# Evolutionary Bayesian Rose Trees

Shixia Liu, Xiting Wang, Yangqiu Song, and Baining Guo

**Abstract**—We present an evolutionary multi-branch tree clustering method to model hierarchical topics and their evolutionary patterns over time. The method builds evolutionary trees in a Bayesian online filtering framework. The tree construction is formulated as an online posterior estimation problem, which well balances both the fitness of the current tree and the smoothness between trees. The state-of-the-art multi-branch clustering method, Bayesian rose trees, is employed to generate a topic tree with a high fitness value. A constraint model is also introduced to preserve the smoothness between trees. A set of comprehensive experiments on real world news data demonstrates that the proposed method better incorporates historical tree information and is more efficient and effective than the traditional evolutionary hierarchical clustering algorithm. In contrast to our previous method [31], we implement two additional baseline algorithms to compare them with our algorithm. We also evaluate the performance of the clustering algorithm based on multiple constraint trees. Furthermore, two case studies are conducted to demonstrate the effectiveness and usefulness of our algorithm in helping users understand the major hierarchical topic evolutionary patterns in text data.

## 1 Introduction

Due to the wide availability of large amounts of streaming text data and their usefulness for decision-making, there is an increasing need to understand the entire picture of topics of interest and their relationships over time. For example, a Microsoft public relations analyst wants to investigate a number of important Microsoft-related events that occurred in the first half of 2013: Xbox One was announced, Motorola sued the company, and Windows 8 was released. To do so, the public relations analyst examines the news stream to understand the major topics in the stream and how they have changed and influenced each other over time. In this way, the analyst can learn whether their public relations strategies have succeeded.

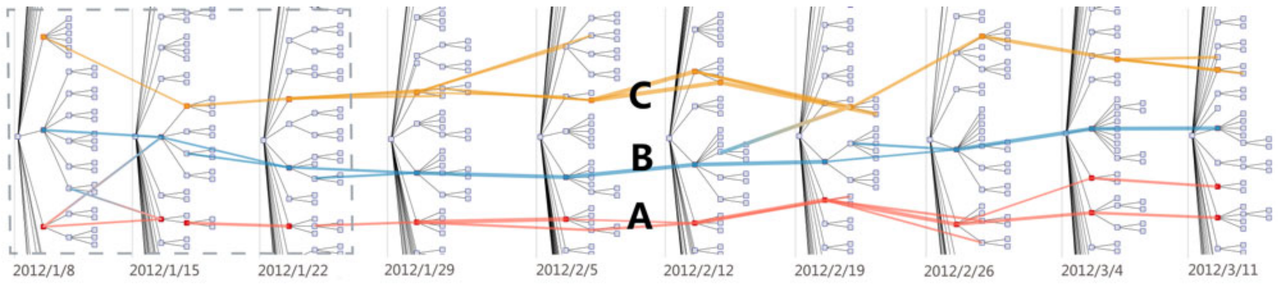
In many applications, topics are naturally organized in a hierarchy and the hierarchy often evolves over time [12]. Consequently, there have been some initial efforts to model such evolving hierarchies. The state-of-the-art approach, evolutionary hierarchical clustering [12], aims to generate evolving binary trees to organize the topics at different times. However, the binary trees may fail to provide interpretable topic results since most of the topic trees in real world applications are not binary [11], [31]. They are usually the trees with an arbitrary number of children at each nonleaf node, which are referred to as multiple branch trees. It is therefore important to effectively learn an evolving multi-branch tree representation, providing users a consistent view of content transitions.

- *S. Liu is with School of Software, Tsinghua University, Beijing, China. E-mail: shixia@mail.tsinghua.edu.cn.*
- *X. Wang is with Tsinghua University. E-mail: thu.xt.wang@gmail.com.*
- *Y. Song is with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. E-mail: yqsong@illinois.edu.*
- *B. Guo is with Microsoft Research Asia and Tsinghua University, Beijing, China. E-mail: bainguo@microsoft.com.*

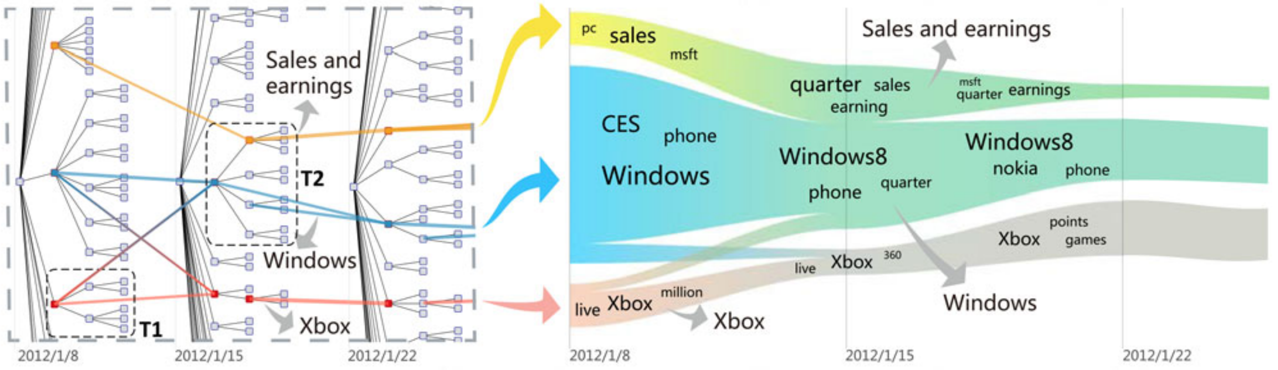
*Manuscript received 16 Nov. 2013; revised 4 May 2014; accepted 21 July 2014. Date of publication 20 Nov. 2014; date of current version 27 Apr. 2015.*
*Recommended for acceptance by E. Keogh.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TKDE.2014.2373384*

In this paper, we define and study the problem of mining evolving multi-branch topic trees inside a text stream, as well as their evolutionary patterns over time. Specifically, we take a news dataset as an example to illustrate the basic idea. The dataset contains 66,528 news articles collected from Bing News [3] using the query word "Microsoft." Fig. 1a shows part of the evolving topics and their hierarchical structures extracted from this dataset. The three labeled topics are "xbox"(A), "windows"(B), and "sales and earnings"(C). We align the corresponding topics across different trees according to their content similarity. From the alignment edges, we can see the three topics are quite stable during this time period, with a few splitting/merging relationships between them over time. With such evolutionary trees and their visual representation in Fig. 1a, users can easily examine: 1) the *evolution of multi-branch trees* and their *content alignment* over time; and 2) the topics of interest and their *evolving patterns* (e.g., splitting/merging) at different levels of these trees.

To better understand the evolving patterns of user-selected topics, we leverage a dynamic topic visualization technique, TextFlow [13], to illustrate the topic merging/splitting patterns. In this visualization, a river flow metaphor is adopted to illustrate topic evolution over time (Fig. 1b). Each colored layer represents a topic. The varying layer height along the horizontal axis represents the number of documents for the topic at each time point. Like a river flow in the real world, the topic flow can either be split into several branches when the corresponding topic splits, or combined with several other branches into one layer when the corresponding topics merge together. Fig. 1b shows the splitting/merging patterns of topics "xbox," "windows," and "sales and earnings" from Jan. 8 to Jan. 28. Topics "windows" and "sales and earnings" merged in the week of Jan. 15 when Microsoft reported its quarterly revenue. To discover more information about the merging, we browsed related news. Some news items reported on the "sales and earnings" of "windows." For example, one of the articles has the title "Windows sales slowdown as Microsoft reports Q2 revenue up 5 percent." These two topics split in the next



(a) Ten evolving trees from Jan. 8 to Mar. 17 and the highlighted topics "xbox"(A), "windows"(B) and "sales and earnings"(C)



(b) TextFlow visualization to show splitting/merging patterns of topics "xbox," "windows," and "sales and earnings" (Jan. 8 - Jan. 28)

Fig. 1. Evolutionary trees of Bing news data (from Jan.8, 2012 - Jul. 21, 2012). We grouped the data by week over the 28-week period. Accordingly, 28 trees were generated. The average tree depth was 4, the average internal node number was 99, and the average node number of the first level was 21. Here we select part of the evolving trees from Jan. 8 to Mar. 17.

week as the association became weaker. Another interesting pattern is that part of the "windows" topic split itself from the main topic in the first week and then joined "xbox" in the next week. The major reason is that Dave Culter, the father of Windows NT, shifted his focus to Xbox and was working "to extend xbox beyond its status as a gaming platform."

Motivated by this example, we aim to generate a sequence of multi-branch topic trees that preserves both the smoothness and fitness. Specifically, each tree in the sequence should be similar in structure and content to the one at the previous time point if the data does not deviate from historical expectations (smoothness). It also needs to well describe the document distribution at that time point (fitness). We define the trees that satisfy the above two requirements as consistent. However, it is quite challenging to achieve the desired results. First, it is not trivial to generate evolving multi-branch tree representations and to model their evolutionary patterns over time. Although the state-of-the-art multi-branch clustering methods [11], [22] can generate a topic tree with a high fitness value, they cannot guarantee the smoothness between trees. We define the tree distance of two nodes in a tree as the shortest path between them in that tree [12]. With this definition, one way to solve this problem is to minimize the sum of the tree distance differences between any two corresponding node pairs at two consecutive time points. This method can improve the smoothness between trees to some extent. However, it may fail to reconstruct an optimized tree for the current time point since the parent-child relationships are lost. Second, online documents (e.g., news articles) arrive regularly and thus are usually large in number. Since the time complexity of tree-based algorithms is of polynomial time [22], it is very time-consuming to generate a sequence of topic trees that well balances the fitness and smoothness.

To tackle these challenges, we propose an evolutionary Bayesian rose tree (EvoBRT) algorithm. This work is an extension of our previous work described in [31]. Previously, we described how EvoBRT can automatically generate a sequence of consistent topic trees based on one constraint tree. In this paper, we focus on EvoBRT's analytic lifecycle, especially on its application to large document corpora and the effectiveness of the multi-branch tree clustering algorithm based on multiple constraint trees. Three key aspects of EvoBRT are described in this paper.

