

ContexTour: Contextual Contour Visual Analysis on Dynamic Multi-Relational Clustering

Yu-Ru Lin¹ Jimeng Sun² Nan Cao^{2,3} Shixia Liu²

¹School of Arts Media and Engineering,
Arizona State University

²IBM Research

³Hong Kong University of Science and Technology

Abstract

Huge amounts of rich context social network data are generated everyday from various applications such as FaceBook and Twitter. These data involve multiple social relations which are community-driven and dynamic in nature. The complex interplay of these characteristics poses tremendous challenges on the users who try to understand the underlying patterns in the social media. We introduce an exploratory analytical framework, *ContexTour*, which generates visual representations for exploring multiple dimensions of community activities, including relevant topics, representative users and the community-generated content, as well as their evolutions. ContexTour consists of two novel and complementary components: (1) Dynamic Relational Clustering (DRC) that efficiently tracks the community evolution and smoothly adapts to the community changes, and (2) Dynamic Network Contour-map (DNC) that visualizes the community activities and evolutions in various dimensions. In our experiments, we demonstrate ContexTour through case studies on the DBLP dataset. The visual results capture interesting and reasonable evolution in Computer Science research communities. Quantitatively, we show 85-165X performance gain of our DRC algorithm over the baseline method.

1. Introduction

Huge amounts of rich context social media data are generated everyday from various applications such as FaceBook, Twitter, Digg, Flickr, DBLP, etc. The main characteristics of social media include:

- *Multiple relations*: Social media data typically involve multiple social relations, e.g., collaboration networks may include an author-to-author relation, an author-to-keyword relation, and an author-to-conference relation.
- *Community-driven*: The activity patterns in social media often reflect the community structure of users, e.g., in collaboration networks, communities emerge due to authors' research interests and co-authorship.
- *Dynamic in nature*: The community structures are constantly evolving over time, e.g., research topics may change over years.

The complex interplay of these characteristics poses tremendous challenges on the users who try to understand the underlying patterns in the social media. Various approaches have been proposed to study the social community structure, including models to explain the overall social phenomena [15] and algorithms to detect specific patterns such as time-evolving communities [17] as well as clusters across multiple relations [2,18,23,24]. However, little work has considered the capability for users to explore multiple aspects of community activities, including relevant topics, representative users and the community-generated content, as well as their evolutions. Existing visualization tools such as tag clouds based on aggregate statistics may help identify important topics (or keywords), but are limited in helping users understand the context of the topics, such as their relationship to social network dynamics. Similarly, traditional network visualization (such as the node-link representation, e.g., [10]) is usually restricted in the way of displaying richer aspects (e.g., concept drift) on the network. This gap is aggravated in time-evolving data.

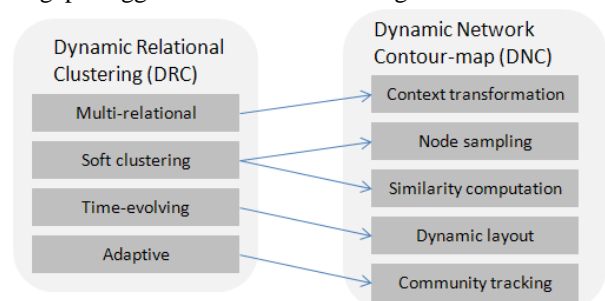


Figure 1: *ContexTour* consists of analysis component DRC and visualization component DNC, which supports large-scale, multi-aspect visual exploration.

There are several challenges in visually analyzing large scale social networked data: (1) *scalability* – how to extract structures from large scale social media data; (2) *legibility* – how to show many important individuals (users or keywords) while also conveying their sum (communities or topics); (3) *rich context* – how to display multiple aspects of information through consistent visual representations.

To tackle these challenges, we introduce an exploratory analytical framework, *ContexTour*, which generates

conuserable patterns for visually analyzing large and dynamic social media data. As shown in Figure 1, ContextTour consists of two novel and complementary components: (1) Dynamic Relational Clustering (DRC) for analyzing the data and (2) Dynamic Network Contour-map (DNC) for visually conveying the analysis results in an intuitive fashion. There are several key ideas in ContextTour:

- *Multi-relational*: DRC clusters multi-relations simultaneously such that all clusters are consistent across all relations. Multi-relational clustering in DRC enables the context switching of DNC, which provides a smooth transition between different dimensions.
- *Soft-clustering*: DRC generates soft clustering results which provide an effective summarization mechanism for selecting important entities and computing entity similarity in DNC.
- *Time-evolving*: DRC incrementally and smoothly tracks community changes over time as new data arrive, which allows DNC to generate smooth dynamic layout.
- *Adaptive*: DRC automatically adjusts the number of communities over time and generates the community transition probability, which enables community tracking in the DNC visualization.
- *Scalable*: DRC is a computationally efficient solution where the required CPU time and memory is linear to the number of non-zeros elements in all relations, which is capable of processing large scale networked data.

We have conducted extensive experiments. Qualitatively, we demonstrate ContextTour through case studies on DBLP data (see Figure 8). The visual results reveal interesting and reasonable evolution patterns in Computer Science research communities. Quantitatively, we show 85-165X performance gain of DRC against the baseline method.

The rest of paper is organized as the follows: we first introduce the related work in section 2. We then present the data mining and visualization components of ContextTour in sections 3 and 4. Section 5 reports the experiments, including a case study and performance evaluation. Finally, we conclude in section 6.

2. Related Work

We discuss the related work in three research directions: dynamic network mining, relational learning and network visualization.

Dynamic network mining. Research based on the statistical properties of online social networks provides important insight regarding the structure and evolution of social behavior [1,15]. The structure of social interactions among people have been modeled as uni-partite or bipartite graphs, in which the community structure can be characterized by clustering methods [17] or latent space model [21]. Sarkar and Moore [21] propose a method that embeds entities into latent spaces where the entity coordinates at consecutive

timesteps are regularized to avoid dramatic changes. Lin et al. [17] use an evolutionary clustering criterion [6] to extract smoothly evolving community structures based on both observed networked data and historic community structure. All these works restrict themselves to a single time-varying pair-wise relation between entities (e.g. user-user or user-paper).

Relational learning. As discussed in [16], classical propositional clustering methods are limited in dealing with data having various types of entities and different semantic relationships among them. The first-order extensions of classical methods (e.g. RDBC [11]) may not be feasible for large relational data due to the quadratic computational complexity. Recent relational clustering techniques have been shown effective in learning the interrelated structures among various entities and multiple relationships [2,18,23,24]. Long et al. [18] propose a general model to cluster multi-type interrelated data objects simultaneously. Their basic idea is to construct new relations with hidden nodes (clusters), which summarize the link information in the original relations, to approximate the original relations. However, few of these methods consider the evolutionary characteristic in relational data. An exception is the clustering method proposed in [23] for mining dynamic multi-mode networks. We will discuss the technical differences between [23] and our work in a later section.

Our goal is to design network visual mining solution on for multi-relational data. The closest one is recent work by Sun et al. [22], which combine tensor technique with network visualization. However, we focus on dynamic and multi-relation data, while [22] aims at static tensor data (single relation involving multiple dimensions).

Network visualization. There have been various techniques for visualizing complex interrelated data. Classical approaches include multidimensional scaling (MDS) [3] and self-organized map (SOM) [12]. Compared to these approaches, node-link network visualization tools (e.g. [10]) are widely applied to reveal both global and local structures, and have been extended to visualize dynamic networks. Research on dynamic network visualization can be characterized as *offline* or *online* approaches. Offline network visualization assumes the complete time-varying networks are given in advance. Chen and Carr [7] propose animated visualization for visualizing the evolution of paper co-citation networks, where the node-link representation is enhanced by link reduction. Other approaches, such as hierarchical force-directed layout [13] and foresighted layout [8] have been proposed to generate animation without drastic movements of nodes in the animation. Online dynamic visualization methods, such as spectral layout [5] and Bayesian dynamic layout [4], use network data available at run time to incrementally update the graph layout.

These network visualization tools, however, are not suitable for visualizing rich context social network data. The inherent

community structures in such data often introduce clutter and occlusion problems in graph layout that prevent users from seeing underlying patterns involving multi-type entities and relationships. We incorporate two novel ideas in our framework to overcome this challenge. First, we leverage an efficient multi-relational clustering algorithm to incrementally generate graph layouts which are smooth across time and different relations. Second, motivated by spatial text visualization techniques [9,26], we introduce a contour metaphor to create a “mental map” about the density of nodes and edges, and to reduce clutter and occlusion as in the conventional node-link representation.

3. Dynamic Relational Clustering

We now present our method for extracting soft clusters from time-varying multi-relational social data. We first define the notations in section 3.1, and then formulate the problem as an optimization problem in section 3.2. We provide an efficient solution to the clustering objective in section 3.3. Finally, we propose an automatic mechanism for determining the number of clusters in section 3.4.

3.1 Notations. The observation of a relation can be represented by a data matrix. Let r denote a relation involves two sets of entities S_x and S_y . Let N_x (N_y) denote the cardinality of S_x (S_y). The data matrix for r at a given time t is denoted as $\mathbf{W}_t^{(r)} \in \mathfrak{R}_+^{N_x \times N_y}$, where element $\mathbf{W}_{t,ij}^{(r)}$ or $w_{t,ij}^{(r)}$ represents the amount of interactions at time t between two entities $i \in S_x$ and $j \in S_y$. \mathfrak{R}_+ indicates all elements of the matrix have nonnegative values, which is common for a data matrix. To avoid notation clutter, we omit the superscript r and subscript t unless they are needed for clarity.

