Interactive Hierarchical Tag Clouds for Summarizing Spatiotemporal Social Contents

Wei Kang [#1], Anthony K.H. Tung [#*2], Feng Zhao [*3], Xinyu Li [*4]

[#] *NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, Singapore*
[1] kangw@nus.edu.sg

[*] *School of Computing, National University of Singapore, Singapore*
{[2] atung, [3] zhaofeng, [4] lixinyu}@comp.nus.edu.sg

***Abstract*—In recent years, much effort has been invested in analyzing social network data. However, it remains a great challenge to support interactive exploration of such huge amounts of data. In this paper, we propose Vesta, a system that enables *v*isual *e*xploration of *s*ocial network data via *ta*g clouds. Under Vesta, users can interactively explore and extract summaries of social network contents published in a certain spatial region during a certain period of time. These summaries are represented using a novel concept called *hierarchical tag clouds*, which allows users to zoom in/out to explore more specific/general tag summaries. In Vesta, the spatiotemporal data is split into partitions. A novel biclustering approach is applied for each partition to extract summaries, which are then used to construct a hierarchical latent Dirichlet allocation model to generate a topic hierarchy. At runtime, the topic hierarchies in the relevant partitions of the user-specified region are merged in a probabilistic manner to form tag hierarchies, which are used to construct interactive hierarchical tag clouds for visualization. The result of an extensive experimental study verifies the efficiency and effectiveness of Vesta.**

I. Introduction

The wide adoption of Social Network Services (SNSs) has resulted in large amounts of social network contents (e.g., tweets in Twitter and newsfeeds in Facebook). For instance, Twitter CEO Dick Costolo revealed in June 2012 that Twitter was seeing 400 million tweets per day[1]. In particular, it is increasingly becoming more common for published contents to include both locational and temporal information which greatly enriches these SNSs posts. By exploring and summarizing these contents, we can discover what people are talking about in certain regions during a certain period of time. For example, in the 2012 US presidential election, Obama and Romney tried to win voters in key swing states by getting their campaign staffers to analyze newly published social network contents that were related to the election in each of these states and then adjust their campaign strategies accordingly to the analysis.

In this paper, we propose a system called Vesta that enables users to interactively browse summaries of social network contents of user-specified spatiotemporal regions[2]. To represent the summaries, we introduce a novel concept of hierarchical tag cloud, which organizes tags (keywords) of a summary in different levels of depth based on their degrees of generality. Namely, tags in higher/lower levels have more general/specific meanings. Users can grasp general ideas about what is popular in a specified spatiotemporal region at first glance, and then zoom in to lower levels if they want to see more details. Therefore, the increasing depths of a hierarchical tag cloud can effectively organize tags of a summary in different levels of abstraction and present details to users step by step.

As an example, Fig. 1 illustrates how users can interactively explore what were happening in London during the 2012 Olympic Games via hierarchical tag clouds. By setting a time range and selecting a geographic region as in Fig. 1(a), top-10 most interesting hierarchical tag clouds will be displayed, where different colors represent different topics. Initially, only tags at the first level of each hierarchical tag cloud are displayed (Fig. 1(b)), with the largest font size to convey the most general meanings. When zoomed in, tags at subsequent levels of each tag cloud are gradually displayed in smaller font size around the first level tags to provide more specific meanings (Fig. 1(c) and 1(d)). This process of hierarchical browsing of tag clouds can be repeated to trigger the display of the tags at any appropriate level.

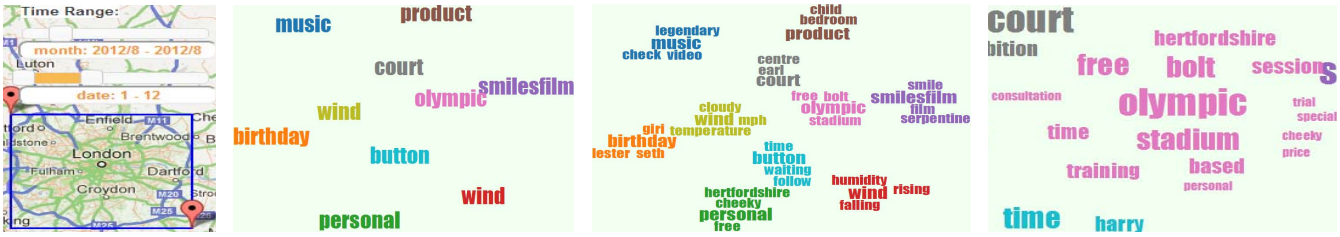
As shown in Fig. 1(b), one tag in the first level is "olympic", which summarizes the pink tag cloud and indicates that the tag cloud talks about the Olympic Games. To explore more about the tag cloud, we zoom into the second level as in Fig. 1(c). We can see that more tags are displayed, including "bolt", "stadium" and so on. Fig. 1(d) shows the pink tag cloud when we zoom into the fourth level, with even more tags added around "olympic" including "training", "time" in the third level and "price", "trial" in the fourth level.

Exploring social network contents interactively in hierarchical tag clouds is a novel and useful operation. It facilitates users in exploring different topics by only viewing summaries instead of directly reading plenty of contents. Users can drill down or roll up in the tag clouds to better understand the discovered knowledge interactively and hierarchically. They can also click any tag in a tag cloud to see the related contents if they want to know the exact underlying context. Moreover, users are allowed to specify two sets of spatiotemporal ranges to compare summaries of different regions for which common tags will be highlighted in *italic* type.

To provide interactive hierarchical tag clouds, one big challenge is to develop efficient methods to summarize social network contents. Latent Dirichlet allocation (LDA) [1] is a

[1] http://www.mediabistro.com/alltwitter/tag/tweets-per-day

[2] Try Vesta at http://db128gb-b.ddns.comp.nus.edu.sg/kangwei/bicluster.



(a) Parameters (b) The first level (c) The second level (d) The fourth level

Fig. 1. Hierarchical Tag Cloud

popular topic model for summarizing and extracting topics from documents, and various extensions [2], [3] have been studied, including extending LDA to deal with short documents like tweets [4]. However, LDA-based methods are not able to handle huge amounts of data efficiently since they often perform inference by adopting MCMC algorithms such as Gibbs sampling. The streaming nature of social network content data renders the summarization task even more challenging.

To tackle the challenges, we propose an efficient biclustering approach based on formal concept analysis [5], [6]. The results are called biclusters, and each consists of a set of tags (keywords in contents) and a set of contents, where the tags frequently co-occurs in the contents. The contents are clustered together due to the common tags they share, which means they are quite likely to discuss similar things. To this effect, tags in a bicluster serve as a summary of the contents. Thus the summarization of social network contents is converted to the generation of biclusters. Besides, biclustering clusters tags and contents simultaneously so that each bicluster contains both tags as a summary and contents from which the summary is extracted, where the contents supplement the tags to provide more context for a summary. This distinguishes our approach from normal clustering that often clusters in a single direction and from LDA-based methods that generate only summaries. Although finding the largest bicluster is NP-complete [7], [8] and many heuristic biclustering methods converge slowly, our approach makes use of formal concept analysis to generate "full-density" biclusters (i.e., each tag of a bicluster appears in each content of that bicluster) very efficiently. "Full-density" biclusters can be strict, but it is easy to relax a bicluster by adding more tags/contents or merging similar biclusters.

To further enhance the system scalability and visualize the summaries in hierarchical tag clouds, we propose an efficient two-phase, disk-based partition-and-merge scheme in Vesta. The partitioning phase, which is done offline, consists of three steps. In the first step, we split the spatiotemporal social network data into partitions. For instance, each such partition contains one day's contents for a spatial region in our case. In the second step, we summarize the social network contents using the proposed biclustering approach. In the third step, we apply the hierarchical LDA (hLDA) model [2], [3] to generate for each partition a topic hierarchy. Topic hierarchies will be merged in the subsequent merging phase to form tag hierarchies, which organizes tags of merged biclusters in different levels from general to specific. In this way, users can visualize the summaries in a hierarchical fashion. At the end of the partitioning phase, we have for each partition a set of biclusters and a topic hierarchy generated by using the contents in these biclusters. The partitioning phase can also be carried out in parallel easily.

The merging phase is done at runtime when users query Vesta by specifying the spatiotemporal ranges. Vesta first computes the partitions that are covered by the query range and then proceeds to merge similar biclusters from the selected partitions. A probabilistic merging algorithm is proposed to combine the corresponding topic hierarchies to form tag hierarchies for visualization purpose. Vesta is efficient as most of the computationally intensive tasks are done offline in the partitioning phase; the runtime merging phase is fast, making Vesta an effective interactive visualization tool.

To select the most interesting summaries, we propose a score function customizable according to users' preference. We also evaluate the mismatch problem quantitatively so as to provide feedback to adjust the partition size adaptively.

