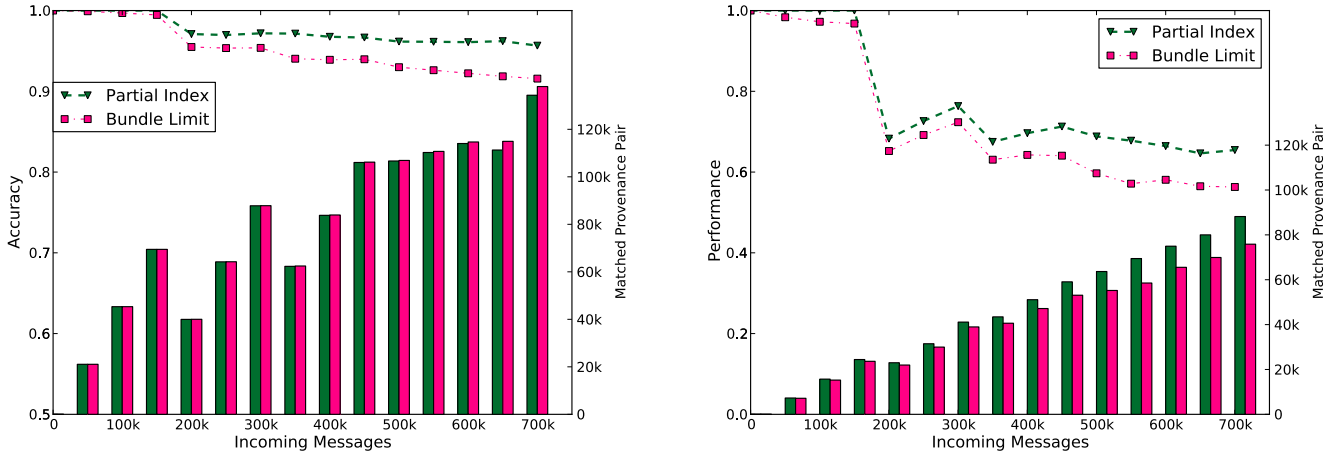
$$accu_1 = |E_1 \cap E_0|/|E_1|$$

It measures the proportion of the correct connections found by method 2.

Another facet of the performance is

$$ret_1 = |E_1 \cap E_0|/|E_0|$$

It reveals how much provenance information the new proposed partial approach can cover.



(a) Accuracy (b) Return

Fig. 8. Different Provenance Index Methods

Performance of two method variants are displayed in Figure 8 respectively. We also add two bars in the lower part of the figures, which means the absolute count of valid edges found in these two methods.

In this experiment, the partial index has a comparable advantage over the bundle limit method. Both these two methods has a slight performance decline compared to baseline ground truth. The added bundle size limitation split some edge connections, which will be preserved in non bundle size limitation counterpart method.

The figures also demonstrate that the bundle distribution statistics of micro-blog messages are useful for provenance maintenance. The partial index strategy skips the isolated bundles which have little tendency to absorb new messages.

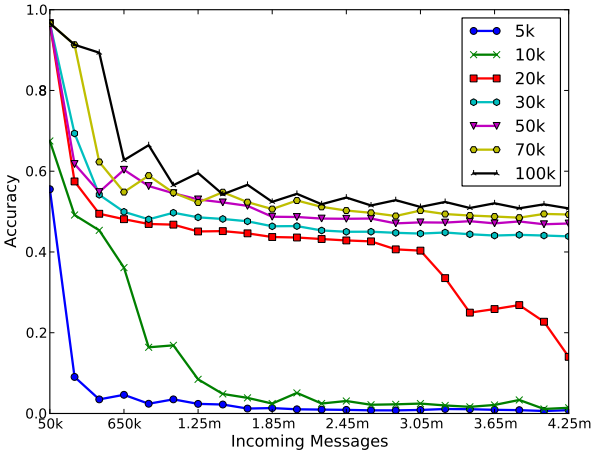


Fig. 9. Accuracy Change under Different Parameters in Partial Index

In Figure 9, we systematically test the parameter choices under a bigger dataset , spanning over two month period, which has about 4.25 million messages. We set the bundle pool limitation, ranging from 5k to 100k. Under different settings, we compare their accuracy outputs of provenance discovery. This result indicates that small bundle pool gets unacceptable accuracy while bigger bundle pools larger than 20k are usually stable over the processing period. The feasible bundle pool size reveals the covering of current micro-blog messages.