First, to preserve both the fitness and smoothness between adjacent trees, we formulate the evolutionary clustering problem as a Bayesian online filtering algorithm. In this framework, a Bayesian rose tree (BRT) [11] is adopted to build multi-branch trees, as well as to maintain a high fitness for human understanding. A tree prior is introduced in the tree learning framework to preserve the smoothness, which is formulated as a Markov random field (MRF).

Second, a set of experiments are conducted to demonstrate that our algorithm is more effective and efficient than the state-of-the-art evolutionary hierarchical clustering algorithm. Specifically, we implemented two evolutionary multi-branch tree clustering algorithms based on one constraint tree and multiple constraint trees, respectively, and then evaluated their performance under different metrics. We also implemented three baseline methods and compared them with our algorithm based on one constraint tree.

Third, we conducted two case studies to demonstrate the effectiveness and usefulness of our algorithm, especially in support of understanding and analyzing a document collection from the global evolutionary structure to the local salient features.

## 2 Related Work

### 2.1 Constrained Hierarchical Clustering

Based on the constraint type, existing constrained hierarchical clustering can be classified into two categories: pairwise approaches and triplewise approaches. Pairwise approaches incorporate the constraints in the form of must-links and cannot-links, which indicate that two samples must or cannot be in the same cluster [14], [23]. Since must-links and cannot-links do not consider hierarchical information, these methods may fail to characterize the hierarchical document distribution.

Triplewise approaches incorporate triple constraints (e.g., two samples must be combined before the other sample is combined with either of them) among the data to generate clusters. Existing methods consider two different ways to use triple constraints, *metric-based* approaches and *instance-based* approaches [7]. Metric-based approaches learn a distance or similarity metric from the constraints and then embed the metric in the clustering process [7], [19], [29], [40]. Instance-based approaches follow all triplewise constraints in the bottom-up merging process and will fail to generate a hierarchy if one of them is violated [7], [20], [39].

In contrast to the above algorithms, which are designed for building a static binary tree, our algorithm aims to generate evolving multi-branch hierarchies. A multi-branch tree contains both triples and fans, so the existing approaches do not work since they cannot handle fans. In addition, our algorithm automatically generates constraints for each time $t$ based on the tree structure at $t-1$, while the constraints in constrained hierarchical clustering are predefined.

### 2.2 Evolutionary Topic Analysis

Recent efforts in topic analysis have focused on developing advanced machine learning algorithms to extract evolving topics, such as dynamic latent Dirichlet allocation [9] and its variations [4], [33], and hierarchical Dirichlet processes [5], [6], [36], [37], [38]. In many applications, the evolving topics may be correlated with others by various relationships over time. The most intuitive relationships are *topic correlation* [32] and *common topics* [34]. These two types of relationships can help users understand how topics co-evolve with each other. Based on the hierarchical Dirichlet processes model and incremental Gibbs sampling algorithm, Gao et al. [13], [15] proposed an approach to track and connect clusters in text data. Although the above methods have achieved some success in modeling a number of evolutionary patterns of topics, none of them focus on mining evolving trees.

The method most related to ours is the evolutionary hierarchical clustering algorithm [12]. It measures the difference between trees by the average distance between all node pairs. However the tree distance metric is not sufficient to reconstruct a tree and measure the smoothness between trees since the parent-child relationships are lost. To tackle this problem, we introduce two constraints, triples and fans, into the model to guarantee the high smoothness between topic trees. In addition, we also choose the Bayesian rose tree [11] as our base representation to discover multi-branch structures in text data instead of binary tree structures.

Researchers have also proposed a number of visualization methods to better convey complex topic mining results [13], [18], [21], [26], [35]. For example, TIARA was designed to visually illustrate individual topic evolutionary patterns over time [21], [35]. Cui et al. [13] designed a TextFlow visualization to help users better understand topic birth, death, splitting, and merging patterns. Since we are interested in the evolutionary patterns of the topics of interest (e.g., splitting/merging), we leverage the TextFlow visualization in our work.

## 3 Background

The base representation of EvoBRT is a multi-branch tree. We then follow the state-of-the-art approach, Bayesian rose tree [11] to infer the tree structure. BRT greedily estimates the tree structure based on probability $P(\mathcal{D}|T)$. Initially, each document is regarded as an individual tree: $T_i = \{\mathbf{x}_i\}$. Then the algorithm repeatedly selects two trees $T_i$ and $T_j$ and combines them into a new tree $T_m$ by a *join*, *absorb*, or *collapse* operation [11], aiming to maximize the likelihood ratio:

$$\frac{p(\mathcal{D}_m|T_m)}{p(\mathcal{D}_i|T_i)p(\mathcal{D}_j|T_j)}, \tag{1}$$

where $p(\mathcal{D}_m|T_m)$ is the likelihood of $\mathcal{D}_m$ given the tree $T_m$, and $\mathcal{D}_m = \mathcal{D}_i \cup \mathcal{D}_j$ represents all the data under $T_m$. Here the likelihood ratio rather than likelihood is employed because the denominator makes Eq. (1) comparable across different choices with trees $T_i$ and $T_j$ of different sizes [17]. For a more detailed description of the BRT model, please refer to [11], [31].

## 4 Evolutionary Tree Modeling

As presented previously [31], we reintroduce the overall process of evolutionary tree construction.

### 4.1 Algorithm Overview

For each document set $\mathcal{D}^t = \{\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_{n_t}^t\}$ at time $t$, we assume there is an underlying tree $T^t$ that organizes the documents at $t$. A previous study [25] has shown that a Bayesian approach to online learning can minimally reduce the loss of information from discarding previous examples at each step. In the meanwhile, $T^t$ needs to preserve both the fitness and smoothness. Thus we formulate the tree learning procedure as a Bayesian online filtering process [25]

$$p(T^t|\mathcal{D}^t, T^{t-1}) \propto p(\mathcal{D}^t|T^t)p(T^t|T^{t-1}). \tag{2}$$

With this formulation, the new tree $T^t$ depends on both the likelihood of the current data $p(\mathcal{D}^t|T^t)$ and conditional prior $p(T^t|T^{t-1})$, which measures the difference between $T^t$ and $T^{t-1}$. Accordingly, our model considers both the fitness and historical smoothness costs, as in the evolutionary hierarchical clustering algorithm [12]. For the simplicity's sake, we first use one-step historical smoothness to illustrate the basic idea.

Directly maximizing $p(\mathcal{D}^t|T^t)p(T^t|T^{t-1})$ is intractable since there are a super-exponential number of candidate trees for $T^t$. We therefore follow the greedy construction method of BRT [11] to select two sub-trees and one of the three types of combinations (join, absorb, and collapse) to construct a larger (sub-)tree. The selection aims to maximize the following posterior test ratio:

$$\frac{p(\mathcal{D}^t_m|T^t_m)p(T^t_m|T^{t-1})}{p(\mathcal{D}^t_i|T^t_i)p(T^t_i|T^{t-1})p(\mathcal{D}^t_j|T^t_j)p(T^t_j|T^{t-1})}, \quad (3)$$

where $T^t_i$ and $T^t_j$ are two candidate sub-trees that are considered for generating $T^t_m$ using the three types of combinations, $D^t_i$ and $D^t_j$ are the corresponding document sets, and $D^t_m = D^t_i \bigcup D^t_j$.

The energy function that measures the smoothness cost of merging $T^t_i$ and $T^t_j$ given $T^{t-1}$ is denoted as

$$V_{T^{t-1}}(\{T^t_i, T^t_j\} \rightarrow T^t_m), \quad (4)$$

we formulate the conditional tree prior in a recursive way

$$p(T^t_m|T^{t-1}) = p(T^t_m|T^t_i, T^t_j, T^{t-1})p(T^t_i|T^{t-1})p(T^t_j|T^{t-1}), \quad (5)$$

where

$$p(T^t_m|T^t_i, T^t_j, T^{t-1}) \triangleq \frac{1}{Z}\exp(-\lambda V_{T^{t-1}}(\{T^t_i, T^t_j\} \rightarrow T^t_m)), \quad (6)$$

and $\lambda$ is the constraint weight that balances the smoothness and tree likelihood. Inspired by a simpler Markov random field defined on flat clusters in [8], we regard the conditional prior $p(T^t|T^{t-1})$ as a Gibbs distribution based on a recursively defined MRF. The energy function of the MRF can be parameterized as a sum of a set of $V_{T^{t-1}}(\{T^t_i, T^t_j\} \rightarrow T^t_m)$. With this parametrization, we rewrite Eq. (3) as

$$\frac{p(\mathcal{D}^t_m|T^t_m)}{p(\mathcal{D}^t_i|T^t_i)p(\mathcal{D}^t_j|T^t_j)} \cdot \frac{1}{Z}\exp(-\lambda V_{T^{t-1}}(\{T^t_i, T^t_j\} \rightarrow T^t_m)). \quad (7)$$

Since the first term corresponds to the BRT algorithm (Eq. (1)), the key is then to calculate the second term, the smoothness cost $V_{T^{t-1}}(\{T^t_i, T^t_j\} \rightarrow T^t_m)$.

## 4.2 Constraint Modeling

We introduce the triple- and fan-based constraints to measure the smoothness cost. A constraint tree is built accordingly for efficient computation.

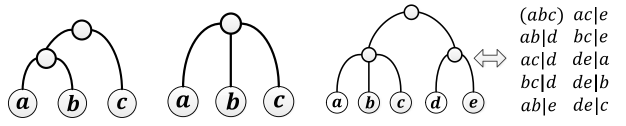
### 4.2.1 Triple, Fan, and Constraint Tree

We first introduce some preliminary definitions.

**Definition 1.** *A* ***triple*** *(Fig. 2a) is a sub-tree with three leaf nodes and two internal nodes. We denote it as* $\mathbf{ab}|\mathbf{c}$ *where* $\mathbf{a}$ *and* $\mathbf{b}$ *are the two closest leaf nodes and* $\mathbf{c}$ *is the third leaf node.*

**Definition 2.** *A* ***fan*** *(Fig. 2b) is a sub-tree with three leaf nodes and one internal node. We denote it as* $(\mathbf{abc})$ *where* $\mathbf{a}$, $\mathbf{b}$, *and* $\mathbf{c}$ *are the three leaf nodes.*

The relationship between a multi-branch tree and its related triples/fans (see Fig. 2c as an example) is very important to reconstruct a tree. As a result, we introduce



(a) Triple: $ab|c$ (b) Fan: $(abc)$ (c) Tree representation

Fig. 2. Tree representation by triples and fans.

the following lemma, which was proposed by Ng and Wormald [24].