We approximate the observed interaction w_{ij} by using a mixture model $w_{ij} \approx \sum_k p_k p^{(x)}_{ik} p^{(y)}_{jk}$, where p_k is the prior probability of an interaction in the k -th cluster, $p^{(x)}_{ik}$ and $p^{(y)}_{jk}$ indicate how likely an interaction in the k -th cluster involves the i -th entity in entity set S_x and the j -th entity in entity set S_y , respectively. Written in a matrix form, we have $\mathbf{W} \approx \mathbf{U}^{(x)} \mathbf{\Lambda} (\mathbf{U}^{(y)})^T$, where $(\cdot)^T$ is the matrix transpose operation, $\mathbf{U}^{(q)} \in \mathfrak{R}_+^{N_q \times K}$ is a nonnegative matrix indicating the cluster membership for the total K clusters. In particular, an element $\mathbf{U}_{ik}^{(q)} = p^{(q)}_{ik}$ for $i \in S_q$ and $\sum_i \mathbf{U}_{ik}^{(q)} = 1$; $\mathbf{\Lambda}$ is a $K \times K$ nonnegative diagonal matrix with $\mathbf{\Lambda}_k = p_k$ and $\sum_k \mathbf{\Lambda}_k = 1$, where $\mathbf{\Lambda}_k$ is short for $\mathbf{\Lambda}_{kk}$. We introduce a special matrix product notion that will be convenient to deal with multiple relations. We write $\mathbf{U}^{(x)} \mathbf{\Lambda} (\mathbf{U}^{(y)})^T$ as $\mathbf{\Lambda} \times \{\mathbf{U}^{(q)}\}_{q \in \Gamma(r)}$, where $\Gamma(r) = \{x, y\}$ represents the two sets of entities (S_x and S_y) involved in relation r . Note that x and y correspond to the first and second mode of the data matrix. The clustering structure can be fully specified by $\mathbf{\Lambda}$ and $\{\mathbf{U}^{(q)}\}$, where $\mathbf{\Lambda}$ is referred as the *core* matrix and $\{\mathbf{U}^{(q)}\}$ as the *factor* matrices. We summarize our notations in Table 1.

3.2 Problem Formulation. We formulate the dynamic relational clustering as an optimization problem where the objective is to find smoothly evolving soft clusters that best

Symbol	Description
r	relation index
q	entity index
S_q, N_q	a set of entities, and its cardinality
\mathbf{W}	a matrix (boldface capital letter)
$\mathbf{W}_t^{(r)}$	a matrix for relation r of a given time t (the superscript r and subscript t will be omitted unless they are needed for clarity)
K, L, M	constants (K denotes the number of clusters; L denotes the number of total relations; M denotes the number of different entity types)
p_k	the prior probability in the k -th cluster
$p^{(q)}_{ik}$	the likelihood in the k -th cluster involves the i -th entity in entity set S_q
$\mathbf{\Lambda}$	core matrix; a diagonal matrix with $\mathbf{\Lambda}_{kk} = p_k$
$\mathbf{U}^{(q)}$	factor matrix corresponding to entity type S_q ; $\mathbf{U}_{ik}^{(q)} = p^{(q)}_{ik}$
$\Gamma(r)$	the sets of entity types involved in relation r , e.g. $\Gamma(r) = \{x, y\}$ indicates sets S_x and S_y are involved in the relation r

Table 1: Description of notations.

explain our observation of multiple relations of data entities over time. We generalize the evolutionary clustering framework used in [17] to multi-relational setting and introduce a novel temporal cost function that determine the number of clusters in a memory efficient manner.

Dynamic relational clustering. We consider measuring the quality of the time-evolving clustering structure in terms of snapshot cost and temporal cost. The total cost is given by:

$$\text{cost} = (1 - \alpha) \cdot \mathcal{CS} + \alpha \cdot \mathcal{CT}, \quad <1>$$

where \mathcal{CS} is the snapshot cost that measures how well the clustering structure fits the observed data of a given time, and \mathcal{CT} is the temporal cost that measures how consistent the clustering structure is with respect to the historic clustering structure. $\alpha \in [0, 1]$ is the weighting factor to specify how much temporal consistency needs to be remained. Based on the mixture model, we define the snapshot cost $\mathcal{CS}^{(r)}$ for relation r as the deviation introduced by the approximation:

$$\mathcal{CS}^{(r)} = D(\mathbf{W}^{(r)} \parallel \mathbf{\Lambda} \times \{\mathbf{U}^{(q)}\}_{q \in \Gamma(r)}), \quad <2>$$

where $D(\mathbf{A} \parallel \mathbf{B}) = \sum_{ij} (\mathbf{A}_{ij} \log \mathbf{A}_{ij} / \mathbf{B}_{ij} - \mathbf{A}_{ij} + \mathbf{B}_{ij})$ is the Kullback-Leibler (KL) divergence between matrices \mathbf{A} and \mathbf{B} , and where $\sum_{ij} \mathbf{A}_{ij} = \sum_{ij} \mathbf{B}_{ij} = 1$. We use KL divergence to measure the approximation error as it is a natural measure of the dissimilarity of two distributions. Without loss of generality, we normalize $\mathbf{W}^{(r)}$ such that $\sum_{ij} \mathbf{W}_{ij}^{(r)} = 1$. Next we will formulate the temporal cost function \mathcal{CT} , which is the key contribution in this paper.

Temporal regularization. The temporal cost is used to regularize the clustering structure at a given time t so that

drastic changes in the structure from time $t-1$ to t are less likely. Typically, there exist two different ways to define the temporal cost in the literature:

(a) The temporal cost function may be defined by comparing the clustering membership for a set of entities across time, e.g., the deviation from $\mathbf{U}_{t-1}^{(q)}$ to $\mathbf{U}_t^{(q)}$. The challenge here is that $\mathbf{U}_{t-1}^{(q)}$ and $\mathbf{U}_t^{(q)}$ cannot be directly compared if there are different numbers of clusters at t and $t-1$.

(b) Another way to handle this is to define the temporal cost function as the difference between the approximated matrix at time $t-1$ and the data matrix at time t . The approximated matrix depends on the clustering structure at time t , but the cost function does not require the numbers of clusters at t and $t-1$ to be the same. This approach has been used in prior work [17,23]. However, it requires the (often large and dense) approximated matrix to be explicitly reconstructed, which is expensive in both computational time and memory space. Hence, the approach is not applicable to large scale social networks. Moreover, the clusters at time $t-1$ are not explicitly mapped to those clusters at time t and therefore some partition-matching algorithms (e.g., [19]) must be used to obtain the optimal cluster mapping between those at time $t-1$ and those at time t , where the partition-matching is an NP-hard problem.

To overcome these challenges, we propose a novel temporal cost function that allows change of cluster numbers. The idea is to learn virtual relations that connect old clusters at $t-1$ to new clusters at t .

Figure 2 illustrates the proposed method. Assume we have two set of entities (e.g. authors and keywords) S_x and S_y , and we have observed relational matrix \mathbf{W}_{t-1} that indicates relationship between entities in sets S_x and S_y (e.g. the number of times an author use a keyword). Assume we obtain clustering for time $t-1$, which are fully specified by Λ_{t-1} , $\mathbf{U}_{t-1}^{(x)}$, and $\mathbf{U}_{t-1}^{(y)}$. Based on the clustering we can establish virtual relations between entities and clusters. Let S_c denotes the set of clusters at time $t-1$. The virtual relations between the entity set S_x and the set of clusters S_c , at time $t-1$, are established by parameters Λ_{t-1} , $\mathbf{U}_{t-1}^{(x)}$, and $\mathbf{U}_{t-1}^{(y)}$ as follows: $\mathbf{W}^{(xc)} = \mathbf{U}_{t-1}^{(x)} \Lambda_{t-1}$ and $\mathbf{W}^{(yc)} = \mathbf{U}_{t-1}^{(y)} \Lambda_{t-1}$. (A probabilistic interpretation for this construction will be provided later.). Now, in order to obtain clustering at time t that smoothly evolves from $t-1$, we require the clustering structure to explain both observed relational matrix \mathbf{W}_t and virtual relational matrices, $\mathbf{W}^{(xc)}$ and $\mathbf{W}^{(yc)}$. The approximation of a virtual relational matrix is calculated similarly as the approximation of an observed relational matrix. For example, $\mathbf{W}^{(xc)}$ can be approximated by the core matrix, Λ_t , and two factor matrices along each mode. In this case, the first mode corresponds to the entity set S_x , and the second mode corresponds to the set of clusters S_c at time $t-1$. Hence, the first factor matrix $\mathbf{U}_t^{(x)}$ indicates the relationship between entities in S_x and the extracted clusters at time t , and another factor matrix, denoted by $\mathbf{U}_t^{(c)}$, indicates the