C. Showing Cases of Discovered Provenance

In Figure 10, we display two examples of extracted provenance bundles. Both of these two events occurred in Sept 2009. The left one is IBM's CICS partner conference and the right one talks about a tsunami occurring at Sumatra, Indonesia [7] Red nodes are first messages respectively, and the following messages are connected through the provenance connection we discovered. These paths reveal the information propagation and development trails.

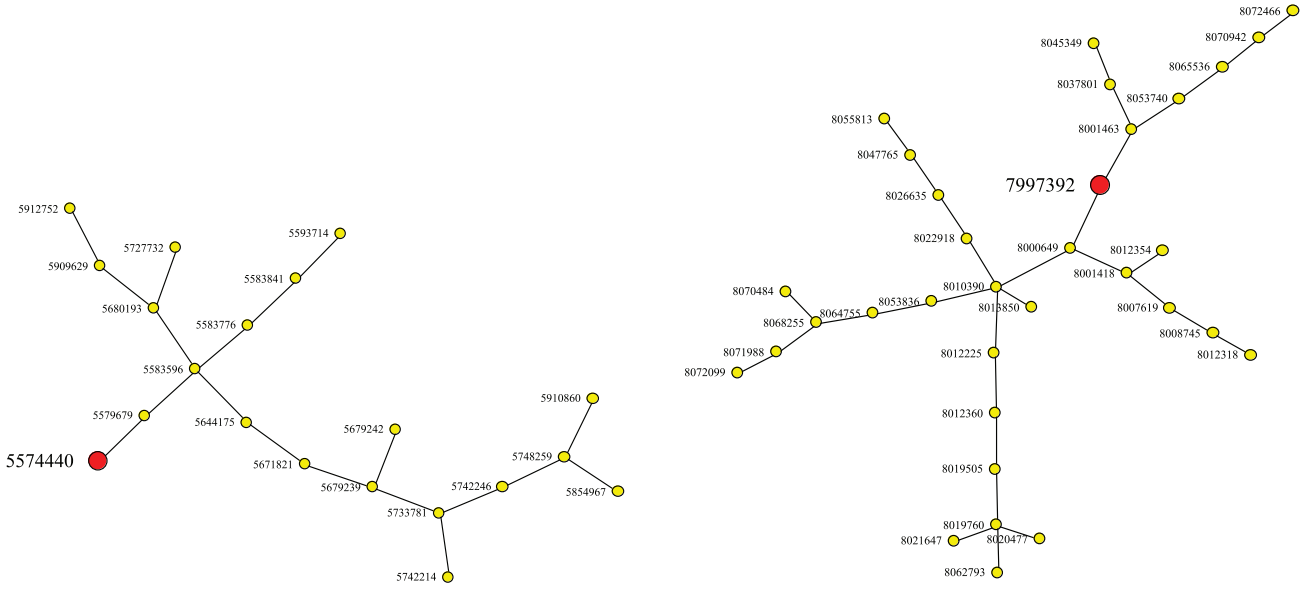
D. Memory Overhead and Time Cost

In Figure 11, we show the memory cost of different index approaches. The left figure displays the actual memory usage and right one shows message counts.

Inevitably, the simple method without limitation has a greedy memory usage increase with the coming of new messages. The other two partial index methods have usage at a steady level. They have comparable advantages over the baseline method. That is, 10M v.s. 170M. To measure this memory metric independently of hardware configuration, we also include the message count in these provenance bundles. Similar results are revealed.

Next we present the time cost of different methods. With the growth of incoming messages, these three approaches all exhibit a linear time cost increase, which are shown in Figure 12. This is stable in most of actual settings. With the low