We make the following contributions in this paper: (1) We propose a novel way to explore spatiotemporal social network contents via hierarchical tag clouds. (2) We propose a summarization method by biclustering the contents and further extend it to a disk-based partition-and-merge scheme for better scalability. (3) We generate and merge topic hierarchies so as to visualize summaries in hierarchical tag clouds. (4) We implement all these mechanisms into a system called Vesta.

The rest of the paper is organized as follows. We review the related work in section II and give the problem formulation in section III. Section IV introduces our new biclustering method. In section V, we propose a partition-and-merge scheme, followed by the introduction of the system implementation in section VI. Section VII presents the experimental study and section VIII concludes the paper.

II. Related Work

Tag clouds are visual presentations of a set of words, or "tags", selected by some rationale, in which attributes of the text (e.g., size, color) are used to represent features (e.g., frequency) of the associated terms. The authors in [9] provided an extensive evaluation of tag clouds for impression formation. Bielenberg et al. [10] presented their tag cloud in a circular layout where tags locating nearer the center are more important. Dubinko et al. [11] proposed to visualize the evolution of tags within the Flickr online image sharing community. In this paper, we combine tag cloud and topic hierarchy to visualize

the summaries of social network contents in hierarchical tag clouds, which is suitable for large-scale tag representation.

Biclustering [12] was first proposed decades ago and became popular after Cheng et al. [7] adopted it for gene expression data analysis. Many approaches were proposed to do biclustering in both gene and condition dimensions simultaneously to find a set of genes expressing coherently in a set of conditions [7], [13], [14]. It is also used in text analysis referred to as "co-clustering" to analyze dyadic data [15], [16], e.g., the document and word co-occurrence frequencies. An information theoretic co-clustering framework was proposed in [17] and efforts were spent in extending existing co-clustering to support constraints on both words and documents [16].

Since finding the largest bicluster is NP-complete for almost all variants of the biclustering problem [7], [8], most biclustering approaches are designed with heuristics in a non-deterministic manner, thereby often taking a long time to converge [8], [14]. Recently, some authors [18] applied biclustering to the analysis of social network data. They proposed to do biclustering by means of formal concept analysis (FCA). FCA [5], [6] is a data analysis method in growing popularity across various domain, which is widely used to discover relationships between a set of objects and a set of attributes. Although biclustering using FCA is more efficient than general biclustering approaches, this approach is still unable to scale to huge amounts of social network data. Besides, it tends to generate sparse biclusters but miss the dense ones [18]. Our proposed biclustering method also makes use of FCA to summarize social network data, but can handle large dataset more efficiently and generate denser biclusters.

Latent Dirichlet allocation (LDA) [1] is a popular topic model for latent topic discovery in text collection. It inspires various extensions such as the correspondence LDA models [19] and topic-sentiment models [20]. Another extension hierarchical LDA (hLDA) [2], [3] is germane to our work. The authors proposed a nested Chinese restaurant process and showed how to use the process to do Bayesian nonparametric inference of topic hierarchies. Ma et al. in [21] introduced a Master-Slave Topic Model to discover and summarize topics in readers comments and the related news articles. Another work [22] proposed a model called Uni-Topical Blockmodels to capture topics from tweet replies among Twitter users. Twevent [23] proposed a segment-based event detection system for tweets, which helps users to understand the topics attracting a large number of common Twitter actors.

Our work focuses on the spatiotemporal social network content data summarization. Then we adopt hLDA to generate tag hierarchies to help visualize the summaries. We will show in the experimental study that hLDA cannot handle large amounts of content data for the analysis of topic hierarchies. Our work mainly differs from hLDA as follows. (1) We discover summaries and the related contents from which the summaries are extracted simultaneously. (2) We generate discrete summaries which capture various interesting topics while hLDA generates topics at different levels of abstraction. (3) We generate a tag hierarchy for each summary while hLDA generates a topic hierarchy for a corpus of documents.

III. Problem Formulation

A. Preliminaries

Definition 1: **Social network contents** (or **contents** for short) are textual microblogs, such as tweets, published by users in SNSs. Each content is denoted by $c_i \in \hat{C}$, where $\hat{C}$ is the collection of all contents. **Tags** are meaningful keywords in contents after stop words removal. Each tag is denoted by $t_i \in \hat{T}$, where $\hat{T}$ is the collection of tags. A **social network matrix** $M_{\hat{T}\hat{C}}$ is a $|\hat{T}|$ by $|\hat{C}|$ matrix whose rows represent tags and whose columns represent social network contents.

Definition 2: A **bicluster**, denoted by (T, C), is a pair consisting of a subset T of tags $\hat{T}$ and a subset C of social network contents $\hat{C}$. The process of finding biclusters is called **biclustering**. A bicluster corresponds to a submatrix M_{TC} of the social network matrix $M_{\hat{T}\hat{C}}$, where the tag set T of the bicluster corresponds to the row set of the submatrix and the content set C to the column set.

The value of any element M_{ij} in $M_{\hat{T}\hat{C}}$ can be 1 or 0, indicating whether tag t_i appears in content c_j or not. Therefore, biclustering in social network data analysis is the process to explore the social network matrix $M_{\hat{T}\hat{C}}$ to discover biclusters (submatrices) satisfying certain criteria. Density is commonly used as an effective measure for determining the quality of biclusters. Without loss of generality, we use density as one of the major measures to assess the quality of a bicluster in this paper. Users can easily replace it with other measures such as variance [12] and mean squared residue [7].

Definition 3: The **density** of (T, C), denoted by $den(T, C)$, is the non-zero rate of its corresponding submatrix M_{TC}, which equals the ratio of the number of 1's to $|T||C|$. The **size** of (T, C), denoted by $sz(T, C)$, is defined to be $\min(|T|, |C|)$.

In a bicluster, a set of tags co-occur in a set of contents such that the tags can be viewed as a summary of the contents. The denser the bicluster, the more "consistent" the contents in a summary. A density threshold δ_{den} and size threshold δ_{sz} can filter out sparse biclusters with very few tags or contents, thereby avoiding bicluster explosion. Having the biclusters, we visualize them in hierarchical tag clouds, which calls for the generation of tag hierarchy using the tags in each bicluster.

Definition 4: If a tag set T can be divided into a few non-empty subsets T_1,T_2,...,T_n such that $T_i \cap T_j = \emptyset$, $\cup_{i=1}^{n} T_i = T$ and $T_i < T_j$ (i.e., every two different subsets follow a total order) for any $i, j \in \{1, 2, ...n\}(i < j)$, we say that T_1,T_2,...,T_n form a **tag hierarchy** $\mathcal{H}$ for the tag set T and that T_i contains tags in the ith level of the hierarchy.

A tag hierarchy organizes the tags of a bicluster in different levels from general to specific. A hierarchical tag cloud is actually the visualized form of a tag hierarchy while a tag hierarchy defines the levels of tags for a hierarchical tag cloud. By showing the tags level by level in a hierarchical tag cloud, users can better understand the meaning of the tags and the relationships among them in an interactive way.

B. Problem Definition

By highlighting a geographic region R_{geo}, our aim is to (1) generate the top-k most interesting biclusters (T_i, C_i) $(1 \leq i \leq k)$, where $den(T_i, C_i) \geq \delta_{den}$ and $sz(T_i, C_i) \geq \delta_{sz}$, to summarize the set of social network contents $\hat{C}$ published within R_{geo} during a user-specified period of time R_{tim}, and (2) build a tag hierarchy $\mathcal{H}_i$ for each tag set T_i so as to visualize these summaries in hierarchical tag clouds. The interestingness of the biclusters is measured by a score function, which will be introduced in section V-D.

With hierarchical tag clouds, users are empowered to explore any geographic region of interest, or even discover commonalities and uniqueness by comparing different regions.

IV. Biclustering Approach

Most biclustering algorithms iterate many times prior to convergence, which renders the execution time large and unpredictable. Besides, biclustering was initially proposed to analyze gene expression data often in small size with no or a small number of empty values. Social network content data, however, is usually large and sparse, further disqualifying the application of common biclustering algorithms in social network data analysis. Next, we will propose an efficient and deterministic way of finding biclusters based on FCA.

A. Introduction to Formal Concept Analysis (FCA)

Definition 5: A **formal context** is a triplet $(\hat{A}, \hat{O}, I)$ where $\hat{O}$ is an object set, $\hat{A}$ is an attribute set and $I \subseteq \hat{A} \times \hat{O}$ represents the relationships of objects in $\hat{O}$ and attributes in $\hat{A}$. A pair (A, O), where $A \subseteq \hat{A}, O \subseteq \hat{O}$, is called a **formal concept** of the formal context $(\hat{A}, \hat{O}, I)$ if it satisfies the *fullness* property: $\forall_{a \in A, o \in O}(a, o) \in I$, and the *maximum* property: $\forall_{o \notin O} \exists_{a \in A}(a, o) \notin I$ and $\forall_{a \notin A} \exists_{o \in O}(a, o) \notin I$.