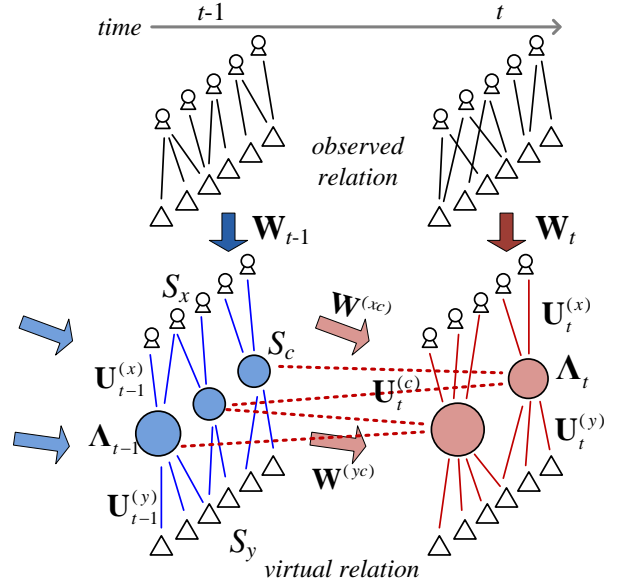


Figure 2: Schematic view of the proposed dynamic relational clustering. The arrows indicate input relations for each time, e.g., for time t , the dark red arrow indicates an observed relational matrix \mathbf{W} involving two entity sets S_x and S_y (e.g. authors and keywords), and the light red arrows indicate virtual relational matrices $\mathbf{W}^{(xc)}$ between S_x and the set of clusters, denoted as S_c , and $\mathbf{W}^{(yc)}$ between S_y and S_c . The cost function is defined based on the loss in approximating the observed relations (snapshot cost) and virtual relations (temporal cost).

relationship between the old clusters (at time $t-1$, i.e. S_c) and new clusters (at time t).

The virtual relations $\mathbf{W}^{(xc)}$ and $\mathbf{W}^{(yc)}$ summarize the hidden structures at time $t-1$ which will be treated as prior knowledge for clustering at time t . The idea is different from [18], where new hidden relations is constructed in order to approximate the original relations.

A virtual relation between a set of entities S_q and the set of historic clusters S_c naturally provides historic information about these entities. Hence, we define the temporal cost $\mathcal{CT}^{(q)}$ for a set of entities S_q as the deviation introduced by the approximation of the corresponding virtual relation:

$$\mathcal{CT}^{(q)} = D(\mathbf{U}_{t-1}^{(q)} \Lambda_{t-1} \| \Lambda_t \times \{\mathbf{U}_t^{(q)}, \mathbf{U}_t^{(c)}\}) \quad <3>$$

where $D(\cdot \| \cdot)$ is the KL divergence defined above.

Putting together, we define the dynamic relational clustering objective as minimizing the following cost function:

$$\begin{aligned} J(\Lambda, \{\mathbf{U}^{(q)}\}) &= (1-\alpha) \sum_r \mathcal{CS}^{(r)} + \alpha \sum_q \mathcal{CT}^{(q)}, \\ \text{s.t. } \mathbf{U}^{(q)} &\in \mathfrak{R}_+^{N_q \times K}, \Lambda \in \mathfrak{R}_+^{K \times K}, \\ \sum_i \mathbf{U}_{ik}^{(q)} &= 1 \quad \forall k, \sum_k \Lambda_k = 1, \end{aligned} \quad <4>$$

where $\mathcal{CS}^{(r)}$ is the snapshot cost for relation r as defined in eq. <2> and $\mathcal{CT}^{(q)}$ is the temporal cost for the set of entities U_q as defined in eq. <3>, and α is a real number between 0

and 1 to specify how much temporal consistency needs to be maintained. This equation can be easily extended to incorporate weights on each relation or entity set. Before we describe how to solve the clustering objective, we now provide a probabilistic interpretation of the solution.

Probabilistic interpretation. Let $p_{i,k}$ denote a virtual relation between the i -th entity and the k -th cluster. By considering the virtual relation as a joint probability of the entity i and cluster k , we have $p_{i,k}=p_k p_{ik}$ (the superscript of the entity set, i.e. S_x is omitted without ambiguity). Ideally, when the data have not changed significantly, the virtual relations at time $t-1$ can be approximated by the clustering structure at time t . The approximation of a virtual relation matrix is again achieved by using a mixture model. Let k' index a cluster at time $t-1$ and k index a cluster at time t , and let $p_{k'|k}$ indicate how likely an interaction in the k -th cluster is previously involved in the k' -th cluster at time $t-1$. We have $p_{i,k} \approx \sum_{k'} p_k p_{ik'} p_{k'|k}$, where $p_{i,k'}$ is the virtual relation between i and k' (at time $t-1$), p_k and p_{ik} are defined as above with respect to cluster k (at time t). The conditional likelihood $p_{k'|k}$ can be considered as the transition probability between cluster k' in $t-1$ and cluster k in t , which indicates the transition of clustering structures across time. In the matrix form, we have: $\mathbf{U}_{t-1}^{(q)} \mathbf{\Lambda}_{t-1} \approx \mathbf{\Lambda}_t \times \{\mathbf{U}_t^{(q)}, \mathbf{U}_t^{(c)}\}$, where $\mathbf{U}_{t-1;ik'}^{(q)} = p_{ik'}$, $\mathbf{\Lambda}_{t-1;k'} = p_{k'}$, $\mathbf{\Lambda}_{t;k} = p_k$, $\mathbf{U}_{t;ik}^{(q)} = p_{ik}$ and $\mathbf{U}_{t;k'k}^{(c)} = p_{k'|k}$. We refer the matrix $\mathbf{U}_t^{(c)}$ as the transition matrix, with elements indicating transition probability of clusters from time $t-1$ to t . One unique characteristics of our formulation is that the cross-time transition between clustering structures, even with varying number of clusters, is *directly* obtained.

The solution matrices $\mathbf{\Lambda}$ and $\{\mathbf{U}^{(q)}\}$ uniquely define the evolving clustering structure for a given time t . $\mathbf{\Lambda}_k = p_k$ indicates the prior probability of the k -th cluster, which can be interpreted as popularity of the cluster. $\mathbf{U}_{ik}^{(q)} = p_{ik}$ indicates the likelihood of an interaction in cluster k involving the entity i , which can be interpreted as the contribution (or importance) of i in k . We determine the soft membership of entity i with respect to a cluster k as the conditional probability of a cluster given the entity, denoted as $p_{ki} = p_k p_{ik} / p_i$, where $p_i = \sum_z p_z p_{iz}$ is the marginal probability of an interaction involving entity i .

3.3 Solution. We provide a solution to the clustering objective defined in eq. <4>. We first rewrite eq. <3> as follows, so that the temporal cost function has the same form as the snapshot cost function:

$$\mathcal{C}T^{(q)} = D(\mathbf{W}^{(q)} \parallel \mathbf{\Lambda}_t \times \{\mathbf{U}_t^{(q)}\}_{q \in \Gamma(q)}), \quad <5>$$

where $\mathbf{W}^{(q)} = \mathbf{U}_{t-1}^{(q)} \mathbf{\Lambda}_{t-1}$ is a virtual relation corresponding to an entity set S_q , which represents the virtual relations between the set of entities S_q and the set of historic clusters S_c . We have written $\mathbf{\Lambda}_t \times \{\mathbf{U}_t^{(q)}, \mathbf{U}_t^{(c)}\}$ as $\mathbf{\Lambda}_t \times \{\mathbf{U}_t^{(q')}\}_{q' \in \Gamma(q)}$, where $\Gamma(q) = \{q, c\}$ denotes the two sets of entities (S_q and S_c) involved in the virtual relation q . Then, eq. <4> can be written as:

$$J(\mathbf{\Lambda}, \{\mathbf{U}^{(q)}\}) = \sum_{r'} \omega^{(r')} D(\mathbf{W}^{(r')} \parallel \mathbf{\Lambda} \times \{\mathbf{U}^{(q)}\}_{q \in \Gamma(r')}) \quad <6>$$

with the same constraints defined in eq. <4>, where $r' \in \{r\} \cup \{q\}$ indexes both the observed relations and virtual relations; $(1-\alpha)/L$ if $r' \in \{r\}$ and $\omega^{(r')} = \alpha/M$ if $r' \in \{q\}$ where L and M are the number of relations and the number of entity sets, respectively; $D(\cdot \parallel \cdot)$ is the KL divergence defined above. By employing the concavity of the log function given in the KL-divergence, we derive a local minima solution to eq. <6> as follows.