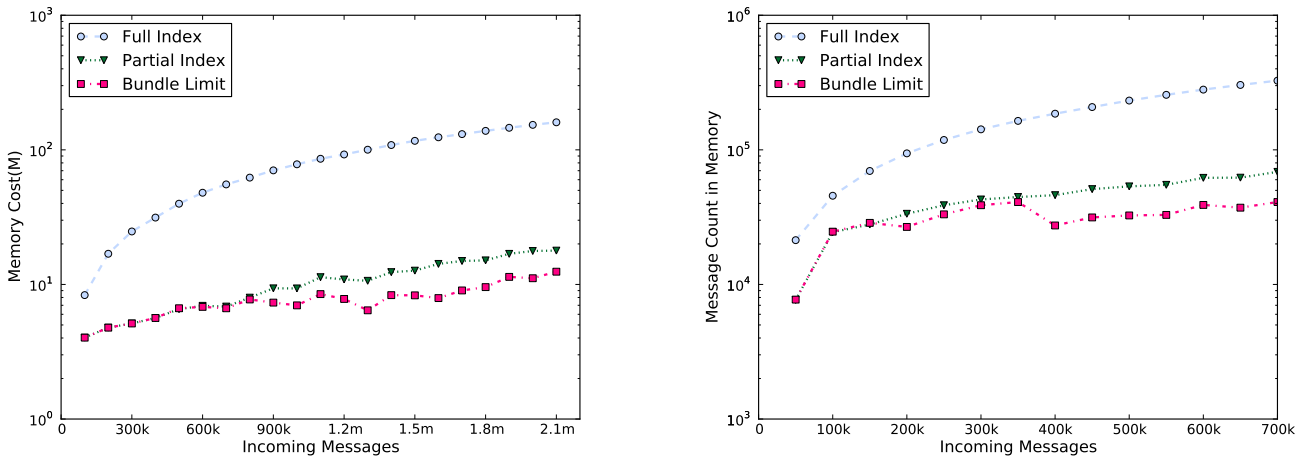
[7]Visit the demo site http://t.pku.edu.cn/tweet/bundle and input the above bundle id,check the bundle details.



(a) IBM CICS Partner Conference, ID:590716

(b) Samoa Tsunami, ID:1459658

Fig. 10. Extracted Provenance Bundles, Sept 2009



(a) Memory Usage

(b) Message Count

Fig. 11. Growth of Memory Cost under Different Approaches

memory overhead and time cost, the improved partial index variants gain a comparable provenance discovery performance.

Then we show the time consumption during different processing stages. In Figure 13, time cost of the three subsequent stages are calculated. The bundle match refers to finding a suitable bundle to insert the incoming new massage. The message placement needs to align the new message with a suitable message node inside the selected bundle. The memory refinement is the cost of bundle pool refinement operation. Time cost in this figure means the accumulated value. All of these implementations have a linear growth and show steadiness over the incoming messages. This is probably caused by the well tuned summary index structure for provenance match and the compact provenance bundle module for message allocation.

To conclude this experiment, the proposed provenance indexing solution has several benefits. First, it avoids the overhead of extensive provenance annotation maintenance. A multi-level provenance collection design is efficient and also capable of satisfying most information needs. Second, The efficient in-memory processing unit are able to cope with the real-time incoming messages. This enables unaware delay of provenance monitoring. Third, the provenance index supporting query experiment shows grate advantages of rich retrieval information over single message based search paradigms.

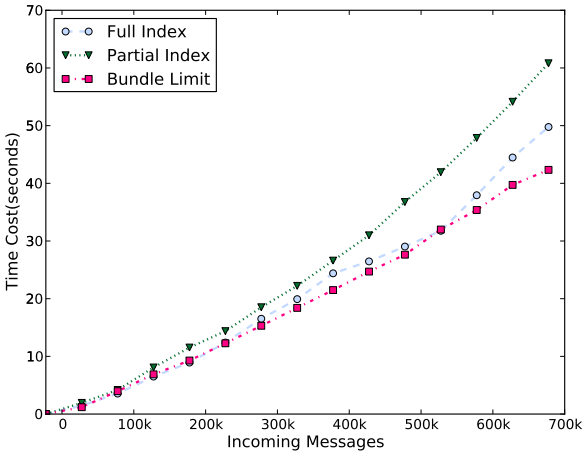


Fig. 12. Time Cost of Provenance Maintenance

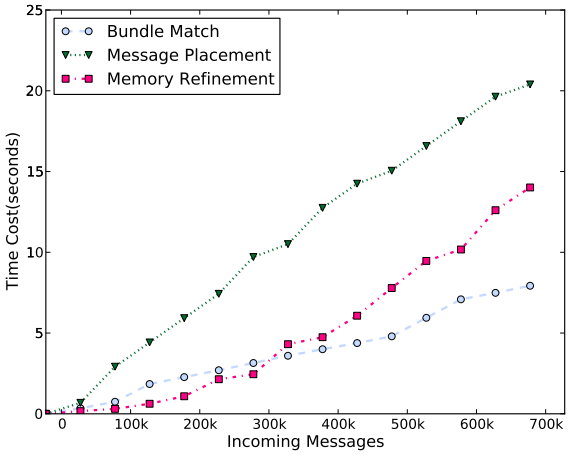


Fig. 13. Time Cost in Different Stages

VII. Conclusion

In this paper, we propose a provenance based indexing approach for micro-blog message management. We define the provenance concept in micro-blog settings, which focuses on trails of message connections and development. In the actual implementation, we utilize a grouping strategy to represent provenance information. An indexing scheme is also adopted to support efficient provenance discovery in high volume micro-blog message stream.