Every formal concept (A, O) is *full* and *maximal*: being full means that every object in O has all attributes in A, while being maximal means that, if any attribute $a \notin A$ (or any object $o \notin O$) is added to A (or O), $(A \cup \{a\}, O)$ (or $(A, O \cup \{o\})$) is not full and thus not a formal concept any more.

Example 1: Fig. 2(a) shows a formal context $(\hat{A}, \hat{O}, I)$ where $\hat{A} = \{a_1, a_2, a_3, a_4\}, \hat{O} = \{o_1, o_2, o_3, o_4, o_5\}$. The cells with X_{ij} means a_i is an attribute of object o_j, i.e., $(a_i, o_j) \in I$. (A, O) is a formal concept where $A = \{a_2, a_3, a_4\}, O = \{o_2, o_4\}$. (A, O) is full because every object $o_j \in O$ has all the attributes in A. It is also maximal. If we add in any attribute or object which is not in A or O, o_5 for instance, (A, O) would not be a formal concept because $(a_4, o_5) \notin I$.

Compared with the definition of social network matrix, it is obvious that a formal context can also be perceived as a matrix M whose rows are attributes and whose columns are objects. The value of an element M_{ij} is set to 1 if the *j*th object has the *i*th attribute, 0 otherwise. Similarly, a formal concept (A, O) can be perceived as a "full-density" bicluster whose density equals 1. From this perspective, (A, O) being full means the submatrix of the bicluster has no empty value while (A, O) being maximal means the submatrix will have empty value(s) if any $a \notin A$ or $o \notin O$ is added to A or O. Therefore, the problem of generating biclusters now becomes the generation of formal concepts.

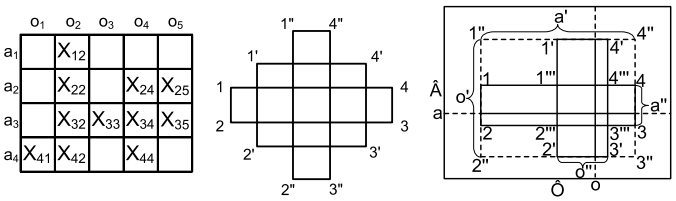
B. Properties of Formal Concept

A partial order "$\leq$" can be defined in a formal context: given two formal concepts (A_1, O_1) and (A_2, O_2), we have $(A_1, O_1) \leq (A_2, O_2)$ if $O_1 \subseteq O_2$ (or $A_1 \supseteq A_2$ equivalently). This also implies a relationship between the object set and attribute set of a formal concept in terms of set size. That is, the larger the size of the object set, the smaller that of the attribute set, vice versa. This actually follows, according to Galois theory, the antitone Galois connection [24] which is defined as a pair of antitone functions $F : \mathbb{A} \to \mathbb{B}$ and $G : \mathbb{B} \to \mathbb{A}$ between two partially ordered sets $\mathbb{A}$ and $\mathbb{B}$, such that $\beta \leq F(\alpha)$ iff $\alpha \leq G(\beta)$ where $\alpha \in \mathbb{A}, \beta \in \mathbb{B}$. Here F can be viewed as a function mapping an attribute set to an object set while G as a function mapping an object set to an attribute set. Next we introduce the Galois operators "$'$" and "$''$" which work as the above functions.

Given an attribute set $A \subseteq \hat{A}$ and an object set $O \subseteq \hat{O}$,

$$A' = \{o \in \hat{O} | \forall_{a \in A}(o, a) \in I\}$$
$$O' = \{a \in \hat{A} | \forall_{o \in O}(o, a) \in I\} \quad (1)$$

where A' is an object set, every object in which has all attributes in A, and O' is an attribute set, every attribute in which is an attribute of all objects in O. Similarly, A'' is an attribute set by applying the Galois operator twice to A (or once to A') and O'' is an object set by applying the Galois operator twice to O (or once to O'). The operator "$''$" is monotone (i.e., $A'' \subseteq B''$ if $A \subseteq B$), idempotent (i.e., $(A'')'' = A''$), and extensive (i.e., $A \subseteq A''$) [18].



(a) Example (b) Change in shape (c) Four types

Fig. 2. Formal Concepts

Lemma 1: Given any attribute set $A \subseteq \hat{A}$ and any object set $O \subseteq \hat{O}$, both (A'', A') and (O', O'') are formal concepts.

Proof: We only prove that (A'', A') is a formal concept, (O', O'') can be proven similarly. According to Eq. 1, it is easy to note that $A'' \subseteq A$, $A' \subseteq O$ and $\forall_{a \in A'', o \in A'}(a, o) \in I$. Next we prove (A'', A') is maximal by contradiction. Suppose the object set A' is not maximal. There must exist an object $o' \notin A'$ that has all attributes in A''. Since $A \subseteq A''$ ("$''$" is extensive), o' has all attributes in A. This contradicts Eq. 1 stating that A' contains all objects having every attribute in A. Suppose the attribute set A'' is not maximal. There must exist an attribute $a' \notin A''$ shared by all objects in A'. This contradicts $A'' = \{a \in \hat{A} | \forall_{o \in A'}(o, a) \in I\}$ according to Eq. 1, meaning that all attributes shared by every object in A' are included in A''. Hence the proof. ■

Example 2: Consider the formal context in Fig. 2(a). Given $A = \{a_2, a_3\}$, we have $A' = \{o_2, o_4, o_5\}$, where each object in A' has all attributes in A, and $A'' = \{a_2, a_3\}$, where each attribute in A'' is an attribute of any object in A', by applying the Galois operators. It is easy to know that (A'', A') is a formal concept according to its definition.

This lemma provides a way to generate formal concepts. Given any attribute or object set, we can apply the Galois operator once and twice respectively to generate the attribute and object sets of a new formal concept. However, should we enumerate all the possible attribute or object sets to generate formal concepts (which leads to $2^{|\hat{A}|} + 2^{|\hat{O}|}$ different sets for a formal context $(\hat{A}, \hat{O}, I)$ in the worse case)? Besides, the formal concepts in a formal context can be ordered to form a concept lattice according to the inclusion relationships of the object or attribute sets. The number of concepts in the lattice is also up to $2^{\min(|\hat{A}|,|\hat{O}|)}$ in the worse case.

To answer the above question, we need to investigate how the size of an attribute or object set affects the size of its resultant formal concept. It is illustrated in Fig. 2(b) where formal concepts are represented by rectangles with the length representing the size of the object set and height representing that of the attribute set. Given three attribute sets A_1, A_2 and A_3 ($A_1 \subseteq A_2 \subseteq A_3$), their formal concepts are (A_1'', A_1'), (A_2'', A_2') and (A_3'', A_3') which are denoted by rectangle 1234, $1'2'3'4'$ and $1''2''3''4''$ respectively in Fig. 2(b). Note that $A_1' \supseteq A_2' \supseteq A_3'$ because a smaller attribute set is likely to be shared by a larger object set and vice versa. Similarly, we have $A_1'' \subseteq A_2'' \subseteq A_3''$. Thus, if we expand A_1 to A_2 and even to A_3, the object set A_1' in the corresponding formal concept starts shrinking and the attribute set A_1'' starts expanding. This can be reflected in Fig. 2(b) by changing the shape of the formal concept from rectangle 1234 to $1'2'3'4'$, and to $1''2''3''4''$. Likewise, given any object set O, when it expands by adding more objects in it, the process can be reflected by changing the shape of the formal concept (O', O'') from $1''2''3''4''$ to $1'2'3'4'$ to 1234. The above discussion indicates that in a formal concept we can expand the attribute (or object) set by shrinking the object (or attribute) set. In this paper, we generate formal concepts based on each attribute and object, which greatly reduce the number of formal concepts to be generated to at most $|\hat{A}| + |\hat{O}|$. Formal concepts with larger attribute (or object) sets can be generated easily by shrinking their object (or attribute) sets.

C. Generating Biclusters

Given any attribute $a \in \hat{A}$ and any object $o \in \hat{O}$, we can find their corresponding formal concepts (a'', a') (we write $(\{a\}'', \{a\}')$ as (a'', a') for simplicity) and (o', o''), which are shown in Fig. 2(c) as rectangle 1234 and $1'2'3'4'$ respectively. They can be viewed as "full-density" biclusters according to the fullness and maximum properties. Besides, there are two other rectangles $1'''2'''3'''4'''$ and $1''2''3''4''$. The former is the overlap of 1234 and $1'2'3'4'$, which is too tight to be used as a bicluster since it often leads to very small attribute set and object set. The latter is treated as extended formal concept (by allowing the density less than 1) and used to generate biclusters in [18]. We find this form also less qualified since it often includes many empty values thus greatly decreases the density. The authors in [18] set a threshold to filter out those with small density. However, this causes a problem that when they filter out some less dense extended formal concept $1''2''3''4''$, they also discard the "full-density" formal concepts 1234 and $1'2'3'4'$ that are within $1''2''3''4''$.