Theorem 1. The cost defined in eq. <6> is non-increasing under the following update rules and therefore converges to a local optimal:

$$\begin{aligned} \mathbf{U}_{ik}^{(q)} &\leftarrow \sum_{\{r': q \in \Gamma(r')\}} \omega^{(r')} \sum_j v_{q:ij}^{(r')} \mu_{ijk}^{(r')}, \\ &\text{then normalize such that } \sum_i \mathbf{U}_{ik}^{(q)} = 1 \quad \forall k, \\ \mathbf{\Lambda}_k &\leftarrow \sum_{r'} \omega^{(r')} \sum_{ij} \mathbf{W}_{ij}^{(r')} \mu_{ijk}^{(r')}, \\ &\text{then normalize such that } \sum_k \mathbf{\Lambda}_k = 1, \quad <7> \\ \text{where } \mu_{ijk}^{(r')} &= \frac{\mathbf{\Lambda}_k \times \{\mathbf{U}_{ik}^{(q)}\}_{q \in \Gamma(r')}}{(\mathbf{\Lambda} \times \{\mathbf{U}^{(q)}\})_{ij}}, \\ v_{q:ij}^{(r')} &= \begin{cases} \mathbf{W}_{ij}^{(r')} & \text{if } q \text{ corresponds to the first mode of } \mathbf{W}^{(r')} \\ \mathbf{W}_{ji}^{(r')} & \text{if } q \text{ corresponds to the second mode of } \mathbf{W}^{(r')} \end{cases} \end{aligned}$$

The proof of Theorem 1 is provided in the appendix. This iterative update algorithm is a multi-matrices factorization that generalizes the algorithm proposed by Lee et al. [14], where their algorithm deals with a single non-negative matrix factorization problem. Table 2 summarizes the process for solving $\mathbf{\Lambda}$ and $\{\mathbf{U}^{(q)}\}$.

Algorithm: Dynamic Relational Clustering (DRC)

Input: data matrices $\{\mathbf{W}^{(r')}\}$ for time t
prior model $\mathbf{\Lambda}_{t-1}$ and $\{\mathbf{U}_{t-1}^{(q)}\}$
Output: $\mathbf{\Lambda}$ and $\{\mathbf{U}^{(q)}\}$ for time t
Method:
For each q , construct virtual matrices
 $\mathbf{W}^{(q)} = \mathbf{U}_{t-1}^{(q)} \mathbf{\Lambda}_{t-1}$
Let $\{\mathbf{W}^{(r')}\} = \{\mathbf{W}^{(r')}\} \cup \{\mathbf{W}^{(q)}\}$
Initialize $\mathbf{\Lambda}$, $\{\mathbf{U}^{(q)}\}$
Repeat until *converge*
For each q , update $\mathbf{U}^{(q)}$ by eq. <7>
update $\mathbf{\Lambda}$ by eq. <7>

Table 2: Algorithm for Dynamic Relational Clustering

A toy example. Let us use the scenario in Figure 2 to illustrate how our algorithm works for clustering evolving relational data. We consider an author-by-keyword relation in this scenario. Assume we observe relational matrices \mathbf{W} at time $t-1$ and t , as follows:

$$\mathbf{W}_{t-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{W}_t = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

We will identify three clusters for $t-1$ and two clusters for t with equal weight between \mathcal{CS} and $\mathcal{CT}(\alpha=0.5)$. Without prior information, we obtain clustering solution at time $t-1$:

$$\Lambda_{t-1} = \begin{bmatrix} 0.33 & 0 & 0 \\ 0 & 0.33 & 0 \\ 0 & 0 & 0.33 \end{bmatrix} \mathbf{U}_{t-1}^{(x)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0.33 & 0 \\ 0 & 0.67 & 0 \\ 0.67 & 0 & 0 \\ 0.33 & 0 & 0 \end{bmatrix} \mathbf{U}_{t-1}^{(y)} = \begin{bmatrix} 0 & 0 & 0.33 \\ 0 & 0 & 0.33 \\ 0 & 0.67 & 0.33 \\ 0 & 0.33 & 0 \\ 0.33 & 0 & 0 \\ 0.67 & 0 & 0 \end{bmatrix}$$

Without considering historic information, we obtain clustering solution at time t :

$$\Lambda_t = \begin{bmatrix} 0.56 & 0 \\ 0 & 0.44 \end{bmatrix} \mathbf{U}_t^{(x)} = \begin{bmatrix} 0.40 & 0 \\ 0.40 & 0 \\ 0.20 & 0.25 \\ 0 & 0.25 \\ 0 & 0.50 \end{bmatrix} \mathbf{U}_t^{(y)} = \begin{bmatrix} 0.40 & 0 \\ 0 & 0.25 \\ 0.20 & 0 \\ 0.40 & 0 \\ 0 & 0.50 \\ 0 & 0.25 \end{bmatrix}$$

The clustering results make sense for each time snapshot, but it is not useful in capturing evolving communities from the time-varying relations.

In our algorithm, we construct the virtual relations:

$$\mathbf{W}^{(xc)} = \mathbf{U}_{t-1}^{(x)} \Lambda_{t-1} \quad \mathbf{W}^{(yc)} = \mathbf{U}_{t-1}^{(y)} \Lambda_{t-1}$$

$$= \begin{bmatrix} 0 & 0 & 0.33 \\ 0 & 0.11 & 0 \\ 0 & 0.22 & 0 \\ 0.22 & 0 & 0 \\ 0.11 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0.11 \\ 0 & 0.22 & 0.11 \\ 0 & 0.11 & 0.11 \\ 0.11 & 0 & 0 \\ 0.22 & 0 & 0 \end{bmatrix}$$

With observed relation \mathbf{W}_t and historic information encoded in virtual relations $\mathbf{W}^{(xc)}$ and $\mathbf{W}^{(yc)}$, we obtain clustering solution at time t :

$$\Lambda_t = \begin{bmatrix} 0.50 & 0 \\ 0 & 0.49 \end{bmatrix} \mathbf{U}_t^{(x)} = \begin{bmatrix} 0 & 0.53 \\ 0.17 & 0.21 \\ 0.19 & 0.26 \\ 0.28 & 0 \\ 0.36 & 0 \end{bmatrix} \mathbf{U}_t^{(y)} = \begin{bmatrix} 0 & 0.39 \\ 0 & 0.23 \\ 0 & 0.38 \\ 0.36 & 0 \\ 0.36 & 0 \\ 0.28 & 0 \end{bmatrix}$$

We can see the clustering results at time t is more consistent with the clustering results at time $t-1$, due to the aid of historic information. For example, the first and second entities (keyword) in \mathcal{S}_y are now assigned to the same clusters because of their previous co-occurring relationship at time $t-1$. It seems that the first and part of the second cluster at time $t-1$ are merged into the first cluster of t , and part of the second and the third cluster of $t-1$ are merged into the second cluster of t . This has been precisely captured by the transition matrix:

$$\mathbf{U}_t^{(c)} = \begin{bmatrix} 0.75 & 0 \\ 0.25 & 0.40 \\ 0 & 0.60 \end{bmatrix},$$

where the (i,j) entry of $\mathbf{U}_t^{(c)}$ indicates the transition probability from the i -th cluster at time $t-1$ to the j -th cluster at time t .

Computational complexity. By employing the sparse property of the data matrices, the update rules can be implemented efficiently. The most time-consuming step in the updating the solution is to compute all the entries in $\mu^{(r)}$ for each relation r . As can be seen in eq.<7>, to update Λ and $\mathbf{U}^{(g)}$ we can take advantage of the sparseness of the data matrix $\mathbf{W}^{(r)}$ and compute only the non-zero elements in $\mathbf{W}^{(r)}$. Let n denote the largest number of non-zero elements in the data matrices. This step has time complexity $O(nKL)$, where K is the number of clusters and L is the number of input relations. Usually K and L are much smaller than n and can be bounded by some constants; hence, the time complexity per iteration is linear in $O(n)$, i.e., the number of non-zero elements in all relational matrices.

3.4 Determination of Cluster Numbers. We discuss how to determine the number of clusters in this subsection. So far we have assumed the number of clusters, K , is given. However, since the number of clusters is usually not known beforehand, we propose an automatic mechanism to determine the number of clusters.

We extend the concept introduced in [20], the *modularity* Q , which measure the strength of the discovered clustering structure. Given a network partitioning into K clusters, the modularity Q is defined as:

$$Q(K) = \sum_{k=1}^K \left(l_{kk} / L - (l_k / 2L)^2 \right), \quad <8>$$

where L is the total number of links in the network, l_{kk} is the number of links between entities belonging to cluster k , l_k is the total degree of entities in cluster k (i.e. the number of links incident to the nodes in cluster k). The idea is to divide the network such that the number of links within clusters is higher than expected, i.e., the modularity is the deviation between fraction of edges within communities (expressed by the first term inside the summation) and the expected fraction of such edges (expressed by the second term inside the summation). Extensive experimental results have demonstrated that Q is an effective measure for evaluating the clustering structure in large networks, where a maximal Q leads to good clustering structure in the network [20,25].

Despite its advantages, *modularity* Q cannot be directly applied to our method because of its hard membership assumption – each entity can belong to only a single cluster. In a multi-relational network, clusters tend to blend together and each entity (an author, a paper, etc.) might be involved in multiple clusters through any of the relations. Hence, the expected links computed based on hard membership will be invalid. To overcome this, we modify Q to incorporate soft membership in a multi-relational network. We define the soft modularity Q_S to be:

$$Q_S(K) \propto \sum_r \sum_{k=1}^K (A_{kk}^{(r)} - A_{k\bullet}^{(r)} A_{\bullet k}^{(r)}), \text{ where } A_{kk}^{(r)} = \sum_{ij} \mathbf{W}_{ij}^{(r)} P_{ki} P_{kj}, \quad <9>$$

$$A_{k\bullet}^{(r)} = \sum_i P_{ki} \sum_j \mathbf{W}_{ij}^{(r)}, A_{\bullet k}^{(r)} = \sum_j P_{kj} \sum_i \mathbf{W}_{ij}^{(r)},$$

and where $\{\mathbf{W}^{(r)}\}$ are the data matrices corresponding to observed relations, p_{ki} and p_{kj} are soft membership of entities i and j with respect of cluster k and can be computed as described in previous subsection. The soft modularity Q_S evaluates the sum of the contribution from each cluster in each relation. The term $A_{kk}^{(r)}$ measures the expected number of links within a cluster, on the network corresponding to $\mathbf{W}^{(r)}$. The two terms $A_{\cdot k}^{(r)}$ and $A_{k \cdot}^{(r)}$ measure the expected total out-degree and in-degree of the cluster k with respect to $\mathbf{W}^{(r)}$. Again, the best clustering structure is given by the maximal Q_S .