**Lemma 1.** *A multi-branch tree* T *can be uniquely defined by a set of triples and fans.*

Based on this lemma, we measure the smoothness cost by the number of violated triples/fans between adjacent trees. A tree with $n$ leaves contains $C^3_n$ triples and/or fans. Thus, directly computing the violation number usually takes $O(n^3)$ memory and $\Omega(n^3)$ time, which is very time- and memory-consuming. To solve this problem, we have built a tree to organize all the related triples and fans. We call it a *constraint tree*.

**Definition 3.** *A* ***constraint tree*** $\tilde{T}^t$ *hierarchically organizes the triples/fans inferred from* $\mathcal{D}^t$. *It is initialized based on the previous tree* $T^{t-1}$ *and modified as the corresponding triples/fans are violated.*

### 4.2.2 Constraint Tree Initialization

The basic idea of initializing the constraint tree $\tilde{T}^t$ is to map each document in $\mathcal{D}^t$ to its most relevant topic in $T^{t-1}$. For a document that does not belong to any topic in $T^{t-1}$, a new topic is then generated at $t$. In our implementation, we propose two alternative measures to compute the similarity between $\mathbf{x}^t_i$ and $\mathcal{D}^{t-1}_m$. The first is based on the cosine similarity between documents, which is the most commonly used measure in text mining

$$sim_{cos}(\mathbf{x}^t_i, \mathcal{D}^{t-1}_m) \triangleq \cos\left(\mathbf{x}^t_i, \sum \mathbf{x}^{t-1}_j\right). \quad (8)$$

As an alternative, the prediction measure, is based on the conditional probability of $\mathbf{x}^t_i$ given $\mathcal{D}^{t-1}_m$ [10]:

$$sim_{pred}(\mathbf{x}^t_i, \mathcal{D}^{t-1}_m) \triangleq \log \int_\theta p(\mathbf{x}^t_i|\theta)p(\theta|\mathcal{D}^{t-1}_m)d\theta. \quad (9)$$

After mapping all the data $\mathbf{x}^t_i$ in $\mathcal{D}^t$ to $T^{t-1}$, we generate a new tree $\tilde{T}^t$, which is the initialization of the constraint tree. With this tree, one way to compute the smoothness cost is as follows: when building a new tree $T^t$, we compute how many triples and fans are violated. Although this method is intuitive, conflicting constraints are ignored when computing the smoothness cost. For example, given a constraint tree $\tilde{T}^{t-1}$ in Fig. 3, if we combine $a$ and $d$, two types of constraint states will be introduced. The first type is violated constraints. In this example, three triples/fans are violated. For simplicity's sake, we take the triple $ab|d$ as an example. This triple indicates that $a$ and $b$ should be combined first and then they are combined with $d$. However, if $a$ and $d$ are combined first, this triple is violated. The second type is conflicting constraints, which are the triples/fans that

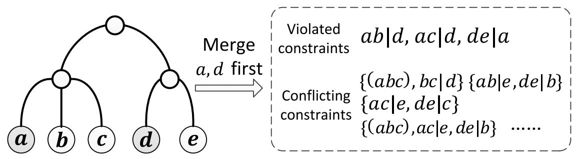


Fig. 3. Two types of constraint states.

cannot co-exist in the constraint tree. Considering fan $(abc)$, if $a$ and $d$ are combined, $(abc)$ conflicts with the triple $bc|d$ since they cannot both be in this tree.

It is not easy to compute the cost of the conflicting constraints since the relationships between them are complicated. For example, one constraint may conflict with multiple constraints. Moreover, in addition to the pairwise conflicts, there are triplewise and multiple conflicts. As shown in Fig. 3, if $a$ and $d$ are combined, then $(abc)$, $ac|e$, and $de|b$ conflict even though any two of them do not conflict with each other. As a result, it is hard to measure the corresponding cost even if we list all the conflicting constraints.

### 4.2.3 Basic Operations and Their Cost

To better measure the cost introduced by violated/conflicting constraints, we define two basic operations, MERGE and SPLIT on the constraint tree (Fig. 4).

**Definition 4.** *MERGE indicates a sub-tree $\tilde{T}_k$ forwards its own data and children to its parent $\tilde{T}_l$. Then $\tilde{T}_k$ itself is removed from the constraint tree.*

**Definition 5.** *SPLIT indicates some children of a sub-tree $\tilde{T}'_l$ redirect their parent to a newly generated child $\tilde{T}_k$ of $\tilde{T}'_l$.*

Next, we illustrate the major reason why these two operations are enough to remove the conflicts from a constraint tree and make it consistent. For simplicity's sake, we use a two-level tree as an example to illustrate the basic idea. BRT provides three types of combinations, a join, an absorb, and a collapse [11]. If sub-trees $T_i$ and $T_j$ are combined with a BRT operation that is different from the one in the constraint tree, conflicting constraints will be introduced. The basic idea of avoiding conflicts is to update the constraint tree to make it consistent with the current data organization. Assume $T_i$ and $T_j$ are combined together in an absorb operation. If they appear in the constraint tree as shown in Fig. 5B, conflicting constraints are introduced. To solve them, we change the constraint tree into the one in Fig. 5C using a MERGE operation. More examples of using the MERGE/SPLIT operation(s) to modify the constraint tree are shown in Fig. 5, which demonstrate that the changes between constraint trees can all be handled with SPLIT or MERGE operation(s).

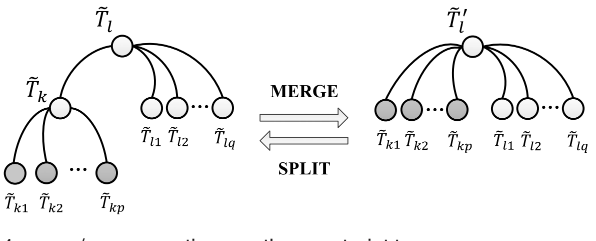

$\tilde{T}_l$
$\tilde{T}_k$
$\tilde{T}_{l1}$ $\tilde{T}_{l2}$ $\tilde{T}_{lq}$
$\tilde{T}_{k1}$ $\tilde{T}_{k2}$ $\tilde{T}_{kp}$
MERGE
SPLIT
$\tilde{T}'_l$
$\tilde{T}_{k1}$ $\tilde{T}_{k2}$ $\tilde{T}_{kp}$ $\tilde{T}_{l1}$ $\tilde{T}_{l2}$ $\tilde{T}_{lq}$

Fig. 4. MERGE/SPLIT operations on the constraint tree.

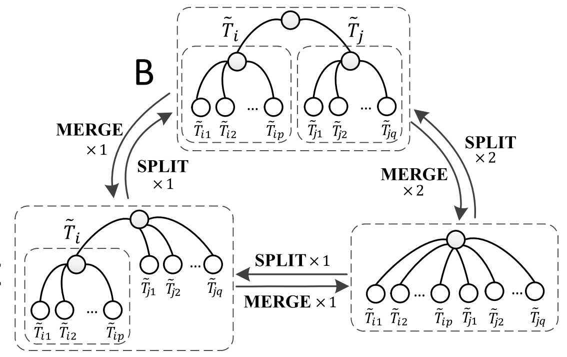

B
$\tilde{T}_i$
$\tilde{T}_j$
$\tilde{T}_{i1}$ $\tilde{T}_{i2}$ $\tilde{T}_{ip}$ $\tilde{T}_{j1}$ $\tilde{T}_{j2}$ $\tilde{T}_{jq}$
MERGE ×1
SPLIT ×1
SPLIT ×2
MERGE ×2
C
SPLIT × 1
MERGE × 1

Fig. 5. Update constraint trees by MERGE/SPLIT.

With MERGE/SPLIT, we can then measure the cost from conflicting constraints by counting the violated constraints in the two types of operations. The calculation method is given by the following theorems.

**Theorem 1.** *In a MERGE operation, only triples may be violated, and violated triples become fans. The number of violated triples is*

$$V_{\text{M}}\big(\{\tilde{T}_k, \tilde{T}_l\} \rightarrow \tilde{T}'_l\big) = \frac{\big|\tilde{\mathcal{D}}_k\big|^2 - \sum \big|\tilde{\mathcal{D}}_{ki}\big|^2}{2}\big(\big|\tilde{\mathcal{D}}_l\big| - \big|\tilde{\mathcal{D}}_k\big|\big), \quad (10)$$

*where $|\tilde{\mathcal{D}}_k|$ is the number of leaves in the tree $\tilde{T}_k$.*

The proof of the theorem is illustrated in our previous work [31]. Similarly, we have the following theorem for the SPLIT operation.

**Theorem 2.** *In a SPLIT operation, only fans may be violated, and the violated fans become triples. The number of violated fans is:*

$$V_{\text{S}}(\tilde{T}'_l \rightarrow \{\tilde{T}_k, \tilde{T}_l\}) = \frac{|\tilde{\mathcal{D}}_k|^2 - \sum |\tilde{\mathcal{D}}_{ki}|^2}{2}(|\tilde{\mathcal{D}}_l| - |\tilde{\mathcal{D}}_k|). \quad (11)$$

In the above discussion, we assume that the sub-trees to be combined are adjacent to each other. In real applications, however, the two sub-trees may not be adjacent. In such a case, we first move them to the closest common ancestor via a sequence of MERGE operations. Once they are under the same ancestor, we perform the MERGE/SPLIT operation(s) to make the related constraint tree consistent. The smoothness cost is then calculated by

$$V_{T^{t-1}}\big(\{T^t_i, T^t_j\} \rightarrow T^t_m\big) = \sum V_{\text{OPT}_l}, \quad (12)$$

where $\text{OPT}_l$ is a MERGE or a SPLIT operation.

## 4.3 Extension to Multiple Constraint Trees

In this section, we illustrate how to extend our model to incorporate multiple constraint trees in the conditional prior $p(T^t|\mathcal{D}^t, T^{t-1}, T^{t-2}, \ldots)$. One major problem with multiple constraint trees is the conflicting constraints among them, which may destroy smoothness between trees. Typically, more constraint trees will introduce more conflicts (Fig. 12). To resolve conflicting constraints among multiple constraint trees, we consistently match multiple constraint trees by extending a error-tolerant pairwise graph matching method based on graph edit distance [27], which is denoted as $f_m$.

