MultiVis: Content-based Social Network Exploration Through Multi-way Visual Analysis

Jimeng Sun, Spiros Papadimitriou, Ching-Yung Lin, Nan Cao, Shixia Liu, Weihong Qian
IBM Research

Abstract

With the explosion of social media, scalability becomes a key challenge. There are two main aspects of the problems that arise: 1) *data volume*: how to manage and analyze huge datasets to efficiently extract patterns, 2) *data understanding*: how to facilitate understanding of the patterns by users?

To address both aspects of the scalability challenge, we present a hybrid approach that leverages two complementary disciplines, *data mining* and *information visualization*. In particular, we propose 1) an analytic data model for content-based networks using tensors; 2) an efficient high-order clustering framework for analyzing the data; 3) a scalable context-sensitive graph visualization to present the clusters.

We evaluate the proposed methods using both synthetic and real datasets. In terms of computational efficiency, the proposed methods are an order of magnitude faster compared to the baseline. In terms of effectiveness, we present several case studies of real corporate social networks.

1 Introduction

With the emergence and rapid proliferation of social media, such as instant messaging (e.g., IRC, AIM, MSN, Jabber, Skype), sharing sites (e.g., Flickr, Picassa, YouTube, Plaxo), blogs (e.g., Blogger, WordPress, LiveJournal), wikis (e.g., Wikipedia, PBWiki), microblogs (e.g., Twitter, Jaiku), social networks (e.g., MySpace, Facebook, Ning), to mention a few, there is little doubt that networks which arise from all sorts of digital and social media, combining content with people and context, are becoming prevalent and here to stay. Therefore, there is a clear need for methods and techniques to analyze, navigate and search them. Such social networks arise from pairwise communications, such as email, instant messaging (IM) or mobile text messaging (SMS).

Motivating application The SmallBlue project [1, 14] within IBM (also part of the Lotus Atlas software suite) aims to analyze the social network within a corporation, thereby answering the important questions of "who are experts?" and "how to reach them?". It relies on a small client application that users consent to install on their personal machines, which locally analyzes their email outbox and their outgoing IM chat transcripts. As of this writing, the number of IBMers that have installed the software is close to 6,000, and growing fast due to network effects (about 3,000 new subscribers since the beginning of the year). The number of distinct recipients is approaching to 330,000 (i.e., close to the entire workforce of IBM). The number of emails and chat transcripts is over eight million. The number of keyword stems in the raw dataset, across all languages, approaches eight million.

Problem motivation and statement Typical approaches to analyzing such networks focus on either the content (e.g., list of words or terms) or on the pairwise associations, in isolation. On one hand, content-based analysis such as latent semantic indexing [29] analyzes the relationship between documents and terms by identifying hidden concepts related to documents and terms. On the other hand, social network analysis, such as graph partitioning, tries to identify communities among people based only on links among individuals. In the past, many techniques considered these two aspects separately. In this case, ad-hoc post-processing is necessary to glue the results from each aspect (content and network), which creates both performance and quality problems. The main reason is that, in many traditional settings, content is associated with *nodes* in the graph (e.g., words in a web page) rather than with *edges* (e.g., the words used in all emails between two specific individuals). Several approaches have appeared in the research literature (e.g., [29, 23, 11, 28]) to deal with graphs where additional information may be attached to individual nodes. However, the nature of the data we consider is different. Perhaps the only widely available public corpus which allows simultaneous analysis of both content and network aspects is the Enron email dataset.

Our goal is to analyze, summarize, navigate and search this growing corpus of valuable information. For example, a business analyst may wish to answer the following questions:

- What are the key topics or areas of expertise within the company?
- Which are the key groups of experts in each of these areas?
- How to navigate through these expertise groups to find relevant experts?

and so on. Our goal is to precisely provide scalable tools that can analyze the corpus and help answer questions such as the ones above.

When dealing with a network with hundreds of thousands of people and millions of terms, *scalability* is a key challenge. This involves both

1. *Computational complexity* involved in processing the data and extracting significant patterns (such as those mentioned above), and
2. *Information complexity* involved in presenting the huge amount of data in a way that is easy for users to navigate and comprehend.

These two aspects are fundamentally intertwined. We thus introduce a comprehensive data modeling, mining and visualization workflow to address these challenges.

Our contributions More specifically, the main contributions of this paper are:

1. We introduce a method to simultaneously model both the network and content aspects as tensors.
2. We present robust and scalable single-mode and cross-mode clustering methods using tensor dimensionality reduction and biased sampling.
3. We bridge these clustering results through novel hierarchical context-specific visualizations.

The rest of the paper is organized as follows: Section 2 overviews the necessary background on tensor algebra. Section 3 introduces our tensor-based data model, Section 4 formalizes the problem and Section 5 presents our proposed framework for content-based network analysis. Section 6 gives an overview of our visualization framework and describes case studies on real datasets. Section 7 presents experimental results that validate the computational efficiency of our methodology. Finally, Section 8 discusses related work and Section 9 concludes.

2 Background

In this section, we introduce the notation and define the key tensor operations required in this paper.

2.1 Tensor

Definition 2.1. (Tensor) *A tensor is a multi-way array. The* order *of a tensor is the number of modes or ways. Within each mode, there are multiple dimensions. The* dimensionality *of a mode is the number of dimensions in that mode.*

For example, a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{6\times 7\times 8}$ has 3 modes with dimensionalities of 6, 7, and 8, respectively.