4. Dynamic Network Contour-map

Now we present the Dynamic Network Contour-map (DNC) to visualize the results of DRC. We first describe how to summarize entities as an entity network in section 4.1. Then we present how to visualize the evolution over time as well as dimension switching in section 4.2.

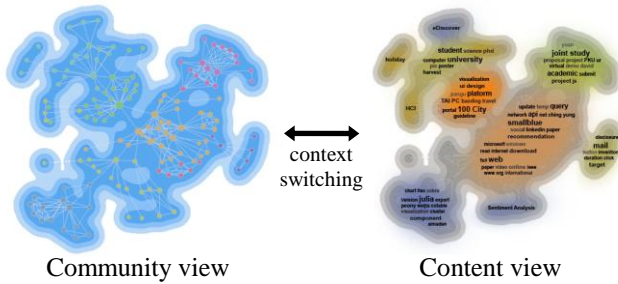


Figure 3: Context switching example

4.1 Entity Network Construction and Summarization.

To visualize the evolution of a large set of entities (e.g., authors, conferences, keywords), the most important thing is to summarize them into a smaller scale which can be efficiently and clearly visualized. In particular, we need to address the following questions:

- (1) What are the most important entities?
- (2) How to compute the similarity between those entities?
- (3) How to construct the entity network for visualization?

Next we present our approach to address these questions, where we leverage the clustering results of DRC.

4.1.1 Select important entities. Typically, the total number of entities is large, e.g., there are over 100K authors and keywords in the DBLP dataset. However, for human understanding, we need to smartly select a small set of representative entities to visualize. The importance of an entity should factor in the data from all the relations at the current time as well as the entity’s historical importance. Fortunately, we can leverage the soft-clustering result of DRC to achieve this seemingly complex goal. In particular, we propose two different ways to measure the importance of an entity:

- *Cluster importance* is the importance score of entity i in cluster k , which is the conditional probability of entity i given cluster k , i.e., $p_{i|k}$.

- *Total importance* is the importance score of entity i over all clusters:

$$s_i = \sum_k p_{i|k} P_k = \sum_k (\mathbf{U}^{(q)} \mathbf{\Lambda})_{ik}, \quad <10>$$

where s_i is the contribution of entity i . This is equivalent to the marginal probability of entity i described in section 3.3.

Given these two importance scores, we propose two sampling schemes to select a subset of entities for network visualization: (1) *Biased sampling* selects entities with large total importance score s_i . This scheme will choose those entities with global importance. (2) *Biased stratified sampling* selects a fixed number of entities from each cluster, and the selection criterion within a cluster is biased towards the entities with large cluster importance score $p_{i|k}$.

In practice, a mixed version of these two may give a good balance, since biased sampling covers the globally important entities, while biased stratified sampling covers all clusters including the small ones.

4.1.2 Similarity computation and network construction.

Once we find important entities, the next question is how to measure the similarity between those entities. Again, we leverage the soft-clustering results to construct this measure. We define the similarity between entities i and j as the similarity of their soft-cluster membership distributions over clusters, which is defined by:

$$s_{ij} = \frac{\phi_i^T \phi_j}{|\phi_i| |\phi_j|} \quad <11>$$

where s_{ij} is the cosine similarity measure of the membership distributions between entities i and j , and where ϕ_i is a vector of entity i ’s soft-clustering membership with the k -th element indicating p_{ki} . A large value of s_{ij} indicates that entities i and j contribute similar weights to the same cluster(s).

Given the pair-wise similarity measure, next we construct the entity network for visualization. In particular, we add an edge between entities i and j if the similarity s_{ij} is among the top- a highest similarity scores. In practice, we choose a to be 5% so that network is sparse but often largely connected. For the disconnected entities, we connect them with the most similar entities in the large connected component. The resulting network is a community-oriented network, in which entities within a cluster often form a clique, but since it is based on soft-clustering, the network is also smooth and connected, which is ideal for visualization.

4.1.3 Contour-map layout. Since not all entities are displayed, we superimpose a contour-map over the entity network to depict the density distribution of all entities. The idea is that we first put kernel functions over all the sampled entities then estimate the joint density distribution of all kernels. Since we want to show the density distribution of all entities (not only the ones in the sampled entity network), we assign each of the non-selected entities to one of the closest sample entities. This operation is well-studied in the

database literature as reverse-kNN query. We employ the idea to display a large set of entities as a community network overlaid contour-map. An example is provided later in Figure 8.

4.2 Dynamic and Context Switching. In this subsection, we first present how to layout the entity network, then show how to track the community evolution over time, and finally illustrate a way to display the related entities of a different dimension.

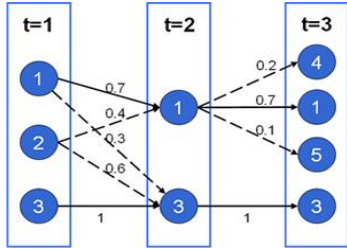


Figure 4: Tracking communities over time: e.g., cluster 1 at $t=2$ is merged from clusters 1 and 2 at $t=1$, and we assign cluster id 1 since cluster 1 has higher probability (.7 over .4).

4.2.1 Dynamic network layout. The most important thing in dynamic network visualization is to maintain a user’s mental map. Therefore, successive layouts of similar graphs should have minimal changes (stability). Furthermore, each of such layouts should still effectively convey the properties of the underlying graph (readability). Thus, our goal is to produce a sequence of graph layouts that balance between the stability and readability of the resulting visualization. With these two goals in mind, the objective is to minimize:

$$\sum_{i < j} \frac{1}{s_{ij}^2} (\|X_i - X_j\| - 1/s_{ij})^2 + \sum_i (X_i - X_i')^2 \quad <12>$$

where X_i and X_i' are the current and old coordinates of the i -th entity, s_{ij} is the similarity value calculated by <11>. The first term specifies the deviation between displayed distance $\|X_i - X_j\|$ and model distance $1/s_{ij}$. The second term is to regularize the change between successive layouts; therefore, it improves the stability. To minimize this objective, we design the following update rule:

$$X_i \leftarrow \sum_{i \neq j} \omega_{ij} \left(X_j' + X_j + \frac{(X_j' - X_j) + (X_i - X_j)}{s_{ij} \|X_i - X_j\|} \right) / \sum_{i \neq j} \omega_{ij} \quad <13>$$

Our layout method generates clustered visualization based on community information and entity similarity, which is suited for large scale social network visualization.

4.2.2 Tracking community evolution. To track the community structure, we assign a specific color and cluster id to each entity based on its soft-clustering scores. More formally, the cluster id for entity i is simply its most probable cluster: $\operatorname{argmax}_c p_{i|c}$. Since the cluster id may change over time, it is important to be able to track the community change over time. For this, we leverage the transition probability between clusters of consecutive timestamps $p_{c|c'}$, aka. the factor matrix for the virtual relation

$U^{(c)}$. More specifically, we map a cluster c at time t to a cluster c' at time $t-1$ if c' has been assigned and $p_{c|c'}$ is the maximum among all the available cluster ids. For any cluster with no matching cluster id, we create a new cluster id. One example is illustrated in Figure 4.

4.2.3 Context switching. Now we present the proposed method to switch context between dimensions. E.g., in DBLP data, based on the visualization of author network evolution, we would like to switch the context to another dimension say keywords. Formally, the problem is to place a subset of entities of dimension B based on a visualization of sampled entities of dimension A.

To solve this problem, we first select the representative entities of dimension B, and then place those representative entities onto the existing visualization of dimension A. To select the representative entities, we can leverage the similar sampling schemes proposed in 4.1.1. To place the representative entities, we again leverage the soft-clustering results. We first compute the similarity between the representative entities of dimension B and the visualized entities of dimension A. Then we place each representative entity in dimension B near the entity of dimension A with the largest similarity score.

5. Experiments

In this section, we first present a case study on the DBLP data through our proposed visualization framework (section 5.1), and then we evaluate the performance of our approach on synthetic datasets (section 5.2).

5.1 Case Study on DBLP Dataset We now demonstrate the capability of ContextTour for enhancing users’ understanding over complex relational data. We use a subset of DBLP data with 3 relations from 1995 to 2008: (1) A conference-by-author matrix where the (i,j) element is a binary variable indicating whether author j published at conference i at that given year. (2) A conference-by-conference matrix where the (i,j) element is the number of common authors in both conference i and j at that given year. (3) An author-by-keyword matrix where the (i,j) element is a binary variable indicating whether author i published a paper using keyword j at that given year.