TABLE 1
Baseline Algorithms

| Baselines | Tree Construction Method | Constraint Model |
|---|---|---|
| BinaryDistance | Evolutionary binary | Distance |
| BinaryOrder | Evolutionary binary | Order |
| MultiDistance | Evolutionary multi-branch | Distance |

Specifically, given $i$ constraint trees, $\{\tilde{T}^{t-i+1}, \dots, \tilde{T}^{t-1}, \tilde{T}^{t}\}$, we first match $\tilde{T}^{t-i+1}$ with $\tilde{T}^{t-i+2}$. The matched result is denoted as $M(t-i+1, t-i+2)$. We then match $M(t-i+1, t-i+2)$ with $\tilde{T}^{t-i+3}$ and get $M(t-i+1, t-i+2, t-i+3)$. We repeat the above matching process and get the final result $M(t-i+1, \dots, t-1, t)$, which is regard as the input constraint tree of our clustering algorithm.

## 5 Experiments

To demonstrate the performance of our algorithm, we conducted a series of experiments on several real datasets. In this section, we first introduce the baseline algorithms. Then the effectiveness of our constraint model is demonstrated. Next, we illustrate how our algorithm can well preserve both the fitness (likelihood) and smoothness. After that, we show that EvoBRT is as efficient as BRT and its variations in [22]. Finally, we evaluate how multiple constraint trees impact smoothness. The results show that our algorithm outperformed the baselines in all aspects that we compared.

In our experiments, if we do not mention the evolutionary multi-branch tree clustering method, we used $K$NN-EvoBRT ($K = 50$) due to its efficiency and relatively stable performance [22]. For each experiment, the rose tree parameters with the highest likelihood value were selected through a grid search.

### 5.1 Baseline Algorithms

We aimed to evaluate the effectiveness and efficiency of the proposed tree construction method and the triple- and fan-based (order) constraint model. To this end, we implemented three baseline algorithms based on the state-of-the-art evolutionary hierarchical clustering method [12] and the Bayesian hierarchical clustering (BHC) algorithm [16]. Table 1 shows the combinations of the tree construction method and constraint model for different baseline algorithms.

We chose BHC because it can generate more accurate and balanced hierarchies [16]. The distance constraint model in [12] was adopted as the baseline constraint model, which is defined as

$$\log p_{Dist}(T^t|T^{t-1}) \triangleq -\lambda E_{\substack{r,s\in leaves(T^t)\\ r\neq s}}\left(d_{T^t}(r,s) - d_{\tilde{T}^t}(r,s)\right)^2, \tag{13}$$

where $d_{T^t}(r,s)$ is the tree distance between $r$ and $s$, $E_{\substack{r,s\in leaves(T^t)\\ r\neq s}}$ denotes the expectation over all distinct leaves, and $\lambda$ is the constraint weight that balances the smoothness and tree likelihood.

### 5.2 Effectiveness of the Constraint Model

The goal of this experiment was to evaluate the clustering quality of the tree built by our constraint model.

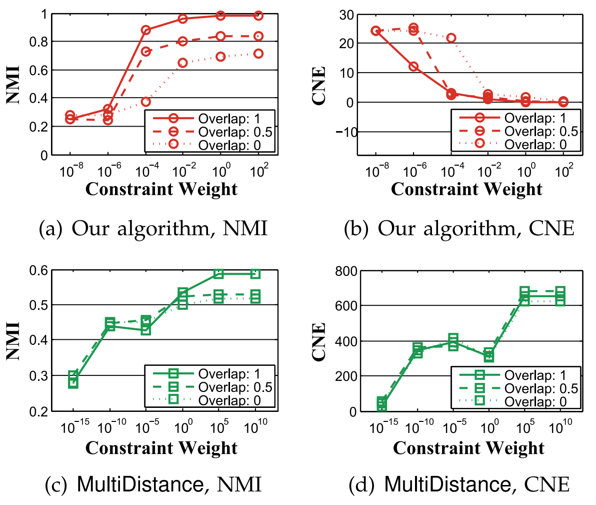


Fig. 6. Tree clustering quality comparison.

#### 5.2.1 Experimental Settings

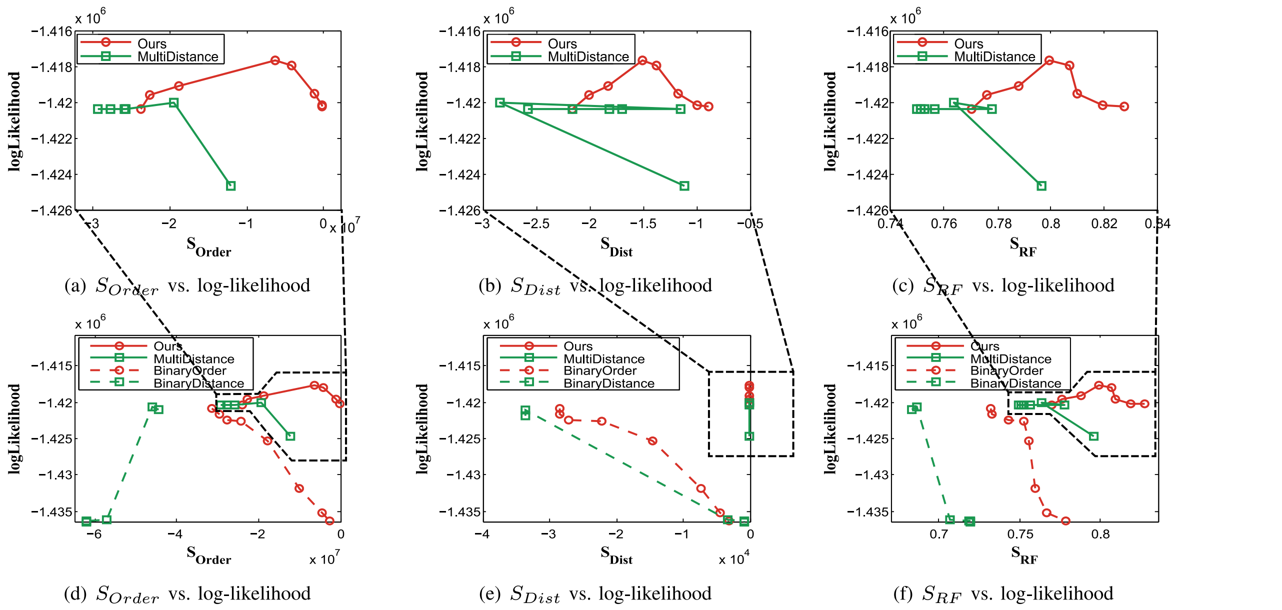
In this experiment, we used the 20 Newsgroups dataset [1]. This dataset has a ground-truth hierarchy with two levels of clusters. In each trial, we randomly sampled 2,000 documents to form a ground-truth labeled tree. We treated it as the tree at $t-1$. We then sampled 2,000 documents again and mapped them to the ground-truth labeled tree, which is the initial constraint tree in our model. The second dataset of 2,000 documents is sampled with a specific percentage of overlap with the first dataset. In our experiment, we considered five overlapping ratios that varied from 0 to 1. For each overlapping ratio, we sampled the two datasets five times and averaged the results of these five trials. The Bayesian rose tree parameter $\gamma$ was set at 0.1, and $\alpha^{(i)} = 0.01 (i = 1, \dots, |\mathcal{V}|)$.

#### 5.2.2 Criteria

We evaluated the tree clustering quality using two criteria: Normalized Mutual Information (NMI) and Cluster Number Error (CNE). NMI is the most commonly used metric to measure the clustering quality [30], but may fail in certain instances, especially when the cluster number difference is large. For example, the ground-truth contains 50 clusters, each of which consists of 20 data samples. If the algorithm builds 1,000 clusters, each of which contains only 1 data sample, then the NMI value is 0.75. To solve this problem, we introduced CNE, which measures the cluster number difference between the generated tree and the ground-truth tree at each level. The larger the CNE value, the worse the tree clustering quality. In our experiments, we evaluated the clustering quality by averaging the NMI/CNE values at different levels. A clustering result with a larger NMI value and smaller CNE value has better quality.

#### 5.2.3 Results

We compared our constraint model with the baseline constraint model to demonstrate its effectiveness in building high-quality tree clustering results. We only compared the results of our algorithm with those of MultiDistance because the ground-truth tree is multi-branch. In each experiment, we also compared the results with the overlapping ratios 0, 0.5, and 1. Fig. 6 shows how the tree clustering quality



(a) $S_{Order}$ vs. log-likelihood (b) $S_{Dist}$ vs. log-likelihood (c) $S_{RF}$ vs. log-likelihood

(d) $S_{Order}$ vs. log-likelihood (e) $S_{Dist}$ vs. log-likelihood (f) $S_{RF}$ vs. log-likelihood

Fig. 7. Comparison of the smoothness and likelihood with different constraint weights.

changed with different constraint weights $\lambda$ (Eq. (6)) for both algorithms. As indicated by the results, our algorithm is much more effective than MultiDistance. When the data overlapping ratio is 1, we can reconstruct the ground-truth labeled tree. Even if the overlapping ratio is 0, our algorithm still maintains a larger NMI (0.7) and a smaller CNE (near 0), while MultiDistance has a smaller NMI (0.5) and much larger CNE (650). In our algorithm, both NMI and CNE become better as the constraint weight increases. In MultiDistance, the NMI becomes better with the increase of the constraint weight, while the CNE gets worse with the increase of the constraint weight.

## 5.3 Tree Likelihood vs. Smoothness

In this experiment, we aimed to demonstrate that our algorithm preserves both the fitness and smoothness.

### 5.3.1 Experimental Settings

In this experiment, we used New York Times news articles (from Jan. 2006 to Jan. 2007) [2]. We grouped the data into nine segments, each of which contained 2 months of articles. We randomly sampled 1,000 documents from each time segment. To eliminate the randomness caused by sampling, we sampled the data five times and ran the experiment five times. The results were computed by averaging the results of the five trials. Parameters $\gamma$ and $\alpha^{(i)}$ were set at 0.03 and 0.0005.

### 5.3.2 Criteria

We used the likelihood to measure the fitness of a tree. We also introduced three metrics to measure the smoothness between adjacent trees.