Therefore, we use the two real formal concepts 1234 and $1'2'3'4'$ to generate biclusters in this paper. They differ from each other in terms of shapes. Specifically, 1234 tends to have a larger object set and smaller attribute set while $1'2'3'4'$ tends to have a smaller object set and large attribute set. In terms of bicluster, 1234 corresponds to biclusters with more contents and fewer tags while $1'2'3'4'$ to those with fewer contents and more tags. Users can generate different forms according to their requirements. We refer to the methods generating biclusters in the form of 1234 and $1'2'3'4'$ as *ours_tag* and *ours_content* respectively below. Also note that many duplicate biclusters could be generated because contents in a bicluster often have similar tags and each of the tag-content pairs would be used to generate biclusters. To avoid duplication, we only generate biclusters without overlap. That is, if a tag or content appears in the tag set or content set of a bicluster, it will not be used to generate other biclusters.

D. Relaxation

Since the "full-density" biclusters are too strict, we can relax them through either transformation or merging.

Transformation refers to adding tags or contents to a bicluster. We take the addition of tags as an example. Given a bicluster (T, C), the contents in C may have another set T' of tags which are not included in T because having those tags makes the bicluster no longer satisfy the fullness property. We now break the property to enlarge T by adding tags in T' to T. Tags in T' should be ordered so that each time the tag leading to the least density decrease is added to T. This continues until the next tag to be added makes the density of the bicluster less than a density threshold δ_{den}. To add even more tags, we can delete a few contents which lead to the largest density increase after the deletion so as to increase the bicluster density first.

Merging can also result in relaxed biclusters. In this paper, we merge biclusters sharing common tags since those biclusters are more likely to discuss similar topics and may thus be merged. Readers can refer to section V-C for details.

V. Partition-and-Merge Scheme

To further improve the scalability and reduce the system response time, we next extend the proposed biclustering approach to a disk-based partition-and-merge (PM for short) scheme. In the offline partitioning phase, we split the data space (all the content data) into partitions and generate biclusters for each partition. Then biclusters in certain partitions are merged efficiently at runtime given user-specified spatiotemporal parameters. To visualize the merged biclusters as hierarchical tag clouds, we adopt hLDA to produce topic

hierarchies for partitions in the partitioning phase and generate tag hierarchies by merging the topic hierarchies at runtime.

A. Offline Partitioning

Three dimensions, i.e., longitude, latitude and time, need to be considered for partitioning. We first slice the data on a daily basis and then split the geographic space for each day adaptively according to data density by using a space-partitioning data structure such as kd-tree [25] or quadtree [26]. Note that the partitioning layout may be different for each day since the data distribution varies every day. Also note that slicing data on a daily basis is a tradeoff between two possible mismatches. Since the temporal parameter can be set to any consecutive days, covering more than one day in a partition may lead to a temporal mismatch between the parameter and partitions. On the other hand, covering less than 24 hours may lead to larger spatial partitions and increase the geographic mismatch discussed in section V-E.

B. Offline Pre-computation

1) Pre-computation of Biclusters: Once the data space is split, we can generate biclusters for each partition using the method proposed in section IV. One potential problem is that partition-based biclustering may leave out biclusters that can only be formed by using contents of multiple partitions. This may arise if the number of contents regarding certain topics are small. In this case generating biclusters in as small size as possible may relieve the problem. However, this could produce uninteresting biclusters and lead to bicluster explosion.

Given this observation, we make an **assumption** that globally interesting summaries are also interesting in certain partitions. Specifically, if a bicluster about an interesting summary is generated from contents in some geographic region, biclusters about the similar summary can also be generated from contents in certain partitions of that region. We will introduce a score function to measure the interestingness of a summary later and validate this assumption in the experimental study.

2) Pre-computation of Topic Hierarchies: hLDA is extended from LDA to generate topic hierarchies for documents. We apply hLDA to social network contents in our context to generate topic hierarchies for different partitions. Topic hierarchies are generated after the biclustering process for each partition by using the contents of the biclusters. Topic hierarchies generated by hLDA are trees, with a few closely related tags in each node [3]. The tags in higher level nodes are more general and those in lower level nodes more specific, which provides the possibility of generating tag hierarchies with different levels of tags by merging topic hierarchies.

The pre-computation of biclusters and topic hierarchies can be done easily for different partitions in parallel.

C. Online Merging

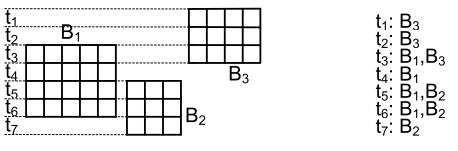
1) Merging Biclusters: Given the spatiotemporal parameters, i.e., a time range R_{tim} measured in days and a geographic region R_{geo} measured in coordinates, biclusters in partitions falling within R_{tim} and R_{geo} need to be merged together to produce "unified" results. Note that if two biclusters (T_1, C_1) and (T_2, C_2) are merged to form a new bicluster (T, C), it will have all tags and contents from (T_1, C_1) and (T_2, C_2), i.e., $T = T_1 \cup T_2$, $C = C_1 \cup C_2$. Suppose the corresponding matrix of (T, C) is denoted by M_{TC}. The density of (T, C), $den(T, C)$, is thus the non-zero rate of M_{TC}. Next we give algorithm 1 for merging biclusters sharing common tags.

Given a group $\mathcal{P}$ of the biclusters and a density threshold δ_{den}, the algorithm produces a set $\mathcal{B}$ of merged biclusters. We initialize a set B_{used} storing used biclusters to empty in line 2 and start to merge biclusters by checking each bicluster in $\mathcal{P}$ but not in B_{used} in line 3. When we start from B_i we first label it as used in line 4. Line 5 and 6 try to find a set $\mathcal{P}'$ of biclusters having common tags with B_i and sort them according to the number of common tags in descending order so that those with more common tags can be merged with B_i earlier. To accelerate the search for $\mathcal{P}'$, we build an inverted list to map each tag to biclusters having that tag. Lines 7 to 13 merge B_i with the ordered biclusters in $\mathcal{P}'$. At first, B_i is merged with the first bicluster in $\mathcal{P}'$ to form a new bicluster B_{new}. Then B_{new} is merged with subsequent biclusters in $\mathcal{P}'$ in turn. Note that the merging action only happens if the density of the new merged bicluster is no less than δ_{den} (line 10) and that biclusters used to form B_{new} are labeled as used (line 12). After the merging phase finishes, B_{new} is added to $\mathcal{B}$ (line 13). The algorithm continues until all biclusters in $\mathcal{P}$ are used. We use an example below to illustrate the algorithm.

Algorithm 1: The Bicluster Merging Algorithm

Input: a group $\mathcal{P}$ of biclusters $(T_i, C_i)(i = 1, 2, ..., n)$ (use B_i to denote each bicluster for short), a user-defined density threshold δ_{den}
Output: a merged bicluster set $\mathcal{B}$

1 **begin**
2 let $B_{used} = \emptyset$;
3 **foreach** $B_i \in \mathcal{P}$ *and* $B_i \notin B_{used}$ **do**
4 let $B_{used} = B_{used} \cup \{B_i\}$;
5 find a subset $\mathcal{P}'$ of biclusters in $\mathcal{P}$ but not in B_{used} such that $\forall_{B \in \mathcal{P}'} B \bigcap B_i \neq \emptyset$;
6 sort $B \in \mathcal{P}'$ by $|B \bigcap B_i|$ in descending order;
7 let $B_{new} = B_i$;
8 **foreach** $B \in ordered\ \mathcal{P}'$ **do**
9 merge B_{new} and B to form a new bicluster B_{tmp};
10 **if** $den(B_{tmp}) \geq \delta_{den}$ **then**
11 let $B_{new} = B_{tmp}$;
12 let $B_{used} = B_{used} \cup \{B\}$;
13 add B_{new} to $\mathcal{B}$;



(a) Biclusters (b) Inverted list

Fig. 3. Bicluster merging example

Example 3: Consider the three biclusters B_1, B_2 and B_3 only sharing common tags (rows) in Fig. 3(a). Suppose δ_{den} is 0.5. Algorithm 1 starts the merging from B_1 by searching for

biclusters having common tags with B_1 as candidates which is B_2 and B_3. B_2 and B_3 should be sorted so that the one (B_2 here) with more common tags is merged with B_1 first. To search more efficiently, we build an inverted list as in Fig. 3(b) showing which tag appears in which biclusters. Since B_1 have four tags, we can look up them in the inverted list to generate an ordered candidate set $\{B_2 : 2, B_3 : 1\}$ (note that B_1 is removed from the list) easily. Each bicluster in the set has a number which is the frequency of the bicluster mapped to the four tags. This number also means the number of common tags shared with B_1 and thus can be used to sort the candidate biclusters. B_2 having more common tags is used to merge with B_1 first to form a new bicluster B_{tmp} whose tag (or content) set is the union of the tag (or content) sets of B_1 and B_2. The density of B_{tmp} is $(4 \times 5 + 3 \times 3)/(5 \times 8) = 0.725 > \delta_{den}$, meaning the merging of B_1 and B_2 is acceptable. Next we use B_3 to merge with B_{tmp}. The new bicluster's density is $(5 \times 8 \times 0.725 + 3 \times 4)/(7 \times 12) = 0.488 < \delta_{den}$, meaning that we should cancel merging B_{tmp} with B_3. Since no further merging can be done, the algorithm outputs B_{tmp} and B_3 and terminates. In our system, we precalculate the density by assuming two biclusters were merged. If the density is less than δ_{den}, we do not merge them actually.