The sizes of the three dimensions are 81 conferences, 10K authors, and 30K keywords. We manually picked 81 conferences from various areas in Computer Science. We apply ContextTour on these three relations to obtain the dynamic community visualization.

Community. The community structure is very prominent. As in Figure 8 (column 1), all the conferences are well clustered into communities: DB, AI, Systems, etc. Besides displaying meaningful communities, ContextTour also preserves good cross-community similarity. E.g., Theory is very close to Crypto community as shown in the conference view, which makes sense since Theory and Crypto communities do have a lot of overlaps.

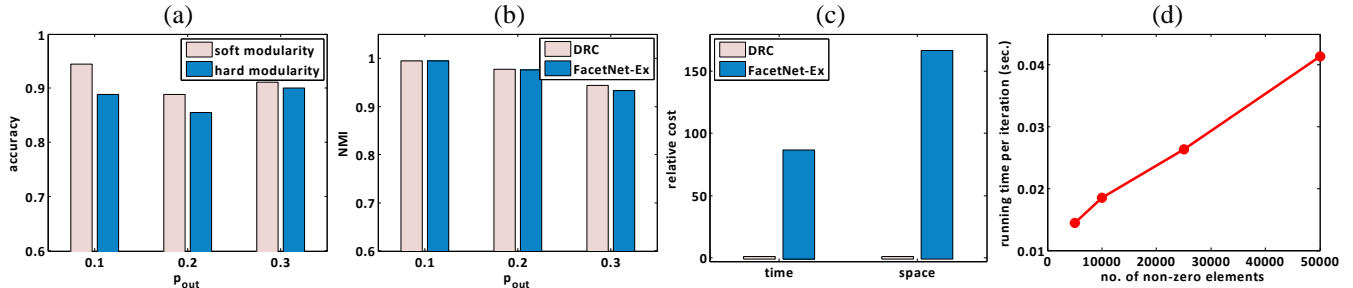


Figure 5: (a) Accuracy of cluster numbers: Determining the cluster numbers by the soft modularity performs well over different degree of community noise (p_{out}). (b) Accuracy of clustering membership in terms of normalized mutual information (NMI): DRC performs slightly better than the extended FacetNet algorithm (FacetNet-Ex). (c) The relative speed and space costs of DRC, compared with FacetNet-Ex: DRC is 85-165X more efficient. (d) Running time per iteration (sec.) over the number of non-zero elements in networks, which indicates DRC scales linearly with data size.

Core member. The member font size is proportional to p_{ic} . E.g., SIGMOD and VLDB are visually more significant than EDBT and ICDT in Figure 8 due to larger p_{ic} .

Evolution. The key benefit of ContextTour is the capability of dynamic community visualization. A smooth video of the transition can be viewed at <http://www.dasfa.net/contextour/>. To illustrate the evolution, Figure 8 presents three snapshots (1996, 2002, 2005). As tracked by the red arrow in Figure 8, KDD community went through an interesting evolution. In 1996, KDD is a relative isolated conference, which is close to AI. In 2002, KDD can be viewed a bridge between AI and DB, but more closer to AI. Meanwhile, KDD is forming its own community with other new DM conference like ICDM, SDM shown in 2002. At 2005, DM becomes an independent community, which is farther away from the core DB conferences like SIGMOD and VLDB.

Context switching. In Figure 8, we present three different views of the data. Besides conferences, we can see the key authors and keywords in each community. E.g., we can easily see how the topics in AI+DM community evolve from classifier in 1996, to Bayesian method in 2002 and 2005.

5.2 Performance Evaluation In this section, we first use synthetic datasets to study the clustering performance and scalability of the Dynamic Relational Clustering (DRC) algorithm. Then we discuss the legibility of ContextTour for presenting soft-community information.

Synthetic datasets. To demonstrate the effectiveness of our DRC algorithm, we generate a family of dynamic tripartite networks by the following process. Each dataset contains N entities in each of three different dimensions (as indicated by three different shapes in Figure 6). Initially (at time step $t=1$), these entities belong to three communities. Each entity i of dimension 1 will attach m edges to dimension 2 and m edges to dimension 3. The entities being attached are selected as follows: with probability $1-p_{out}$, entity i will randomly connect to an entity within the same community, and with probability p_{out} , entity i will randomly connect to an entity outside its community. The evolution process is simple but useful for evaluating the dynamic clustering: at time step

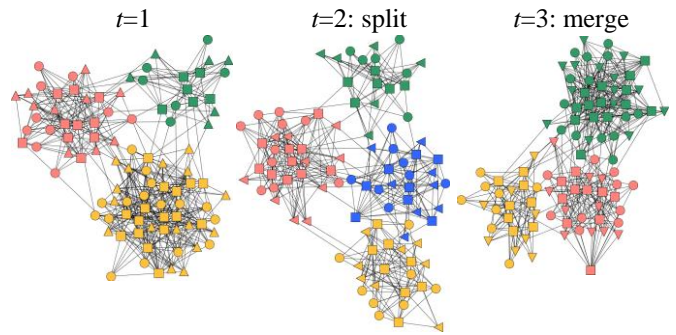


Figure 6: A realization of the synthetic dynamic tripartite networks, with number of entities $N=32$ (for each mode), average entity degree $m=11$ and $p_{out}=0.05$. The entities of three different modes are represented by different shapes (circle, rectangle and triangle), and different communities are indicated by different colors. There are three communities at time step $t=1$. At $t=2$, the orange community splits into two, and at $t=3$, the blue community merges into the green one.

$t=2$, the bigger community splits into two, and at $t=3$, the last community merge to the first community. During both steps, the entities from dimension 1 will attach to other entities as described before according to the new community assignment. Figure 6 shows an example of the dynamic synthetic networks with number of entities $N=32$ (for each mode), average entity degree $m=11$ and out-linking probability $p_{out}=0.05$. The out-linking probability p_{out} can be viewed as the degree of noise to an ideal clustering structure.

Clustering performance. We test our DRC algorithm on the synthetic networks generated with different parameter values for N and p_{out} , and m is set to be half of N . At each time step, the cluster numbers are automatically determined by calculating the soft modularity defined by $\langle 9 \rangle$. We evaluate the clustering performance from two aspects: (a) the accuracy of cluster numbers, and (b) the accuracy of clustering membership assigned to entities. The accuracy of cluster numbers is measured based on the true cluster

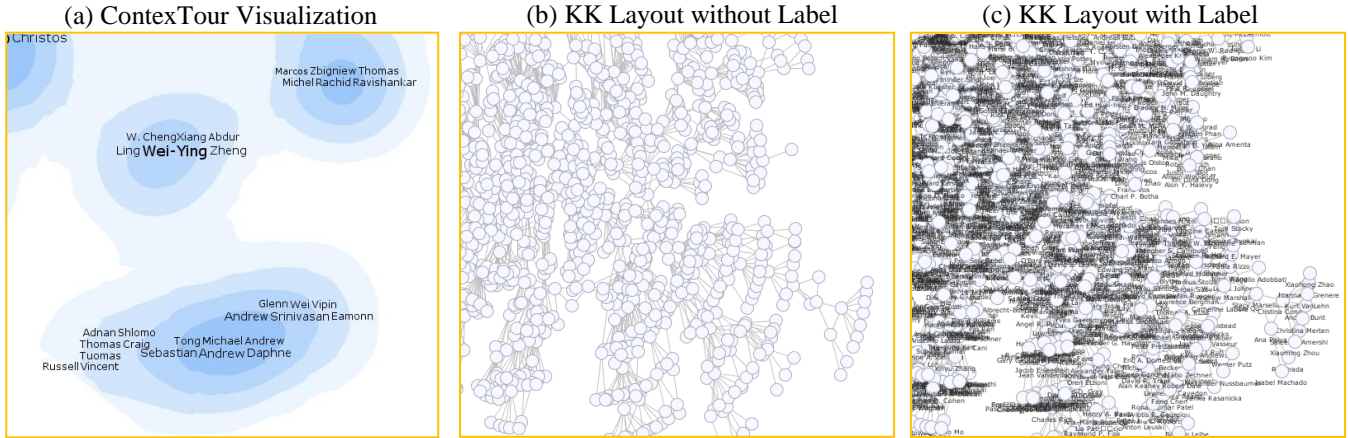


Figure 7: Visualization comparison – (a) ContextTour which enhances community by providing the optimized contour. (b) KK network layout [10] does not consider community; (c) is the same visualization as (b) with all the labels.

numbers of the synthetic datasets, which are 3, 4 and 3 from $t=1$ to $t=3$.

Figure 5(a) shows the mean accuracy of the cluster numbers against different degree of community noise given by the out-linking probability p_{out} , compared with hard modularity. Our soft modularity outperforms the hard modularity method [20] in selecting the cluster numbers. The accuracy of clustering membership is computed by the normalized mutual information (NMI) between the derived membership and the ground truth, where a higher NMI indicates better accuracy (ref. [27] for detailed definition of the mutual information between two partitions). We compare the results of our DRC algorithm to a baseline method – the extended version of the FacetNet algorithm proposed in [17] which allows change in cluster number. We denote this baseline method as *FacetNet-Ex*. The comparison is shown in Figure 5(b), which indicates that our DRC algorithm outperforms the FacetNet-Ex algorithm. Next, we show the scalability performance of our DRC method.