*Order smoothness* ($S_{Order}$). This metric is defined based on the smoothness cost of our algorithm. It is the negative value of the violated triples/fans compared with the previous tree: $S_{Order} = \frac{1}{\lambda}\log(p(T^t|T^{t-1}))$.

*Distance smoothness* ($S_{Dist}$). This metric is defined based on the smoothness cost of the baseline constraint model. It measures the tree structure difference by aggregating the tree distance difference between two corresponding leaf pairs of the adjacent trees: $S_{Dist} = \frac{1}{\lambda}\log(p_{Dist}(T^t|T^{t-1}))$.

*Robinson-foulds smoothness* ($S_{RF}$). This metric is based on the widely used Robinson-Foulds distance metric for phylogenetic trees [28]

$$S_{RF} = 1 - \frac{d_{RF}(T^t, T^{t-1}(\mathcal{D}^t)) + d_{RF}(T^{t-1}, T^t(\mathcal{D}^{t-1}))}{2}, \quad (14)$$

where $T^{t-1}(\mathcal{D}^t)$ represents the constraint tree built on $T^{t-1}$ and data $\mathcal{D}^t$.

### 5.3.3 Results

Two sequences of binary trees (average tree depth: 366) and two sequences of multi-branch trees (average tree depth: 5) were built using our algorithm and the baseline algorithms, respectively. Fig. 7 shows how the smoothness scores changed with the likelihood of the four algorithms. We did a grid search of eight constraint weights for these algorithms. Since different constraint models usually need different parameters, we used two different sets of control weights in this experiment. The constraint weights for our algorithm and BinaryOrder were $\{3e^{-6}, 1e^{-5}, \ldots, 3e^{-3}, 1e^{-2}\}$. While the constraint weights for MultiDistance and BinaryDistance were $\{1e^{-25}, 1e^{-20}, \ldots, 1e^{10}\}$. The larger the constraint weight, the more emphasis was put on the smoothness factor. In each of the figures, we connected the points in the order of increasing constraint weights. Based on an analysis of the results, we have drawn the following conclusions.

First, our algorithm can generate a much smoother tree sequence than the baseline algorithms while maintaining larger likelihood scores. In addition to performing very well under the metric of $S_{Order}$, our algorithm also achieved a comparable performance under the metrics of $S_{Dist}$ and $S_{RF}$. This demonstrates that triples and fans contain all the hierarchical information of a multi-branch tree and thus the

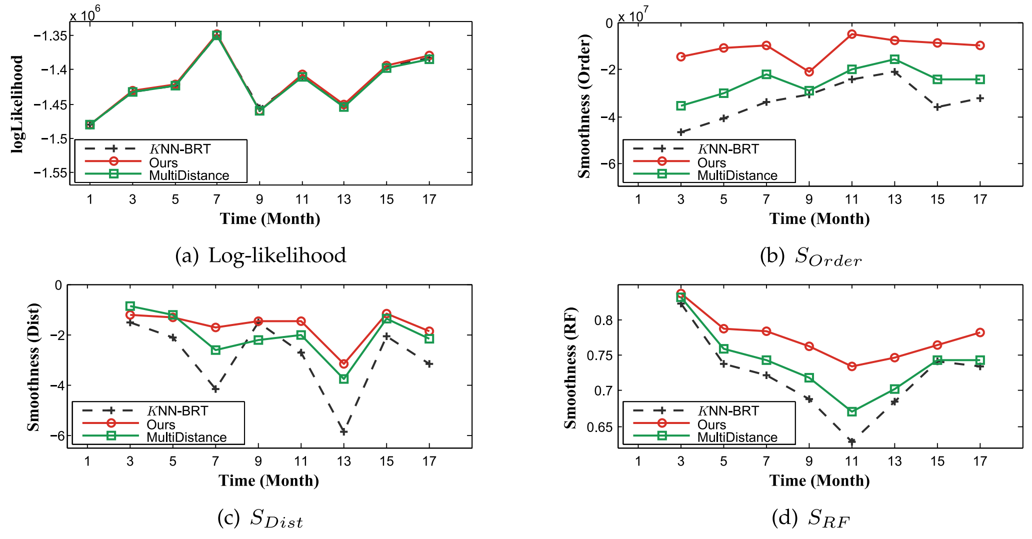


Fig. 8. Comparison of the smoothness and likelihood at different times (our algorithm and multi-branch algorithms).

cost function based on them is very effective at preserving both the smoothness and fitness. MultiDistance performed reasonably well under the metric of $S_{Dist}$. However, it failed to get a good result under the metrics of $S_{Order}$ and $S_{RF}$. This is due to the fact that tree distance constraints do not consider parent-child relationships in a tree. BinaryOrder and BinaryDistance did not perform well in terms of both the likelihood and smoothness. The lower likelihood scores indicate that the binary model does not well fit the document distribution at each time point. This result is consistent with the results shown by Blundell et al. [11]. The lower smoothness scores indicate that the binary tree model is more sensitive to noise. By examining the results, we found the structures of the generated binary trees often changed a lot between adjacent times. Furthermore, BinaryOrder generated smoother structures compared with BinaryDistance, which demonstrates that the triple- and fan-based constraint model outperformed the distance-based constraint model.

Second, the smoothness of our algorithm increases consistently with the increase of the constraint weight, while the likelihood increases at first and then decreases. This indicates that incorporating a certain amount of historical information actually helps to increase the fitness. The major reason for this is that the highly reliable triples/fans are kept in the current tree and they can help the greedy algorithm find a better solution. However, MultiDistance does not exhibit a similar pattern. Even if the constraint weight increases, the smoothness between trees is not guaranteed. This is because the baseline constraint model does not well consider multi-branch structures and does not handle conflicting constraints. Similarly, the smoothness score of BinaryOrder increases with the increase of the constraint weight while that of BinaryDistance did not, which again demonstrated that the triple- and fan-based constraints performed better.

Next, we found that our algorithm can well preserve both the smoothness and likelihood at each time point. The result was the average values of the eight trails with different constraint weights. As shown in Figs. 8b, 8c, and 8d, our algorithm outperformed MultiDistance in term of the smoothness at almost every time point. In addition, MultiDistance also performed better in preserving the smoothness than the $K$NN-BRT method, which simply performed $K$NN-BRT at each time point. As for the likelihood, our algorithm and MultiDistance were as good as $K$NN-BRT (Fig. 8a). This demonstrates again that our algorithm can preserve the smoothness between trees without sacrificing the likelihood of each tree. We further compared our algorithm with the binary-based algorithms. As shown in Figs. 9b, 9c, and 9d, our algorithm outperformed BinaryDistance and BinaryOrder in term of the smoothness at almost every time point. In addition, BinaryDistance and BinaryOrder also worked better at preserving the smoothness than the BHC method. As for the likelihood, our algorithm performed better than BinaryDistance, BinaryOrder, and BHC (Fig. 9a). This again demonstrates that the binary tree model does not well fit the document distribution at each time.

## 5.4 Efficiency

In this section, we first demonstrated that our algorithm was more efficient than MultiDistance and BinaryDistance, and comparable with BindaryOrder. Then we compared the run time of our algorithm with different constraint weights. The experimental results showed that EvoBRT was as efficient as BRT.

### 5.4.1 Experimental Settings

We randomly sampled 100,000 documents from the New York Times corpus and evaluated the efficiency of our model based on BRT, $K$NN-BRT, and SpillTree-BRT. We classified the data into two groups. The first was for constructing a constraint tree and the second was for building a new tree based on the constraint tree. The vocabulary size used in the experiment was 665,261.

### 5.4.2 Results

As shown in Fig. 10a, our algorithm outperformed MultiDistance in terms of efficiency. Its performance was also comparable to $K$NN-BRT. Figs. 10b, 10c, and 10d show

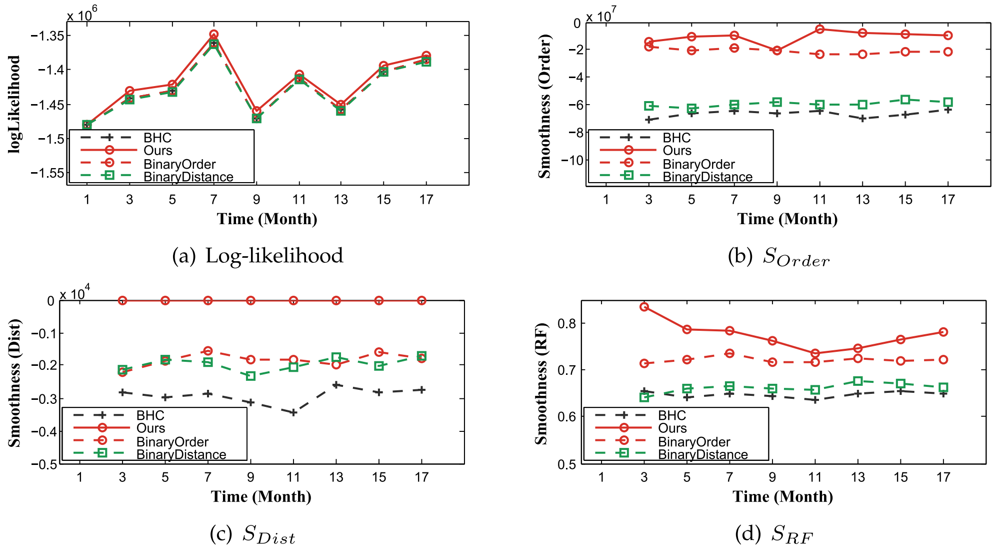


Fig. 9. Comparison of the smoothness and likelihood at different times (our algorithm and binary-based algorithms).

the run time of EvoBRT, $K$NN-EvoBRT, and SpillTree-EvoBRT, respectively. We also compared the implementations with different constraint weights. The results again demonstrates that the performance of our algorithm was comparable to the three BRT-based algorithms. In this example, when the constraint weight was small (i.e., 1e-10), our algorithm was similar to the BRT-based algorithm in terms of efficiency. When the constraint weight was large, our algorithm was even faster for a corpus with a larger number of documents. After checking the results with the faster run time, we found their constraint trees were quite balanced. A balanced constraint tree leads to a balanced tree structure. Typically, BRT and its variations can build a balanced tree much faster. Due to the complexity of our algorithm, it can also handle larger scale datasets in a reasonable amount of time.