2) Merging Topic Hierarchies: When biclusters are merged together, the topic hierarchies for partitions having those biclusters also need to be merged to form a tag hierarchy so as to visualize the new bicluster. Given that the same tag can appear in multiple levels of the topic hierarchies generated by hLDA, we next propose a probabilistic approach to generating "unified" tag hierarchies in which each tag only appears in one level. Each of the resultant tag hierarchies has one node containing several tags in each level, and each node has at most one child in terms of the tree structure.

	$level_1$	$level_2$	...	$level_v$
tag_1	c_{11}	c_{12}	...	c_{1v}
tag_2	c_{21}	c_{22}	...	c_{2v}
...	...	...	...	...
tag_u	c_{u1}	c_{u2}	...	c_{uv}

Fig. 4. Tag-Level Matrix

Given m topic hierarchies T_1^i,T_2^i,...,$T_{n_i}^i (1 \le i \le m)$ to be merged together, where T_j^i denotes the tag set in the jth level of the ith topic hierarchy which has n_i levels in total, we first construct a vote-based tag-level matrix shown in Fig. 4. Rows in the matrix corresponds to tags while columns to levels. c_{ij} $(1 \le i \le u, 1 \le j \le v)$ in the ith row and jth column indicates the frequency of tag i appearing in level j throughout the m topic hierarchies. Based on the tag-level matrix, we define a weight function for each tag i in level j as follows.

$$weight(t_i, l_j) = c_{ij}^2/(\Sigma_i c_{ij} \Sigma_j c_{ij}) \quad (2)$$

The weight has two factors. The first one $c_{ij}/\Sigma_i c_{ij}$ captures how likely different tags are chosen for level j while the second one $c_{ij}/\Sigma_j c_{ij}$ captures how likely different levels contain tag i. The weight chooses tags for each level with the intuition that the larger the weight for tag i and level j, the more likely the tag appears in that level. In algorithm 2, we normalize $weight(t_i, l_j)$ for all tags in each level to select the most likely tags hierarchically. The algorithm computes $weight(t_i, l_j)$ using the tag-level matrix built on the m topic hierarchies from line 1 to 6. Then, starting from the first level, it chooses tags probabilistically for each level in turn from line 7 to 13. Note that by drawing tags without replacement, we disallow one tag to be chosen for multiple levels (line 9 to 12). After drawing tags for the $n_0 - 1$ levels, we leave the unused tags in T_{unused} for the last level directly (line 13).

Algorithm 2: The Topic Hierarchy Merging Algorithm

Input: m topic hierarchies T_1^i,T_2^i,...,$T_{n_i}^i (i \in \{1, 2, ..., m\})$, the number of levels n_0 ($n_0 \le \max(n_1, n_2, ...n_m)$) and the number of tags s_l ($l \in \{1, 2, ..., n_0\}$) in each level of the resultant tag hierarchy
Output: a tag hierarchy $\mathcal{H}$: T_1^0,T_2^0,...,$T_{n_0}^0$

1 **begin**
2 let $T_1^0 = T_2^0 = ... = T_{n_0}^0 = \emptyset$;
3 build the tag-level matrix M using the m topic hierarchies;
4 let $T_{unused} = T_{all} = \{1, 2, ...\}$ be the set of all tag indexes in M;
5 let $u = |T_{all}|, v = \max(n_1, n_2, ...n_m)$;
6 compute $weight(t_i, l_j)$ based on M according to Eq. 2 for all i and j where $1 \le i \le u$, $1 \le j \le v$;
7 **foreach** *level* $l \in \{1, 2, ..., n_0 - 1\}$ **do**
8 **while** $|T_l^0| < s_l$ **do**
9 normalize $weight(t_i, l)$ such that $\sum_i weight'(t_i, l) = 1$ for each $i \in T_{unused}$;
10 draw a tag i according to the normalized probabilities;
11 $|T_l^0| = |T_l^0| \cup \{i\}$;
12 $T_{unused} = T_{unused} \backslash \{i\}$;
13 let $T_{n_0}^0 = T_{unused}$;

D. Ranking Merged Biclusters

Although the number of biclusters decreases after merging, there are still many biclusters due to the abundance of various topics emerging in social networks. Besides, users may want to see the most interesting summaries, e.g., those discussed by more people or providing more information. Below we propose a score function so that users can rank all the biclusters to find the most interesting ones according to their preference by simply tuning a single parameter.

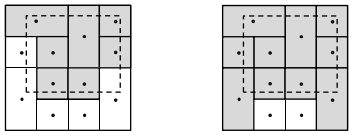
$$Score(T, C) = den(T, C) \cdot \log(|T||C|) \cdot (|C|/|T|)^p \quad (3)$$

where $|T|$ and $|C|$ are the numbers of tags and contents respectively in a bicluster (T, C) and p is an integer tuning parameter. In the score function, the first two factors $den(T, C)$ and $\log(|T||C|)$ indicate that biclusters with larger density or size would be ranked higher. The third factor $(|C|/|T|)^p$ incorporates users' preference by tuning $p \in \{0, \pm 1, \pm 2, ...\}$. For instance, users can set $p = 1$ in favor of biclusters with more contents or set $p = -1$ in favor of biclusters with more tags. The larger the absolute value of p, the stronger users stress their preference. However, the absolute value of p should not be very large to avoid the dominance of the third factor.

E. Mismatch Problem

Recall that biclusters are merged when the corresponding partitions fall into the user-specified spatiotemporal parameters

R_{tim} and R_{geo}. Since one partition can only be associated with a certain date, it is easy to check whether a partition falls within R_{tim}. Next we consider two different cases determining whether a partition falls in R_{geo} (dashed boxes in Fig. 5): (1) the centroids of the partitions are within R_{geo} and (2) the partitions overlap R_{geo}. Both cases can lead to mismatch problem. In case one, biclusters in the shadowed partitions are merged because their centroids fall in R_{geo}. For instance, although the top-left partition is not entirely included in R_{geo}, all the biclusters in it are merged with other partitions (false positive). Similarly, although part of the bottom-left partition is included in R_{geo}, none of its biclusters are merged (false negative). In case two, biclusters in the shadowed partitions overlapping R_{geo} are merged.



(a) Case one (b) Case two

Fig. 5. Two Cases of the Mismatch Problem

We cannot avoid the mismatch problem because a bicluster formed by various contents in a partition has no geo-coordinates in itself. Thus merging should be performed on the basis of partitions rather than biclusters. However, we can evaluate the degree of the mismatch for quality control and use it as feedback to adjust the partitioning parameter δ_{cnt}.

Firstly, we consider case one in Fig. 5(a). Partitions covered by R_{geo}, either partially or entirely, form three different sets P_{fp}, P_{fn} and P_{en}. The first two contain partitions partially covered by R_{geo}: P_{fp} contains those whose centroids fall in R_{geo} while P_{fn} contains those whose centroids do not. P_{en} has partitions entirely falling in R_{geo}. For partitions in P_{fp} and P_{fn}, we estimate the number of mismatched biclusters according to the coverage rate of the partitions as follows.

$$\begin{aligned} num_{fp} &= \textstyle\sum_{pa \in P_{fp}} num(pa) \cdot (1 - cov(pa)) \\ num_{fn} &= \textstyle\sum_{pa \in P_{fn}} num(pa) \cdot cov(pa) \end{aligned} \quad (4)$$

where $num(pa)$ is the number of biclusters in partition pa and $cov(pa)$ is the percentage of pa covered by R_{geo}. Thus num_{fp} is the estimated number of additional biclusters (false positives) used in the merging phase while num_{fn} is the estimated number of missed biclusters (false negatives) falling in R_{geo} but omitted in the merging phase. Next we compute the estimated number of biclusters in R_{geo} by

$$num_{tp} = \textstyle\sum_{pa \in P} num(pa) \cdot cov(pa) \quad (5)$$

where $P = P_{fp} \cup P_{fn} \cup P_{en}$. With the above equations, we can evaluate the quality of the PM scheme by approximating the mismatch rate as follows.

$$rate_{mis} = \frac{num_{mis}}{num_{tp}} = \frac{num_{fp} + num_{fn}}{num_{tp}} \quad (6)$$

For case two in Fig. 5(b), P_{fn} is empty because biclusters in any partition overlapping the geographic region R_{geo} are merged. However, the false positive partition set, denoted by P'_{fp}, now contains all partitions partially overlapping R_{geo}, including those with centroids falling outside R_{geo}. The mismatch rate in this case is as follows.

$$rate_{mis} = \frac{num'_{fp}}{num_{tp}} = \frac{\sum_{pa \in P'_{fp}} num(pa) \cdot (1 - cov(pa))}{num_{tp}} \quad (7)$$

VI. System Implementation

Based on the PM scheme, we build a system called Vesta to browse and explore the top-ranked summaries interactively.