Scalability. We use the synthetic datasets described previously to illustrate the scalability of our clustering algorithm. We compare the time and storage costs of our algorithm with FacetNet-Ex. Figure 5(c) shows the comparison for the case $N=2000$, where the speed and space costs are normalized by the costs for running DRC. In this case, our DRC algorithm is 85-165X more efficient than the baseline, and our performance gain increase with N . Figure 5(d) shows the running time per iteration for $N=2000$ over different number of edges, i.e. the number of non-zero elements in relational matrices. The running time per iteration scales linearly with the data size (the number of non-zero elements).

Legibility. We compared our visualization design with conventional node-link representation through the well-known Kamada and Kawai (KK) algorithm [10]. Figure 7 shows a coauthor network with 1500 entities and 3000 edges visualized by our ContextTour (Figure 7(a)) and a Force-

Directed layout graph (Figure 7 (b) and (c)). The comparison shows that ContextTour is able to produce a visual presentation that better illustrates the soft community structure and related content, while KK displays the whole network with a cluttered result. Our informal user study shows that users prefer the ContextTour representation, since it is not only aesthetically pleasing, but also allows users to better comprehend the complex relational clustering results.

6. Conclusions

In this paper, we present a visual analytic solution ContextTour that detects and visualizes meaningful community evolution in dynamic multi-relational data. In order to achieve better user understanding, we argue data mining and visualization need to be combined and redesigned to maximize the values of both. With this goal, ContextTour introduces two novel components: DRC to cluster time-evolving multi-relational data into smooth soft-clusters with clustering transition information, and DNC to summarize and display DRC results into a dynamic, multi-aspect visualization. From a design point of view, DRC addresses the data challenge raised by social media applications; DNC addresses the understanding challenge in the complex patterns. Finally, we demonstrate the interesting patterns detected by ContextTour in the DBLP data as well as present the scalability and usability studies.

For future work, we plan to apply ContextTour to more diverse domains and extend visualization technique to support streaming data for real-time applications.

7. Appendix

Proof of Theorem 1. We employ the concavity of log function to prove the correctness of eq.<7>. Because $\log(\sum_k a_{ik}b_{kj})$ is a convex function, the following equality holds for all i, j , and $\sum_k \mu_{ijk} = 1$:

$$-\log\left(\sum_k a_{ik}b_{kj}\right) \leq -\left(\sum_k \mu_{ijk} \log \frac{a_{ik}b_{kj}}{\mu_{ijk}}\right), \text{ where } \mu_{ijk} = \frac{a_{ik}b_{kj}}{\sum_k a_{ik}b_{kj}}$$

Hence, we have:

$$\begin{aligned}
& J(\Lambda, \{\mathbf{U}^{(r)}\}) \\
&= \sum_{r'} \omega^{(r')} \sum_{ij} \left(-\mathbf{W}_{ij}^{(r')} \log \sum_k \Lambda_k \times \{\mathbf{U}_{ik}^{(q)}\}_{q \in \Gamma(r')} + \sum_k \Lambda_k \times \{\mathbf{U}_{ik}^{(q)}\}_{q \in \Gamma(r')} \right) + \text{const} \\
&\leq \sum_{r'} \omega^{(r')} \sum_{ijk} -\mathbf{W}_{ij}^{(r')} \mu_{ijk}^{(r')} \log \frac{\Lambda_k \times \{\mathbf{U}_{ik}^{(q)}\}_{q \in \Gamma(r')}}{\mu_{ijk}^{(r')}} + \Lambda_k \times \{\mathbf{U}_{ik}^{(q)}\}_{q \in \Gamma(r')} + \text{const} \\
&\stackrel{\text{def}}{=} \sum_{r'} \omega^{(r')} \mathcal{Q}^{(r')}(\Lambda, \{\mathbf{U}^{(q)}\}_{q \in \Gamma(r)}; \mu_{ijk}^{(r')})
\end{aligned}$$

where $\mu^{(r)}$ is defined as in eq.<7>. With the constraints $\sum_i \mathbf{U}_{ik}^{(q)} = 1$ and $\sum_k \Lambda_k = 1$, the Lagrangian of \mathcal{Q} is defined as:

$$\begin{aligned}
L &= \sum_{r'} \omega^{(r')} \mathcal{Q}^{(r')}(\Lambda, \{\mathbf{U}^{(q)}\}_{q \in \Gamma(r)}; \mu_{ijk}^{(r')}) + \\
&\quad \sum_q \varepsilon_q \left(\sum_i \mathbf{U}_{ik}^{(q)} - 1 \right) + \varepsilon_\Lambda \left(\sum_k \Lambda_k - 1 \right)
\end{aligned}$$

Update Λ : with $\{\mathbf{U}^{(q)}\}$ fixed, we have:

$$\frac{\partial L}{\partial \Lambda_k} = \sum_{r'} \omega^{(r')} \sum_{ij} -\mathbf{W}_{ij}^{(r')} \mu_{ijk}^{(r')} / \Lambda_k + \varepsilon_\Lambda + \text{const} = 0$$

$$\frac{\partial L}{\partial \varepsilon_\Lambda} = \sum_i \Lambda_k - 1 = 0$$

By solving the equations, we obtain the update rule for Λ .

Update $\mathbf{U}^{(q)}$: with Λ and $\{\mathbf{U}^{(q')}\}_{q' \neq q}$ fixed, we have:

$$\frac{\partial L}{\partial \mathbf{U}_{ik}^{(q)}} = \sum_{r'} \omega^{(r')} \frac{\partial \mathcal{Q}^{(r')}}{\partial \mathbf{U}_{ik}^{(q)}} + \varepsilon_q + \text{const} = 0, \quad \frac{\partial L}{\partial \varepsilon_q} = \sum_i \mathbf{U}_{ik}^{(q)} - 1 = 0$$

To solve these equations, observe that if q correspond the first mode of $\mathbf{W}^{(r')}$, we have:

$$\frac{\partial \mathcal{Q}^{(r')}}{\partial \mathbf{U}_{ik}^{(q)}} = \sum_j -\mathbf{W}_{ij}^{(r')} \mu_{ijk}^{(r')} / \mathbf{U}_{ik}^{(q)}$$

Otherwise, we have:

$$\frac{\partial \mathcal{Q}^{(r')}}{\partial \mathbf{U}_{ik}^{(q)}} = \sum_j -\mathbf{W}_{ji}^{(r')} \mu_{ijk}^{(r')} / \mathbf{U}_{ik}^{(q)}$$

Define $v_{q;ij}^{(r')}$ as in eq.<7>, we have:

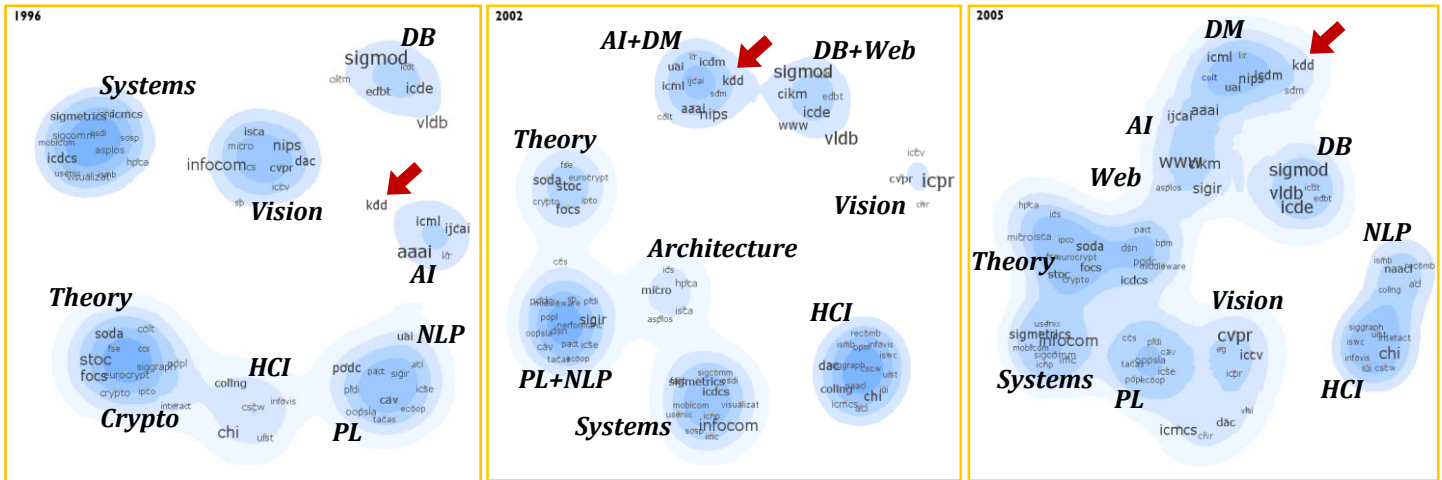
$$\frac{\partial L}{\partial \mathbf{U}_{ik}^{(q)}} = \sum_{\{r': q \in \Gamma(r')\}} \omega^{(r')} \sum_j -v_{q;ij}^{(r')} \mu_{ijk}^{(r')} / \mathbf{U}_{ik}^{(q)} + \varepsilon_q + \text{const} = 0$$