## 5.5 Multiple Constraint Trees

In some applications, it may require users to keep track of the evolution of a certain sub-tree that has already existed for some time. For example, in the first half of 2013, the topics "windows," "xbox," and "sales and earnings" co-occurred for several months. Assume that a Microsoft public relations manager is interested in understanding the temporal correlations of these topics in the second half of 2013. Then s/he can gradually learn whether the public relations strategies of the company have succeed on the major products. To this end, it is required to preserve the frequent sub-tree structures by using multiple constraint trees. In this experiment, we aim to evaluate the usage of multiple constraint trees, as well as how the number of constraint trees impacts the smoothness. The experimental settings were the same with that of Section 5.3. We leveraged the last $N_C$ trees. In our experiment, we set $1 \leq N_C \leq 5$.

### 5.5.1 Criteria

We extended the previous smoothness metrics (Section 5.3) to measure the smoothness between $T^t$ and $T^{t-k}$:

$$S^k_{Order} = \frac{1}{\lambda} \log(p_{Order}(T^t|T^{t-k})), \tag{15}$$

$$S^k_{Dist} = \frac{1}{\lambda} \log\left(p_{Dist}(T^t|T^{t-k})\right), \tag{16}$$

$$S^k_{RF} = 1 - \frac{d_{RF}(T^t, T^{t-k}(\mathcal{D}^t)) + d_{RF}(T^{t-k}, T^t(\mathcal{D}^{t-k}))}{2}, \tag{17}$$

where $k$ is the step size between trees. Here the smoothness between non-adjacent trees was also considered for better tracking of the sub-trees of interest over time.

### 5.5.2 Results

Figs. 11a, 11b, and 11c show how smoothness changed with the number of the constraint trees used. It can be seen that

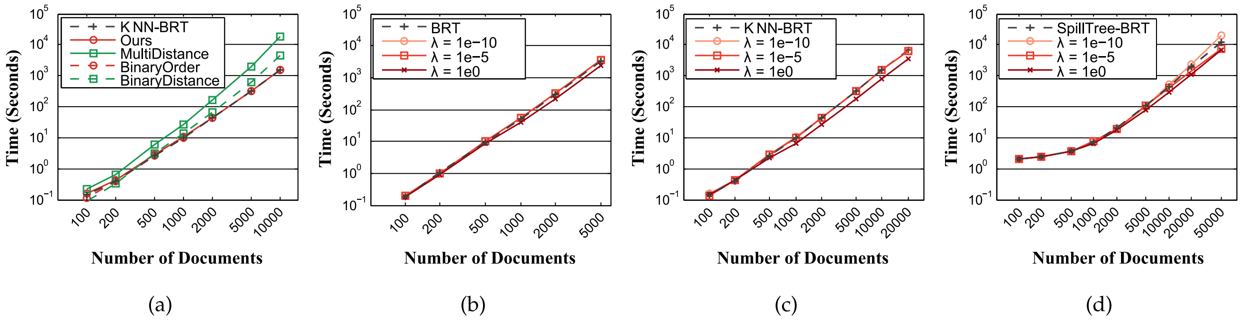


Fig. 10. Efficiency comparison: (a) comparison of different methods ($K$NN); (b) EvoBRT and BRT; (c) $K$NN-EvoBRT and $K$NN-BRT; (d) SpillTree-EvoBRT and SpillTree-BRT.

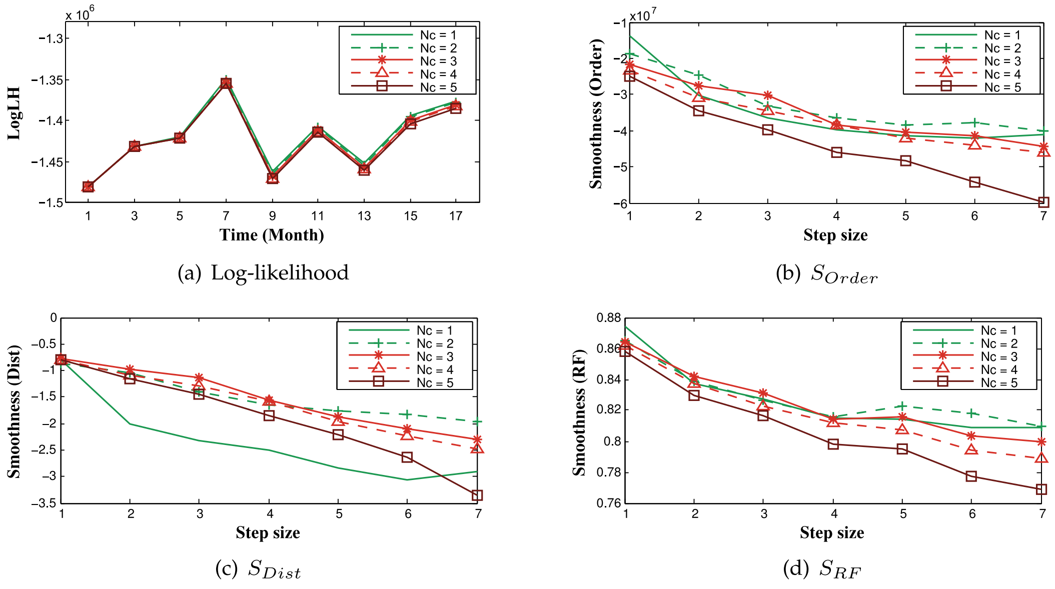


Fig. 11. Comparison of the smoothness and likelihood with different numbers of constraint trees.

more constraint trees usually led to the better smoothness between trees whose step sizes were greater than 1. However, too many constraint trees will reduce the smoothness between trees. For example, when $N_c = 5$, the smoothness decreased under the three metrics. This is because more constraint trees will introduce more conflicting constraints (Fig. 12), which will lead to lower smoothness scores between trees.

Next, we evaluate how the proposed graph matching techniques can help resolve the conflicting constraints among trees and influence both the likelihood and smoothness. In this experiment, we changed the error tolerance (conflict ratio) from 0 to 1 with a step of 0.1. As shown in Fig. 13, the likelihood value decreased as the conflict ratio increased. In the meanwhile, for all the three smoothness metric, the smoothness value also decreased as the conflict ratio increased. This demonstrates that more conflicting constraints lower both the likelihood and smoothness of our clustering algorithm.

## 6 Applications

We conducted two case studies on news data to demonstrate the usefulness of our approach.

### 6.1 Statistics Summary and Comparison

We first compared various statistics between our algorithm and the baseline algorithms on the two datasets used in the case studies. The first one is the Microsoft dataset, which is introduced in Section 1. The second dataset contains 288,423 news articles on the European debt crisis (from Feb. 1 to Jul. 24, 2012). We grouped the data by week and generated 25 trees. The average tree depth was 4, the average internal node number was 276, and the average node number of the first level was 77.

We also briefly illustrated the major advantages of our algorithm. For each algorithm, we did a grid search of $\gamma$ and $\alpha$ and selected the one that yields the largest likelihood. We further did a grid search of the constraint weight and manually select the one that well balances the fitness and smoothness. As shown in Tables 2 and 3, our algorithm outperformed the three baselines in terms of both the likelihood and smoothness. This conclusion is consistent with that derived by the experiments in Section 5.3.

The statistics summary of the tree structure such as tree depth, the node number of the first level, and the internal node number, together with the smooth values indicate that our algorithm generates the most balanced and smooth trees across time. The topic trees generated by BinaryDistance and BinaryOrder were very deep (larger

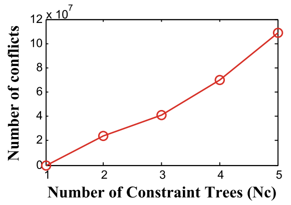


Fig. 12. More constraint trees introduce more conflicts.

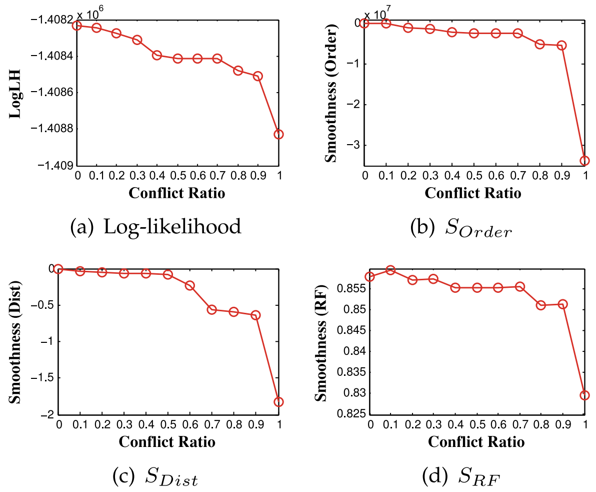


Fig. 13. Evaluation of the graph matching method for resolving conflicts among constraints.