A. System Architecture

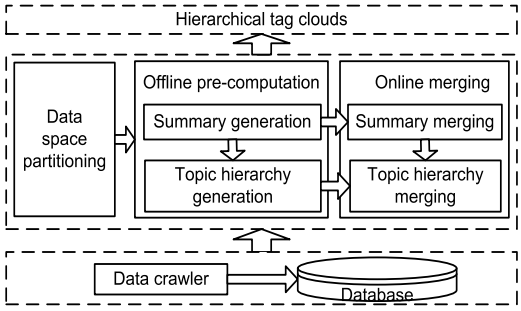


Fig. 6. System Architecture

Fig. 6 shows our system architecture, consisting of data crawling, processing and visualization components from bottom to top. The data crawling component keeps crawling geo-coded social network contents, cleaning and storing the data in the database. The processing component is the core of the system, including three parts which perform data partitioning, offline pre-computation and online merging respectively. The data partitioning part starts splitting the data into partitions after the crawler finishes preparing the contents of each day. Offline pre-computation is then performed to generate summaries and topic hierarchies for the partitions. Note that our PM scheme makes it possible that the pre-computation can be done for different partitions in parallel. The summaries and topic hierarchies in corresponding partitions will be merged respectively in the online merging part once users specify the spatiotemporal parameters. The merged summaries are finally passed to the visualization component to be displayed as hierarchical tag clouds according to their corresponding tag hierarchies generated by merging topic hierarchies.

Although our system performs the partitioning and pre-computation on a daily basis which leads to a one-day delay, we can reduce the delay to make Vesta a semi-realtime system. For instance, we can process the data every 3 hours and merge the results gradually until all contents of the day is processed.

B. Visual Layout

Next we illustrate the layout of the hierarchical tag clouds in the visualization component, which is built based on Google

Maps[3] and D3[4]. Recall the example in Fig. 1, the first-level tags of the hierarchical tag clouds will be displayed first in the largest font size. As users zoom in, tags in subsequent levels of each tag cloud will be placed near and around their first-level tags in smaller size level by level. To reduce the possibility of overlap among tags in different tag clouds and leave enough space to arrange tags of the same tag cloud close to each other, we scatter the first-level tags evenly in two concentric circular orbits. The radii of the two orbits can be adjusted to reduce the overlap among tag clouds as much as possible. Collision detection is performed to help determine the positions of the tags in subsequent levels when they are placed around the corresponding first-level tags.

VII. Experimental Study

A. Data Sets and System Environment

The experiments are conducted over the real-world geo-coded tweets crawled using the Twitter Streaming API[5]. Each tweet is associated with a pair of latitude-longitude coordinates and a time. On average, we can receive 4.1M to 4.3M (million) geo-coded tweets every day, among which about 0.7M are in English. The four data sets used in the experiments contains 0.1M, 0.7M, 2.5M and 4.3M tweets respectively. As around 1% of tweets are geo-coded[6], 4.3M is a proper approximation of the number of geo-coded tweets published daily, given that 400M tweets are generated altogether per day.

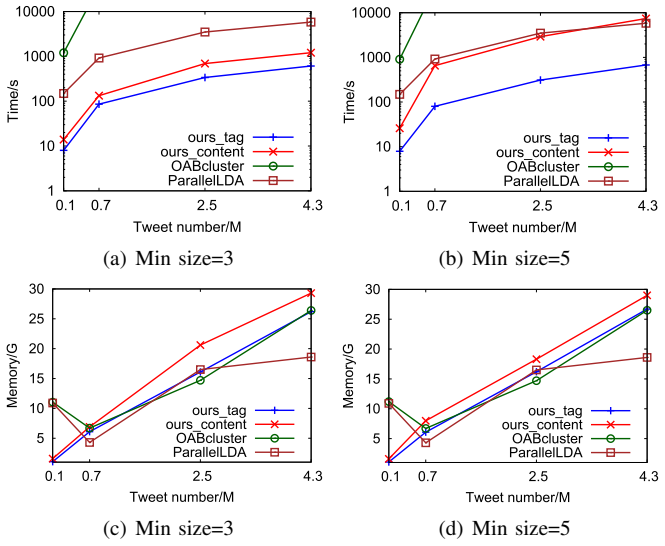
All the source code is written in Java, and the Twitter data is stored in MySQL 5.1.60. The experiments were conducted on Windows Server 2003 Enterprise x64 Edition with 16-core 2.29GHz CPU, 64GB RAM and JRE6.

B. Comparison of Different Summarization Methods

Firstly we compare our methods *ours_tag* and *ours_content* based on FCA with two other biclustering methods (*OABicluster* [18] and *FLOC* [13], [14]) and two topic modeling methods (*LDA* [1] and *hLDA* [2], [3]). *OABicluster* was also proposed for analyzing social network data based on FCA. It generates formal concepts for each tag-content pair, without guaranteeing the bicluster quality. *FLOC* was originally proposed for expression data analysis. It first generates k initial biclusters randomly and then improves their quality iteratively, by allowing missing values which correspond to the sparsity in social network content data. It uses the residue and volume (i.e., the number of non-empty values [14]) in a gain function to measure the bicluster quality. To make *FLOC* comparable, we replace the residue in the gain function with density instead, which does not affect the main procedure. The two topic modeling methods generate topics which can also be perceived as summaries. We use MALLET[7] which has implemented *ParallelLDA* and *hLDA*, where *ParallelLDA* is a parallel version of *LDA* to accelerate the speed.

[3]https://maps.google.com/
[4]A JS library for data visualization (http://d3js.org/).
[5]https://dev.twitter.com/docs/streaming-apis
[6]http://www.scotthale.net/blog/?p=307
[7]A java package for machine learning (http://mallet.cs.umass.edu/).

For the first four biclustering methods, we set the minimum density δ_{den} from 0.5 to 1.0 and the minimum size δ_{sz} from 3 to 5. Since different values of δ_{den} do not affect the performance obviously, we only report the results with δ_{den} equal to 0.8 and δ_{sz} equal to 3 and 5. Note that δ_{den} and δ_{sz} do not apply to *ParallelLDA* and *hLDA*. The execution time and memory usage of *ParallelLDA* in Fig. 7 are duplicated for different δ_{sz} for comparison purpose only. For *ParallelLDA*, we set the topic number to 100 and thread number to 4. For *hLDA*, we set the level number of the topic hierarchy to 5. The iteration numbers for both are set to 1000.



(a) Min size=3 (b) Min size=5

(c) Min size=3 (d) Min size=5

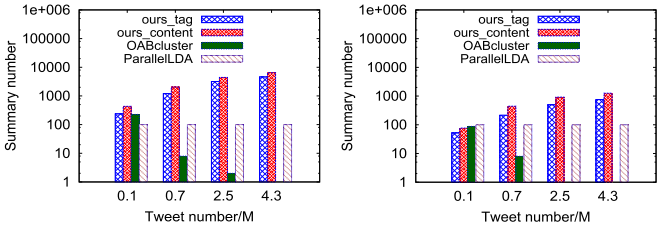
Fig. 7. Performance Comparison

1) Performance: Fig. 7 shows the performance of the above methods over four data sets with different number of tweets. We terminated *FLOC* and *hLDA* because they cannot finish within 10 hours for any data set. They run slowly due to the iterative nature. The density and volume parts of the gain function also decelerate *FLOC* probably because the density part tends to shrink the bicluster size while the volume favors larger size. To accelerate *FLOC* and make the densities of the initial biclusters increase faster, we remove the volume part and denote the modified *FLOC* by *FLOC_modified*. Although we see some improvements, *FLOC_modified* still cannot finish within 10 hours. *hLDA* is even slower and cannot finish the first 10 of 1000 iterations within 10 hours. Although we show later that *hLDA* works for the offline topic hierarchy generation in our system where it usually handles only thousands of tweets for a partition, *hLDA* in itself cannot scale to larger data sets containing hundreds of thousands of tweets. Thus *hLDA* cannot be applied to large amounts of social network data for topic hierarchy generation directly. We do not report the results for *FLOC*, *FLOC_modified* and *hLDA* here.

Fig. 7(a) and 7(b) show the execution time of the other four methods. When δ_{sz} is 3, our proposed methods *ours_tag* and *ours_content* outperform *ParallelLDA* by almost an order of magnitude. When δ_{sz} is 5, the performance of *ours_tag* almost remains unchanged while the execution time of *ours_content*

approaches that of *ParallelLDA* for tweet number larger than 0.1M. Note that *ParallelLDA* is set to generate only 100 topics. It will take more time if generating as many as *ours_content* does. For both values of δ_{sz}, *OABicluster* can only finish running within 10 hours when the tweet number is 0.1M.