Empirical experiments and the online demo demonstrate the advantages of this provenance indexing method for better micro-blog content management. There are some promising future work ahead. For example, the provenance operators built on these provenance bundle and indexing structure could be investigated. By harnessing the user feedbacks and interaction inside bundles, we can develop the social provenance tools to enable collaborative data quality assessments.

Acknowledgment

This research was supported by the grants of Natural Science Foundation of China (No. 61073019 and 60933004), and HGJ Grant No. 2011ZX01042-001-001.

We also thank anonymous reviewers for their beneficial comments, which helps us improve this work.

References

[1] G. Miller, "Social scientists wade into the tweet stream," *Science*, vol. 333, no. 6051, pp. 1814 –1815, 2011.

[2] S. A. Golder and M. W. Macy, "Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures," *Science*, vol. 333, no. 6051, pp. 1878 –1881, 2011.

[3] J. Bollen and H. Mao, "Twitter mood as a stock market predictor," *Computer*, vol. 44, no. 10, pp. 91–94, Oct. 2011.

[4] J. Teevan, D. Ramage, and M. R. Morris, "#twittersearch: a comparison of microblog search and web search," in *Proc. of WSDM*, 2011, pp. 35–44.

[5] L. Moreau, "The foundations for provenance on the web," *Foundations and Trends in Web Science*, vol. 2, pp. 99–241, Feb 2010.

[6] J. Cheney, L. Chiticariu, and W. Tan, "Provenance in databases: Why, how, and where," *Foundations and Trends in Databases*, vol. 1, pp. 379–474, Apr 2009.

[7] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga, "The provenance of electronic data," *Communications of the ACM*, vol. 51, pp. 52–58, 2008.

[8] A. P. Chapman, H. V. Jagadish, and P. Ramanan, "Efficient provenance storage," in *Proc. of SIGMOD*, 2008, pp. 993–1006.

[9] B. Glavic and G. Alonso, "Perm: Processing provenance and data on the same data model through query rewriting," in *Proc. of ICDE*, 2009, pp. 174–185.

[10] G. Karvounarakis, Z. G. Ives, and V. Tannen, "Querying data provenance," in *Proc. of SIGMOD*, 2010, pp. 951–962.

[11] W. Zhou, M. Sherr, T. Tao, X. Li, B. T. Loo, and Y. Mao, "Efficient querying and maintenance of network provenance at internet-scale," in *Proc. of SIGMOD*, 2010, pp. 615–626.

[12] G. Barbier and H. Liu, "Information provenance in social media," in *Proc. of the 4th international conference on Social computing, behavioral-cultural modeling and prediction*, pp. 276–283.

[13] N. N. Vijayakumar and B. Plale, "Towards low overhead provenance tracking in near real-time stream filtering," in *Provenance and Annotation of Data, International Provenance and Annotation Workshop*, 2006, pp. 46–54.

[14] D. Budak and A. El Abbadi, "Information diffusion in social networks: Observing and influencing societal interests," *PVLDB*, vol. 4, no. 12, pp. 1–5.

[15] J. Leskovec, L. Backstrom, and J. Kleinberg, "Meme-tracking and the dynamics of the news cycle," in *Proc. of KDD*, 2009, pp. 497–506.

[16] S. Wu, J. Hofman, W. Mason, and D. Watts, "Who says what to whom on twitter," in *Proc. of WWW*, 2011, pp. 705–714.

[17] C. Chen, F. Li, B. C. Ooi, and S. Wu, "Ti: An efficient indexing mechanism for real-time search on tweets," in *Proc. of SIGMOD*, 2011, pp. 649–660.

[18] D. Ramage, S. Dumais, and D. Liebling, "Characterizing microblogs with topic models," in *Proc. of ICWSM*, 2010.

[19] X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E.-P. Lim, and X. Li, "Topical keyphrase extraction from twitter," in *Proc. of ACL*, 2011, pp. 379–388.

[20] *The Engineering Behind Twitters New Search Experience*, 2011, http://engineering.twitter.com/2011/05/engineering-behind-twitters-new-search.html.

[21] H. Kwak, C. Lee, H. Park, and S. Moon, "What is Twitter, a social network or a news media?" in *Proc. of WWW*, 2010, pp. 591–600.

[22] M. D. Choudhury, Y.-R. Lin, H. Sundaram, K. S. Candan, L. Xie, and A. Kelliher, "How does the data sampling strategy impact the discovery of information diffusion in social media?" in *Proc. of ICWSM*, 2010.