TABLE 2
Statistics Summary of Microsoft Data

| | Likelihood | $S_{Order}$ | $S_{Dist}$ | $S_{RF}$ | Running time (s) | $n_d$ [min, max] | $n_1$ | $n_2$ |
|---|---|---|---|---|---|---|---|---|
| BinaryDistance | −987123.01 | −12052929.74 | −123.42 | 0.91759 | 479.51 | 111 [38, 295] | 2 | 2375 |
| BinaryOrder | −965588.08 | −465636.32 | −116.45 | 0.9109070 | 478.07 | 117 [35, 362] | 2 | 2375 |
| MultiDistance | −952959.24 | −2719459.86 | −1.21 | 0.961792 | 977.19 | 4 [3, 7] | 53 | 103 |
| Ours | −916420.34 | −104211.85 | −0.72 | 0.964418 | 651.71 | 4 [3, 5] | 21 | 99 |

*$n_d$ represents the average tree depth, $n_1$ represents the average node number of the first level, and $n_2$ represents the average internal node number.*

TABLE 3
Statistics Summary of the European Debt Crisis Data

| | Likelihood | $S_{Order}$ | $S_{Dist}$ | $S_{RF}$ | Running time (s) | $n_d$ [min, max] | $n_1$ | $n_2$ |
|---|---|---|---|---|---|---|---|---|
| BinaryDistance | −4603529.80 | −1099465912.72 | −793.13 | 0.941171 | 9409.47 | 123 [73, 235] | 2 | 11536 |
| BinaryOrder | −4572004.32 | −13693773.8 | −709.88 | 0.947099 | 11709.53 | 215 [136, 281] | 2 | 11536 |
| MultiDistance | −4333273.41 | −49185577.18 | −2.32 | 0.972980 | 44480.8 | 4 [3, 15] | 187 | 468 |
| Ours | −4292360.54 | −343917.64 | −0.24 | 0.977457 | 27213.76 | 4 [3, 7] | 77 | 276 |

*$n_d$ represents the average tree depth, $n_1$ represents the average node number of the first level, and $n_2$ represents the average internal node number.*

than 100 levels in both datasets) with only two topics at the first level of the tree ($n_1$) and thousands of internal nodes ($n_2$). Such tree structures are spurious and thus hard for people to understand [22]. Moreover, BinaryDistance and BinaryOrder often generates trees (at different time points) with significantly varying tree depths, which makes it hard for human understanding and tracking. For example, the "windows" hierarchy generated by BinaryDistance had a depth of 141 in the first week and changed to 46 in the second week. MultiDistance generated the flattest trees, with many topics at the first level. For the European debt crisis dataset, the topic trees generated by MultiDistance have averagely 187 nodes at the first level, which introduce some unnecessary fine-grained or even overlapping topics at the first level. For example, the "economy" hierarchy in Fig. 16 contains only one node at the first level, while the one generated by MultiDistance contains 6 nodes, "shares, spain, rate," "index, recession, april," "oil, energy, crude," "oil, crude, barrel," "gold, fed, precious," and "gold, price, bet," at the first level. After examining the related news articles, we found both "oil, energy, crude" and "oil, crude, barrel" were talking about oil trading and should be merged into one topic. So are "gold, fed, precious" and "gold, price, bet." Furthermore, too many fine-grained and overlapping topics make it difficult to understand the overall evolutionary patterns of the "economy" hierarchy over time.

## 6.2 Microsoft Data

The first application aims to illustrate how hierarchical structures can help analyze a document collection from global evolutionary patterns to local, finer-grain topics. We briefly looked at the global patterns in the introduction (Fig. 1). In Fig. 1b, part of "xbox" splits from its main topic in the first week and merges with "windows" in the second week. To understand the underlying causes, we focused on the corresponding sub-trees of the "xbox" topic (T1) and "windows" topic (T2). As shown in Fig. 14, "xbox" has two second-level sub-topics. One is the Ultimate Fighting Championship (UFC) launched on Xbox (A), another is major products of Xbox (B), including "Xbox Live" (B1), "Xbox 720" (B2), and "Xbox 360" (B3). From the alignment edges, we saw that topic B was involved in the interaction with "windows." To examine exactly which products were involved, we checked sub-topics B1, B2, and B3 and found there were two alignment edges between the third-level topics of "xbox" and "windows." One alignment edge was between "Xbox Live" (B1) and "Windows phone" (C1). By browsing the relevant news articles, we found the alignment was caused by the discussion that the "Must Have Games" initiative returned for Windows Phone. Another alignment edge was between "Xbox 360 sales" (B3) and "Windows sales" (C2). This is due to people frequently compared the success of Xbox 360 sales with the slumps on Windows sales when Microsoft reported its second quarter revenue ("Server and Xbox Overcome Windows Weakness in Microsoft's 2Q").

## 6.3 European Debt Crisis Data

The second application is for demonstrating how EvoBRT can help analyze hierarchical evolutionary patterns at different granularities. Fig. 15 shows the evolutionary patterns of the selected first-level topics. It conveys four major topics "fund" (red), "economy" (green), "greece" (yellow),

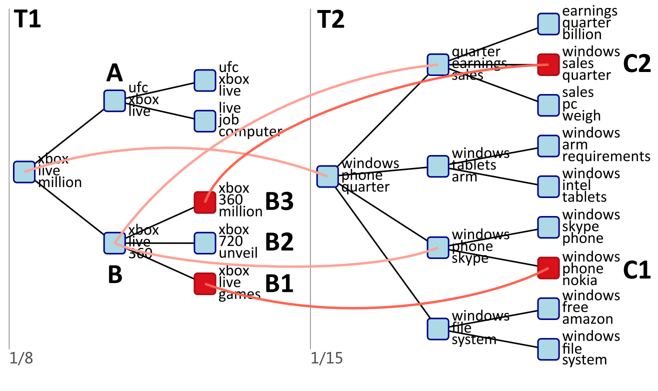


Fig. 14. Detailed information of sub-trees T1 and T2 in Fig. 1. The alignment edges are encoded by red curves.

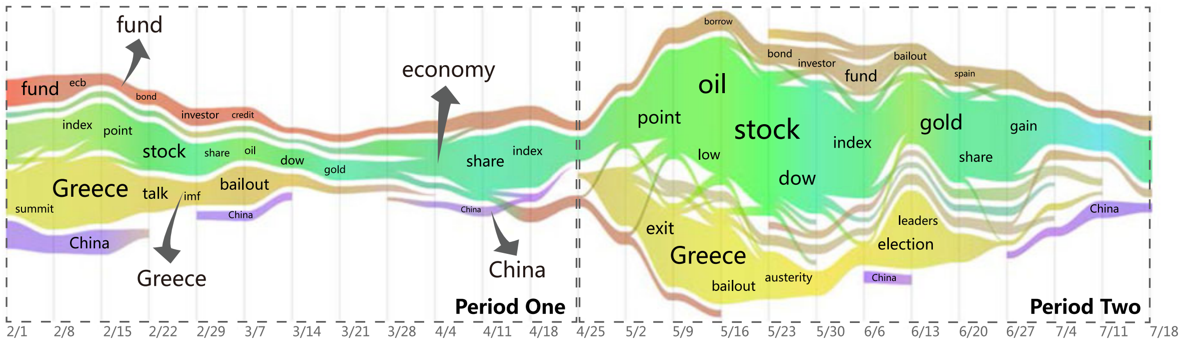


Fig. 15. Visualization of European debt crisis data (Feb. 1 to Jul. 24). The timeline is divided into two periods. In the first period, the topics are relatively independent. In the second period, the topics frequently interact with each other.

and "china" (purple). According to the change degree of topic "greece," we divided the timeline into two periods. In the first period, "greece" had fewer interactions with the others; while in the second period, this topic had more interactions with the others. We were interested in discerning what caused this, so we studied the topic "greece" carefully. The first period of "greece" concerned the bailout for Greece. The bailout was provided by European countries to help Greece to reduce debt. During this period, the topic was relatively independent and gradually shrank because the discussion on the bailout had come to an end in the week of Feb. 15 ("$170B Greek bailout approved"), although it was finally settled in the week of Mar. 14. The second period of "greece" concerned the Greek election. Greece failed to form a government in the first election on May 6. This failure increased the danger of a Greek exit, which will largely aggravate the Eurozone crisis. ("Could the euro survive a Greek exit?"). Because of the severity of the issue, the topics in this period were very active and interacted with each other frequently.

Since Greece leaving the EU would be a disaster, we studied the second period in detail. More exactly, we drilled in to the topic "economy" that had the most interactions with "greece." Fig. 16 shows its topic hierarchies from Apr. 25 to May 15, where "greece" and "economy" had rich interactions with each other. There were three second-level sub-topics under the node "economy" during this period: "gold" (A), "oil" (B), and "stock" (C). We found that all of the sub-topics were influenced by "greece" after the first Greek election on May 6. The content of "gold" (A) changed gradually with more emphasis on the keywords "greece" and "low." For example, one of the news articles had the title of "Gold falls to 4-1/2 month *low* on *Greece* risks." "Oil" (B) displayed a similar evolution to that of "gold." One of the related news items was "Oil price at *lowest* of year: *Greece*, European debt crisis blamed." Moreover, the structure of "oil" grew and generated two third-level sub-topics in the week of May 9. One of them was talking about palm oil and the other was about crude oil (D). "Stock" (C) grew much larger during this period, from two levels to four levels. A new third-level sub-topic "currency" (E) appeared under "stock," describing how the danger of the Greek exit was affecting the currencies of other countries, including New Zealand ("NZ dollar falls as Greek woes heat up"), Australia ("Australian dollar drops 1 percent"), and India ("Rupee hits all time low of 54.46"). A new fourth-level sub-topic "commodities" (F) appeared under "stock" because "post-election turmoil in Europe drove down stocks and commodities; Dow ends down 76." One of the sub-topics, G, under "stock," moved to the fourth level, with more child topics, indicating more discussion on how the Greece exit would impact the stocks in Europe ("European stocks fall

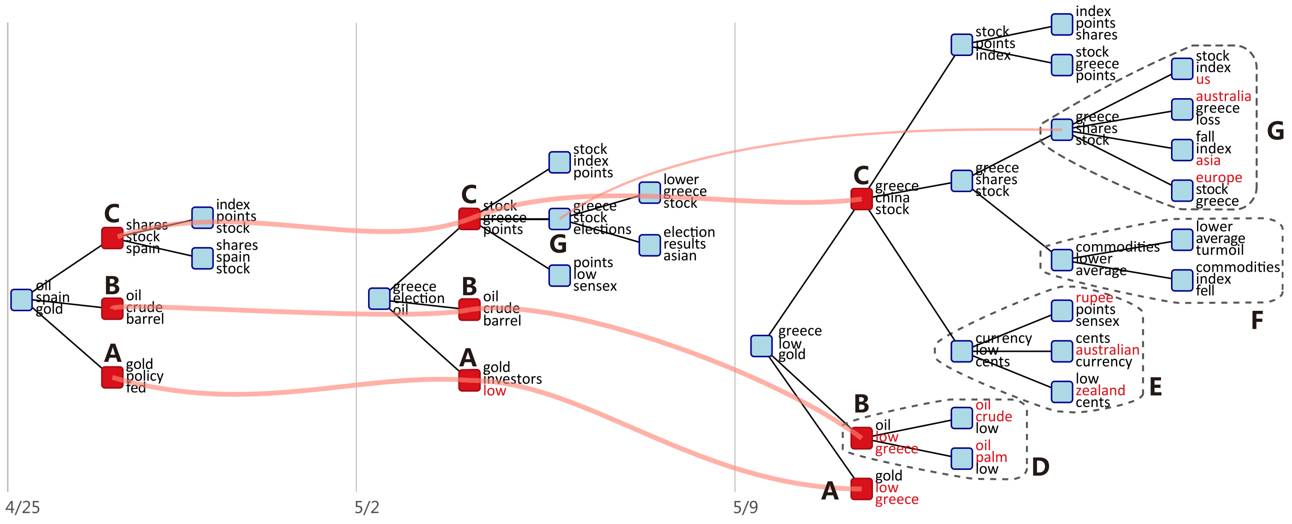


Fig. 16. The "economy" hierarchies. Each sub-tree grows over time due to the intensive discussion of the Greek exit.

sharply amid Greece woes"), Asia ("Asia stocks fall Amid political turmoil in Greece"), Australia ("Australian stocks end lower on Greek concerns"), and *the* U.S ("U.S. stock-index futures decline on Greece's political impasse"). This example shows that our approach can better illustrate the growth of a topic tree over time, including its content and interactions with each other.