Fig. 7(c) and 7(d) show the memory usage. Almost all the methods consume more memory when the tweet number gets larger, although some drops are observed for *OABicluster* and *ParallelLDA* when the tweet number comes to 0.7M. *ParallelLDA* uses less memory than other methods when the tweet number is 4.3M. This is probably because we set the topic number to 100 which is a relatively small value. The memory usage of our proposed methods increases linearly, which brings up concerns that they may not fit into memory given even larger data sets. We will show later that our PM scheme converts this memory-based problem to a disk-based problem so that memory usage becomes manageable.



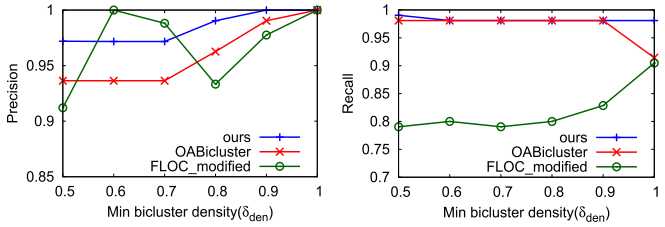
(a) Min size=3 (b) Min size=5

Fig. 8. Summary Detection Capability Comparison

2) Summary Detection Capability: Fig. 8 shows the number of biclusters/topics each method can generate, which to some extent reflects their capability of detecting various summaries. Since *OABicluster* failed to finish within 10 hours for the other three data sets, we report the number of biclusters it had generated when we terminated it. We omit *FLOC* and *FLOC_modified* again since they cannot finish running within 10 hours for any data set. Upon termination, *FLOC* generated no bicluster with density larger than 0.5 while *FLOC_modified* only generated several biclusters with density larger than 0.8 for the data set having 0.1M tweets. As the number of tweets increases, it is intuitive that more summaries or topics will be covered. Our methods conform to this intuition and generate more biclusters for larger data sets. *ParallelLDA* generates 100 topics for all data sets since we set the topic number to 100.

We also conduct experiments given δ_{den} set to 0.5 to 1.0 over the 0.1M tweets where *OABicluster* can finish running within 10 hours. *OABicluster* generates slightly more than or comparable to the number of biclusters our methods generate when δ_{den} ranges from 0.5 to 0.7. The number of biclusters generated by *OABicluster* decreases rapidly when δ_{den} increases from 0.8 to 1.0, indicating that it tends to miss many high-density biclusters. The number of biclusters generated by our methods does not drop obviously as δ_{den} increases. Because of space limitations, we omit to present the figures. In addition, both *OABicluster* and *FLOC* may lead to cases that a single bicluster covers multiple unrelated summaries for small δ_{den}. For instance, given two biclusters $(\{t_1, t_2, t_3\}, \{c_1, c_2, c_3\})$ and $(\{t_4, t_5, t_6\}, \{c_4, c_5, c_6\})$, both with density equal to 1, *OABicluster* and *FLOC* may generate a bicluster $(\{t_1, t_2, t_3, t_4, t_5, t_6\}, \{c_1, c_2, c_3, c_4, c_5, c_6\})$ with density equal to 0.5 but covering two unrelated summaries. Our methods can avoid this problem since they first generate "full-density" biclusters and related biclusters will be merged if the density of the resultant bicluster is larger than δ_{den}.

3) Precision and Recall: Evaluating the quality of summaries of social network contents quantitatively is often hard because of the huge number of contents and lack of predefined summaries or topics to compare against. Thus, we try to sample a small set of tweets by hand and predefine summaries using the tweets in order for the evaluation. Specifically, we manually select 82 out of the 0.1M tweets discussing 4 different topics to verify whether these methods can generate proper summaries for the topics. The tweets fall into 4 groups according to the topics they belong to. Common tags in all tweets of a group are chosen as the summary (or ground truth) of that group. 18 other randomly selected tweets are added as noise. All the 100 tweets are used to generate biclusters using different methods and the results are compared with the group truth. The number of true positive tags and tweets is divided by that of all positive tags and tweets to obtain the precision, and the former again is divided by the number of all tags and tweets in the ground truth to obtain the recall. Since *ParallelLDA* and *hLDA* cannot find the related tweets of the generated topics, we exclude them from the comparison.



(a) Precision (b) Recall

Fig. 9. Precision and Recall

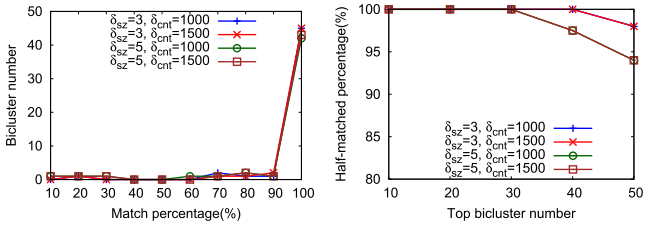
Fig. 9 shows the results when δ_{sz} is 3 and δ_{den} varies from 0.5 to 1. Since the results of *ours_tag* and *ours_content* overlap, we denote them as *ours* for simplicity. The precision and recall of our methods are stable and larger than those of other methods for most δ_{den} values. The recall of *OABicluster* is close to ours but drops suddenly when δ_{den} comes to 1. The average precision of *FLOC_modified* is competitive, however, the precision fluctuates greatly and the recall is relatively low. Besides, *FLOC_modified* does not guarantee to find all four topics every time. We do not report *FLOC* as it often discovers only one topic and mixes other topics together.

C. Partition-and-Merge Scheme Evaluation

Next we evaluate the effectiveness of our PM scheme, including assumption validation and mismatch evaluation.

1) Assumption Validation: We validate the assumption proposed in section V-B1 that globally interesting summaries are also interesting in certain partitions. In terms of biclusters, those generated without partitioning the data space and highly ranked can also be generated in some partitions of the data space. The validation can prove the validity of our PM scheme.

The validation is performed over the data set of 0.7M tweets. We first partition the data space and generate biclusters by running *ours_tag* and *ours_content* for each partition. δ_{cnt}, the largest number of contents in a partition, is set to 1000 or 1500. The biclusters in all partitions form set B_{par}. Then we generate biclusters, which form set B_{no}, directly without partitioning and test whether the most interesting biclusters in B_{no} also exist in B_{par}. To find the most interesting biclusters in B_{no}, we rank them according to our proposed score function with p set to 0. δ_{sz} for biclusters in B_{par} and B_{no} is set to the same value which is 3 or 5. We do not set δ_{den} as all biclusters in B_{par} and B_{no} are "full-density".



(a) Number of top biclusters (b) Half-matched percentage

Fig. 10. Assumption Validation

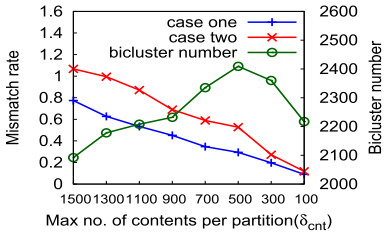


Fig. 11. Mismatch Evaluation

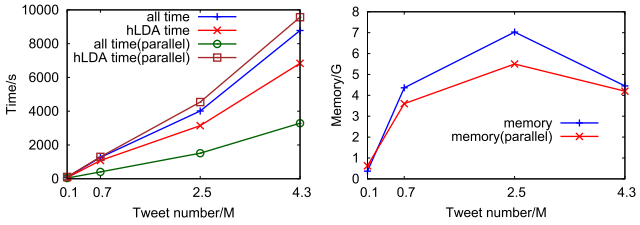
Fig. 10 validates our assumption. In Fig. 10(a), we choose the top 50 biclusters from B_{no} and compare them against biclusters in B_{par}. For each bicluster b from the top 50, we find a bicluster b' in B_{par} which shares the largest number of common tags with b. The match percentage is the ratio between the number of common tags and the number of tags in b, reflecting to what extent an interesting summary generated without partitioning can also be discovered when partitioning is performed. Fig. 10(a) shows, for different parameter settings, most of the top 50 biclusters from B_{no} can find a bicluster in B_{par} which shares all the tags in the top biclusters, indicating that many of them can be rediscovered using the partitioning scheme. A high match percentage between 50% and 100% can also indicates that a summary is quite likely to be rediscovered. Fig. 10(b) shows the percentage of the top k biclusters that have match percentage more than 50% when k is set to 10, 20, 30, 40 and 50. All the top 30 biclusters have match percentage more than 50%. As k increases, the half-matched percentage begins to decrease. However, there are still more than 90% of the top 50 biclusters whose match percentage is over 50%, meaning summaries associated with more than 45 out of 50 biclusters are highly likely to be discovered using the partitioning scheme. We also note that the results would be even better if biclusters in B_{par} are merged.

2) Mismatch Evaluation: Below we take a fixed geographic region containing 0.1M tweets as an example to explain the mismatch problem. Fig. 11 shows the mismatch rate $rate_{mis}$ and bicluster number as δ_{cnt} varies when $\delta_{sz} = 3$ and $\delta_{den} = 1$. Note that $rate_{mis}$ for either case of the mismatch in Fig. 5 can be larger than 1, indicating that a severe mismatch occurs.