By solving these, we obtain the update rule for $\mathbf{U}^{(q)}$. \square

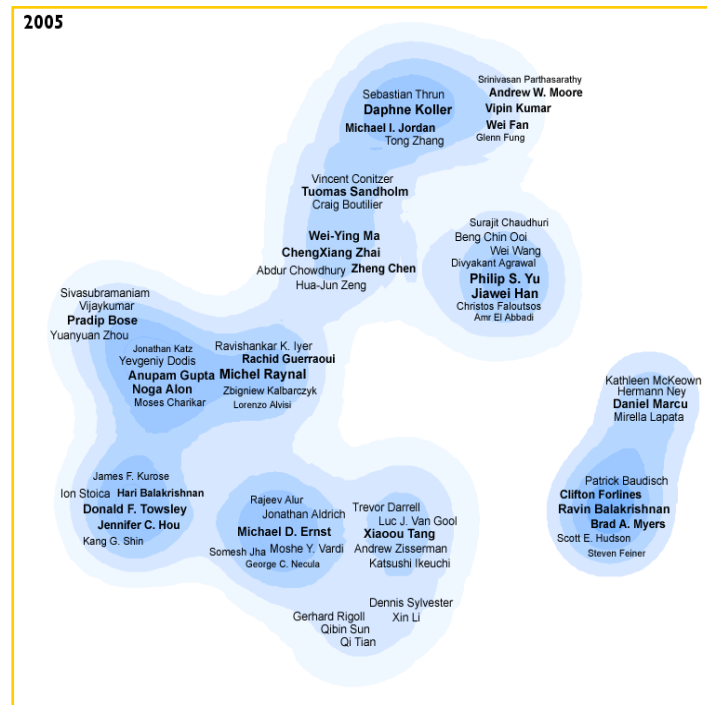
References

- [1] L. BACKSTROM, D. HUTTENLOCHER, J. KLEINBERG and X. LAN (2006). *Group formation in large social networks: membership, growth, and evolution*, SIGKDD, 44-54, 2006.
- [2] R. BEKKERMAN, R. EL-YANIV and A. MCCALLUM (2005). *Multi-way distributional clustering via pairwise interactions*, ACM Intl. Conf. Proc. Series, 41-48,
- [3] I. BORG and P. GROENEN (2005). *Modern multidimensional scaling: Theory and applications*, Springer.
- [4] U. BRANDES and D. WAGNER (1997). *A Bayesian paradigm for dynamic graph layout*. *Lecture Notes in Computer Science* **1353**: 236-247.
- [5] U. BRANDES, D. FLEISCHER and T. PUPPE (2006). *Dynamic spectral layout of small worlds*. *Lecture Notes in Computer Science* **3843**: 25.
- [6] D. CHAKRABARTI, R. KUMAR and A. TOMKINS (2006). *Evolutionary clustering*, SIGKDD, 554-560,
- [7] C. CHEN and L. CARR (1999). *Visualizing the evolution of a subject domain: a case study*, Proceedings of the IEEE Visualization Conference, 449-452, 1999.
- [8] S. DIEHL and C. GORG (2002). *Graphs, they are changing*. *LECTURE NOTES IN COMPUTER SCIENCE*: 23-30.
- [9] K. FUJIMURA, S. FUJIMURA, T. MATSUBAYASHI, T. YAMADA and H. OKUDA *Topigraphy: Visualization for Large-scale Tag Clouds*.
- [10] T. KAMADA and S. KAWAI (1989). *An algorithm for drawing general undirected graphs*. *Information processing letters* **31**(1): 7-15.
- [11] M. KIRSTEN and S. WROBEL (1998). *Relational distance-based clustering*, Inductive logic programming: 8th international conference, ILP-98, Madison, Wisconsin, USA, July 22-24, 1998: proceedings, 261,
- [12] T. KOHONEN (1982). *Self-organized formation of topologically correct feature maps*. *Biological cybernetics* **43**(1): 59-69.
- [13] G. KUMAR and M. GARLAND (2006). *Visual exploration of complex time-varying graphs*. *IEEE Transactions on Visualization and Computer Graphics* **12**(5): 805.
- [14] D. LEE and H. SEUNG (2001). *Algorithms for non-negative matrix factorization*, NIPS, 556-562, 2001.
- [15] J. LESKOVEC, L. BACKSTROM, R. KUMAR and A. TOMKINS (2008). *Microscopic Evolution of Social Networks*, SIGKDD, 2008.
- [16] T. LI and S. ANAND (2007). *Diva: a variance-based clustering approach for multi-type relational data*, SIGKDD, 147-156, 2007.
- [17] Y.-R. LIN, Y. CHI, S. ZHU, H. SUNDARAM and B. L. TSENG (2008). *FaceNet: A Framework for Analyzing Communities and Their Evolutions in Dynamics Networks*, WWW, 2008.
- [18] B. LONG, X. WU, Z. ZHANG and P. YU (2006). *Unsupervised learning on k-partite graphs*, SIGKDD, 317-326, 2006.
- [19] L. LOVASZ and M. PLUMMER (1986). *Matching theory*, North Holland.
- [20] M. NEWMAN and M. GIRVAN (2004). *Finding and evaluating community structure in networks*. *Physical Review E* **69**(2): 26113.
- [21] P. SARKAR and A. MOORE (2005). *Dynamic social network analysis using latent space models*. *SIGKDD Explorations Newsletter* **7**(2): 31-40.
- [22] J. SUN, S. PAPADIMITRIOU, C. LIN, N. CAO, S. LIU and W. QIAN *MultiVis: Content-based Social Network Exploration Through Multi-way Visual Analysis*.
- [23] L. TANG, H. LIU, J. ZHANG and Z. NAZERI (2008). *Community evolution in dynamic multi-mode networks*, SIGKDD, 2008.
- [24] X. WANG, J. SUN, Z. CHEN and C. ZHAI (2006). *Latent semantic analysis for multiple-type interrelated data objects*, SIGIR, 236-243, 2006.
- [25] S. WHITE and P. SMYTH (2005). *A spectral clustering approach to finding communities in graphs*, SIAM, 2005.
- [26] J. WISE, J. THOMAS, K. PENNOCK, D. LANTRIP, M. POTTIER, A. SCHUR and V. CROW (1995). *Visualizing the non-visual: Spatial analysis and interaction with information from text documents*, Proceedings of the 1995 IEEE Symposium on Information Visualization,
- [27] W. XU, X. LIU and Y. GONG (2003). *Document clustering based on non-negative matrix factorization*, SIGIR, 267-273, 2003.

Conference View



Author View



Keyword View

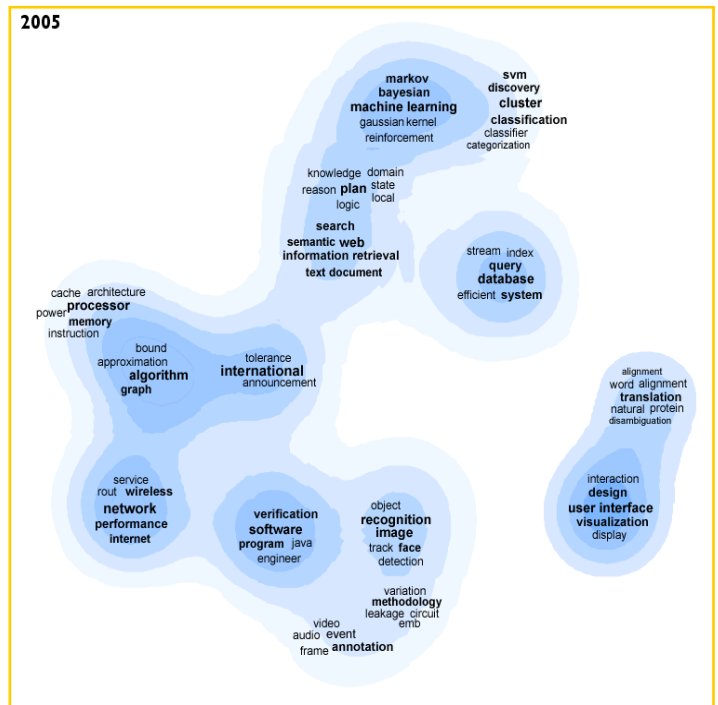


Figure 8: Three snapshots of the conference community visualization (1996,2002,2005). Different patterns are observed:

- 1) *Communities*: we observe conference communities like DB, AI, HCI, NLP, systems, and architecture, IR and PL.
- 2) *Cores*: the different font sizes indicate the importance of that conference to its community. E.g., sigmod, vldb are more important than icde and icdt for DB community at 1996, according to the observed relations.
- 3) *Evolution*: the change of relative location indicates the change of social ties. E.g., KDD starts at 1996 as a relatively isolated conference, and then becomes the bridge between AI and DB in 2002. Finally the ties are weakened at 2005, since DM becomes a mature community of its own, which actually is more closer to Machine learning than DB at 2005.
- 4) *Context switching*: all three views (columns) focus on three different dimensions. In author and keyword view we can see the representative authors and keywords in each community with respect to the conference layout. E.g., in 2005, the keywords in DB community include stream, query, database, index; the authors include Philip Yu, Beng Chi Ooi, etc.