## 7 Conclusions

In this paper, we present an evolutionary multi-branch hierarchical clustering algorithm, EvoBRT, to automatically learn dynamic tree structures over time. We leverage a Bayesian online filtering framework to formulate our evolutionary clustering problem. To build multi-branch trees, we adopt the state-of-the-art multi-branch tree clustering method, Bayesian rose trees. To preserve the tree smoothness over time, we use the conditional prior over the tree structures to retain the information from previous trees. Particularly, we introduce the concepts of triples and fans, which can uniquely represent a multi-branch tree. To efficiently compute the tree structure differences, we define a constraint tree from triples and fans, as well as the corresponding operations to make it consistent over time.

Our experiments show that our algorithm outperforms the traditional evolutionary hierarchical clustering algorithm both in tree clustering quality and construction efficiency. The complexity analysis demonstrated that our algorithm can be applied to large-scale datasets. Moreover, two case studies on news data showed that using evolutionary multi-branch tree clustering can help users find many interesting evolutionary patterns of tree structures at different granularities.

## Acknowledgments

The authors would like to thank Hao Wei for helping generate the TextFlow visualization, and Charles Blundell for his insights.

## References

[1] 20 newsgroups dataset [Online]. Available: http://qwone.com/jason/20Newsgroups/, 2014.
[2] 20 newsgroups dataset [Online]. Available: http://nytimes.com, 2014.
[3] Bing news [Online]. Available: http://www.bing.com/news, 2014.
[4] A. Ahmed, Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo, "Unified analysis of streaming news," in *Proc. Int. Conf. World Wide Web*, 2011, pp. 267–276.
[5] A. Ahmed and E. P. Xing, "Dynamic non-parametric mixture models and the recurrent Chinese restaurant process: With applications to evolutionary clustering," in *Proc. SIAM Int. Conf. Data Mining*, 2008, pp. 219–230.
[6] A. Ahmed and E. P. Xing, "Timeline: A dynamic hierarchical Dirichlet process model for recovering birth/death and evolution of topics in text stream," in *Proc. 26th Int. Conf. Uncertainty Artif. Intell.*, 2010, pp. 20–29.
[7] K. Bade and A. Núrnberger, "Creating a cluster hierarchy under constraints of a partially known hierarchy," in *Proc. SIAM Int. Conf. Data Mining*, 2008, pp. 13–24.
[8] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2004, pp. 59–68.
[9] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 113–120.
[10] C. Blundell, Y. W. Teh, and K. Heller, "Discovering non-binary hierarchical structures with Bayesian rose trees," in *Mixtures: Estimation and Applications*. New York, NY, USA: Wiley, 2011, pp. 161–187.
[11] C. Blundell, Y. W. Teh, and K. A. Heller, "Bayesian rose trees," in *Proc. Int. Conf. Uncertainty Artif. Intell.*, 2010, pp. 65–72.
[12] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2006, pp. 554–560.
[13] W. Cui, S. Liu, L. Tan, C. Shi, Y. Song, Z. Gao, H. Qu, and X. Tong, "Textflow: Towards better understanding of evolving topics in text," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 12, pp. 2412–2421, Dec. 2011.
[14] I. Davidson and S. S. Ravi, "Using instance-level constraints in agglomerative hierarchical clustering: Theoretical and empirical results," *Data Min. Knowl. Discov.*, vol. 18, no. 2, pp. 257–282, 2009.
[15] Z. Gao, Y. Song, S. Liu, H. Wang, H. Wei, Y. Chen, and W. Cui, "Tracking and connecting topics via incremental hierarchical Dirichlet processes," in *Proc. 11th Int. Conf. Data Mining*, 2011, pp. 1056–1061.
[16] K. A. Heller and Z. Ghahramani, "Bayesian hierarchical clustering," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 297–304.
[17] K. Heller, "Efficient Bayesian methods for clustering," PhD thesis, Gatsby Unit, UCL, London, U.K., 2008.
[18] C. K. Reddy, H. Park, J. Choo, and C. Lee, "Utopian: User-driven topic modeling based on interactive nonnegative matrix factorization," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 1992–2001, Dec. 2013.
[19] N. Kumar, K. Kummamuru, and D. Paranjpe, "Semi-supervised clustering with metric learning using relative comparisons," in *Proc. Int. Conf. Data Mining*, 2005, pp. 693–696.
[20] E. Y. Liu, Z. Zhang, and W. Wang, "Clustering with relative constraints," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2011, pp. 947–955.
[21] S. Liu, M. X. Zhou, S. Pan, Y. Song, W. Qian, W. Cai, and X. Lian, "TIARA: Interactive, topic-based visual text summarization and analysis," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, pp. 25:1–25:28, 2012.
[22] X. Liu, Y. Song, S. Liu, and H. Wang, "Automatic taxonomy construction from keywords," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2012, pp. 1433–1441.
[23] S. Miyamoto and A. Terami, "Constrained agglomerative hierarchical clustering algorithms with penalties," in *Proc. Int. Conf. Fuzz Syst.*, 2011, pp. 422–427.
[24] M. P. Ng and N. C. Wormald, "Reconstruction of rooted trees from subtrees," *Discrete Appl. Math.*, vol. 69, nos. 1/2, pp. 19–31, 1996.
[25] M. Opper and O. Winther, "A bayesian approach to on-line learning," in *On-Line Learning in Neural Networks*, D. Saad, Ed. New York, NY, USA: Cambridge Univ. Press, 1999, pp. 363–378.
[26] S. Pan, M. X. Zhou, Y. Song, W. Qian, F. Wang, and S. Liu, "Optimizing temporal topic segmentation for intelligent text visualization," in *Proc. Int. Conf. Intell. User Interfaces*, 2013, pp. 339–350.
[27] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vis. Comput.*, vol. 27, no. 7, pp. 950–959, 2009.
[28] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Math. Biosci.*, vol. 53, pp. 131–147, 1981.
[29] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 41–48.
[30] A. Strehl and J. Ghosh, "Cluster ensembles - A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2003.
[31] X. Wang, S. Liu, Y. Song, and B. Guo, "Mining evolutionary multi-branch trees from text streams," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2013, pp. 722–730.
[32] X. Wang, C. Zhai, X. Hu, and R. Sproat, "Mining correlated bursty topic patterns from coordinated text streams," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2007, pp. 784–793.
[33] X. Wang, C. Zhai, and D. Roth, "Understanding evolution of research themes: a probabilistic generative model for citations," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2013, pp. 1115–1123.
[34] X. Wang, K. Zhang, X. Jin, and D. Shen, "Mining common topics from multiple asynchronous text streams," in *Proc. 2nd ACM Int. Conf. Web Search Data Mining*, 2009, pp. 192–201.

[35] F. Wei, S. Liu, Y. Song, S. Pan, M. X. Zhou, W. Qian, L. Shi, L. Tan, and Q. Zhang, "TIARA: A visual exploratory text analytic system," in *Proc. ACM SIGKDD Knowl. Discovery Data Mining*, 2010, pp. 153–162.

[36] T. Xu, Z. Zhang, P. Yu, and B. Long, "Dirichlet process based evolutionary clustering," in *Proc. 8th Int. Conf. Data Mining*, 2008, pp. 648–657.

[37] T. Xu, Z. Zhang, P. Yu, and B. Long, "Evolutionary clustering by hierarchical Dirichlet process with hidden Markov state," in *Proc. 8th Int. Conf. Data Mining*, 2008, pp. 658–667.

[38] J. Zhang, Y. Song, C. Zhang, and S. Liu, "Evolutionary hierarchical Dirichlet processes for multiple correlated time-varying corpora," in *Proc. 13th ACM SIGKDD Knowl. Discovery Data Mining*, 2010, pp. 1079–1088.

[39] H. Zhao and Z. Qi, "Hierarchical agglomerative clustering with ordering constraints," in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 195–199.

[40] L. Zheng and T. Li, "Semi-supervised hierarchical clustering," in *Proc. 8th Int. Conf. Data Mining*, 2011, pp. 982–991.

**Shixia Liu** received the BS and MS degrees in computational mathematics from the Harbin Institute of Technology, the PhD degree in computer science from Tsinghua University. She is an associate professor at Tsinghua University. Her research interests include visual text analytics, visual social analytics, and graph visualization. Before joining Tsinghua University, she worked as a lead researcher at Microsoft Research Asia and a research staff member at IBM China Research Lab.

**Xiting Wang** received the BS degree in electronics engineering from Tsinghua University. She is currently working toward the PhD degree in the Institute for Advanced Study at Tsinghua University, China. Her research interests include visual text analytics and text mining.

**Yangqiu Song** received the BE and PhD degree from Tsinghua University, China. He is a postdoctoral researcher at the University of Illinois at Urbana-Champaign. His current research focuses on using machine learning and data mining to extract and infer insightful knowledge from big data, including the techniques of large scale learning algorithms, natural language understanding, text mining and visual analytics, and knowledge engineering.

**Baining Guo** received the BS degree from Beijing University, the MS and PhD degrees from Cornell University. He is the assistant managing director of Microsoft Research Asia, where he also serves as the head of the graphics lab. Prior to joining Microsoft in 1999, he was a senior staff researcher with the Microcomputer Research Labs of Intel Corporation in Santa Clara, California. His research interests include computer graphics and visualization, in the areas of texture and reflectance modeling, texture mapping, translucent surface appearance, real-time rendering, and geometry modeling.