From Fig. 11, we can see that $rate_{mis}$ for either mismatch case decreases as δ_{cnt} drops and that the first case leads to a smaller $rate_{mis}$. There are three stages for the decrease, namely when δ_{cnt} is larger than 700, between 700 and 500, and smaller than 500, where more obvious is the second stage. When δ_{cnt} decreases from 1500 to 700, $rate_{mis}$ reduces in a relatively fast speed, which means that any value between 1500 and 700 may not be proper for δ_{cnt} since $rate_{mis}$ is not stable enough. When δ_{cnt} comes from 700 to 500, $rate_{mis}$ decreases less dramatically, indicating it is a reasonable value range for δ_{cnt}. After δ_{cnt} drops below 500, $rate_{mis}$ decreases rapidly again. The instability of $rate_{mis}$ in this stage is probably because δ_{cnt} becomes too small. A very small $rate_{mis}$ is not always meaningful. It may cause loss of biclusters due to very small partitions, which is reflected by the green bicluster number line in Fig. 11. Another strategy is to choose a value, 500 in Fig. 11 for instance, for δ_{cnt} when the number of biclusters reaches its peak. This is also consistent with choosing between 700 and 500, and more practical in case the second stage is less detectable. Consequently, users do not have to set δ_{cnt} blindly by themselves. As δ_{cnt} can be determined automatically by conducting the mismatch evaluation, different values of δ_{cnt} can be set for different geographic regions adaptively.

D. System Scalability Analysis

Vesta adopts the PM scheme which converts the in-memory biclustering to a disk-based approach and makes it possible to parallelize the approach. Next we analyze the system scalability for the partitioning and merging phases.



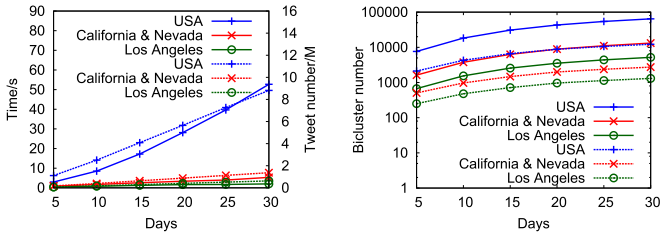
(a) Time (b) Memory

Fig. 12. Offline Scalability

1) Offline Scalability: The offline partitioning phase includes data space partitioning and pre-computation of biclusters and topic hierarchies. We plot the execution time and memory usage over different data sets in Fig. 12 when δ_{sz}, δ_{cnt} and the number of iterations for *hLDA* are set to 3, 1000, 1000 respectively. The blue line shows the execution time increases almost in proportion to the data set size while most of the time is consumed by *hLDA* (the red line) to generate topic hierarchies. The green line indicates a significant improvement when we parallelize the partitioning phase using 4 threads. Because we simply add up the time consumed by *hLDA* in all threads without considering the overlap, the *hLDA* time under parallelism (the brown line) gets very large. This indicates

that *hLDA* can be parallelized to greatly accelerate the process. Through parallelism, the overall time of the partitioning phase reduces from two and a half hours to less than an hour for the data set with 4.3M tweets, which approximate the number of geo-coded tweets published every day in reality.

Fig. 12(b) shows the memory usage which decreases greatly compared with that in Fig. 7(c) and 7(d). The memory usage does not always increase as the data set gets larger. Note that the amount of memory recorded here is the peak value which is mainly caused by *hLDA* when faced too many tweets as input. That value often drops after a short while during execution. Again, the parallelism also leads to less memory usage. Thus Fig. 12 indicates that the offline partitioning phase can scale to even larger data sets especially through parallelism.



(a) Time (b) Bicluster number decrease

Fig. 13. Online Scalability

2) Online Scalability: The online merging phase is done at runtime when users specify the spatiotemporal ranges. The solid lines in Fig. 13(a) show the response time for three geographic areas on different scales w.r.t. various time ranges. The dotted lines show the corresponding number of tweets involved in certain ranges. δ_{den} is set to 0.5, which is the lower bound density of merged biclusters. Given a geographic area, the response time increases linearly with the number of days (also the number of biclusters to merge or tweets). The response time is often small (e.g., 2 seconds for L.A. or 5 seconds for California & Nevada over 30 days) when the geographic area is on a city or state scale. When it comes to countries such as the entire USA, the response time is relatively longer because of the huge number of tweets.

Although the number of biclusters to merge is often large, that of the merged biclusters decreases greatly. This is exhibited in Fig. 13(b) where solid (or dotted) lines represent the number before (or after) merging, showing that this number can be reduced by $63\% \sim 81\%$. It means that a large part of biclusters from different partitions can be merged together and thus validates again the effectiveness of our PM scheme.

VIII. Conclusion

In this paper, we proposed a system called Vesta which enables interactive exploration of different regions by summarizing and browsing social network contents via hierarchical tag clouds. We proposed to generate summaries by biclustering the contents based on FCA and then extended the approach by introducing a disk-based PM scheme for better scalability. For visualization purpose, we adopted hLDA to generate topic hierarchies and merge them to form a tag hierarchy for each summary. The experimental study demonstrated the efficiency and effectiveness of our methods. Future work includes the extension of Vesta to a truly realtime system through incremental update and the emerging event prediction in SNSs.

Acknowledgement

This work was partially supported by the FRC Grant R-252-000-486-112 and the SeSaMe Centre sponsored by the Singapore NRF under its IRC@SG Funding Initiative and administered by the IDMPO.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, 2003.
[2] D. M. Blei, T. L. Griffiths, M. I. Jordan, and J. B. Tenenbaum, "Hierarchical topic models and the nested chinese restaurant process," in *NIPS*, 2003.
[3] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *J. ACM*, vol. 57, no. 2, 2010.
[4] D. Ramage, S. T. Dumais, and D. J. Liebling, "Characterizing microblogs with topic models," in *ICWSM*, 2010.
[5] B. Ganter, R. Wille, and R. Wille, *Formal concept analysis*. Springer Berlin, 1999.
[6] J. Poelmans, P. Elzinga, S. Viaene, and G. Dedene, "Formal concept analysis in knowledge discovery: A survey," in *ICCS*, 2010.
[7] Y. Cheng and G. M. Church, "Biclustering of expression data," in *ISMB*, 2000.
[8] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Trans. Comput. Biology Bioinform.*, vol. 1, no. 1, 2004.
[9] A. W. Rivadeneira, D. M. Gruen, M. J. Muller, and D. R. Millen, "Getting our head in the clouds: toward evaluation studies of tagclouds," in *CHI*, 2007.
[10] K. Bielenberg and M. Zacher, "Groups in social software: Utilizing tagging to integrate individual contexts for social navigation," *Master Thesis*, 2005.
[11] M. Dubinko, R. Kumar, J. Magnani, J. Novak, P. Raghavan, and A. Tomkins, "Visualizing tags over time," in *WWW*, 2006.
[12] J. Hartigan, "Direct clustering of a data matrix," *Journal of the american statistical association*, vol. 67, no. 337, 1972.
[13] J. Yang, W. Wang, H. Wang, and P. S. Yu, "delta-clusters: Capturing subspace correlation in a large data set," in *ICDE*, 2002.
[14] J. Yang, H. Wang, W. Wang, and P. S. Yu, "Enhanced biclustering on expression data," in *BIBE*, 2003.
[15] M. Rege, M. Dong, and F. Fotouhi, "Co-clustering documents and words using bipartite isoperimetric graph partitioning," in *ICDM*, 2006.
[16] Y. Song, S. Pan, S. Liu, F. Wei, M. X. Zhou, and W. Qian, "Constrained coclustering for textual documents," in *AAAI*, 2010.
[17] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic coclustering," in *KDD*, 2003.
[18] D. Gnatyshak, D. I. Ignatov, A. Semenov, and J. Poelmans, "Gaining insight in social networks with biclustering and triclustering," in *BIR*, 2012.
[19] D. M. Blei and M. I. Jordan, "Modeling annotated data," in *SIGIR*, 2003.
[20] C. Lin and Y. He, "Joint sentiment/topic model for sentiment analysis," in *CIKM*, 2009.
[21] Z. Ma, A. Sun, Q. Yuan, and G. Cong, "Topic-driven reader comments summarization," in *CIKM*, 2012.
[22] B. Dai, E. Lim, and P. Prasetyo, "Topic discovery from tweet replies," in *MLG*, 2012.
[23] C. Li, A. Sun, and A. Datta, "Twevent: segment-based event detection from tweets," in *CIKM*, 2012.
[24] G. Gierz, K. Hofmann, K. Keimel, J. Lawson, M. Mislove, and D. Scott, *Continuous lattices and domains*. Cambridge University Press, 2003, vol. 93.
[25] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, 1975.
[26] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, 1974.