Higher-order (N-way with $N \geq 3$) tensors are denoted by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{X}}$. Matrices (tensors of order two) are denoted by boldface capital letters, e.g., $\mathbf{A}$; vectors (tensors of order one) are denoted by boldface lowercase letters, e.g., $\mathbf{a}$; and scalars are denoted by lowercase letters, e.g., a. The ith entry of a vector $\mathbf{a}$ is denoted by a_i, element (i, j) of a matrix $\mathbf{A}$ by a_{ij}, and element (i, j, k) element of a third-order tensor $\boldsymbol{\mathcal{X}}$ by x_{ijk}. Indices typically range from 1 to their capital version, e.g., $i = 1, \ldots, I$. The n-th element in an ordered collection is denoted by a superscript in parentheses. For example, $\mathbf{v}^{(n)}$ denotes the n-th vector in a collection. Therefore, $v_i^{(n)}$ denotes the ith element on the n-th vector $\mathbf{v}^{(n)}$. Likewise, $\mathbf{A}^{(n)}$ denotes the n-th matrix in a collection. Also, $\mathbf{a}_{:r}^{(n)}$ and $\mathbf{a}_{r:}^{(n)}$ denote the r-th column and row, respectively, of the matrix $\mathbf{A}^{(n)}$. Finally, we denote the entire ordered collection of all $\mathbf{A}^{(n)}$ matrices, for all n, by omitting the parenthesized superscript and using braces, i.e., $\{\mathbf{A}\}$.

Definition 2.2. (Tensor norm) *The norm of an N-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_M}$ is*

$$\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_M=1}^{I_M} x^2_{i_1 i_2\cdots i_M}}.$$

2.2 Basic Tensor Operations

Definition 2.3. (Matricization) Matricization*, also known as* unfolding *or* flattening*, is the process of reordering the elements of an N-way array into a matrix.*

The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1\times\cdots\times I_M}$ is denoted by $\mathbf{X}_{(n)}$ and arranges the mode-n fibers to be the columns of the matrix $\mathbf{X}_{(n)}$. Specifically, tensor element $(i_1, i_2, \ldots, i_N)$ maps to matrix element (i_n, j) where

$$j = 1 + \sum_{\substack{k=1\\k\neq n}}^{N}(i_k - 1)J_k, J_k = \prod_{\substack{m=1\\m\neq n}}^{k-1} I_m.$$

Similarly, we can define the *vectorization* operation, which linearizes the tensor into a vector.

Definition 2.4. (Mode-n product) *The* mode-n matrix product *of a tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$ *with a matrix* $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ *is denoted by* $\mathcal{X} \times_n \mathbf{U}$ *and is of size* $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_M$. *Element-wise, we have*

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j\, i_{n+1} \cdots i_M} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_M}\, u_{j i_n}.$$

The mode-n product transforms $\mathcal{X}$ to a new tensor $\mathcal{X} \times_n \mathbf{U}$ by applying the linear transformation described by the matrix $\mathbf{U}$ to each of the mode-n fibers of $\mathcal{X}$. From [5], we adopt the following shorthand notation for multiplication in every mode:

$$\mathcal{X} \times \{\mathbf{A}\} \equiv \mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_M \mathbf{A}^{(M)}.$$

It is also useful to introduce notation for multiplication in every mode but one [5]:

$$\mathcal{X} \times_{-n} \{\mathbf{A}\} \equiv \mathcal{X} \times_1 \mathbf{A}^{(1)} \cdots \times_{n-1} \mathbf{A}^{(n-1)} \times_{n+1} \mathbf{A}^{(n+1)} \cdots \times_M \mathbf{A}^{(M)}.$$

2.3 Tensor Decompositions Tensor decompositions factorize/approximate a (large) input tensor into a smaller tensor associated a set of factor matrices (one for each mode). Many tensor decompositions have been proposed (see [24] for a detailed survey), among which CANDECOMP/PARAFAC (CP) [8, 16] and the Tucker decomposition [35] are most popular. Here we focus on the Tucker decomposition.

Definition 2.5. (Tucker Decomposition) *Let* $\mathcal{X}$ *be a tensor of size* $I_1 \times \cdots \times I_M$. *A Tucker decomposition of* $\mathcal{X}$ *yields a core tensor* $\mathcal{G}$ *of specified size* $R_1 \times \cdots \times R_M$ *and factor matrices* $\mathbf{U}^{(n)}$ *of size* $I_n \times R_n$ *for* $n = 1, \ldots, N$ *such that*

$$\mathcal{X} \approx \mathcal{G} \times \{\mathbf{U}\}. \tag{2.1}$$

The Tucker decomposition approximates a tensor as a smaller core tensor (i.e., a compressed version of the original tensor) times the product of matrices that span appropriate subspaces in each mode. The approximation tries to minimize the total squared error $\|\mathcal{X} - \mathcal{G} \times \{\mathbf{U}\}\|$. Typically, the factor matrices $\mathbf{U}^{(n)}$ are assumed to be orthonormal.

3 Data Model

In this section, we formally introduce the data model. The raw information of content-based social networks are typically obtained through communication flows, such as emails in the form of $\langle$sender, recipient, message$\rangle$. We model such data as content-based network tensors, defined as the follows:

Sym.	Definition
$\mathcal{X}$	$I_1 \times \cdots \times I_M$ data tensor
$\mathbf{X}_{(d)}$	mode-d matricization
$\mathcal{G}$	$R_1 \times \cdots \times R_M$ core tensor
$\mathbf{U}^{(d)}$	$I_d \times R_d$ mode-d factor matrix
$\{\mathbf{U}\}$	all factor matrices
$\mathbf{a}_{:r} (\mathbf{a}_{r:})$	r-th column (row) of matrix $\mathbf{A}$
$\mathbf{C}^{(d)}$	dimension clustering (partitioning) on mode-d
K_d	number of clusters along mode-d
$\mathcal{C}_i^{(d)}$	the i-th cluster along mode d, $1 \le i \le K_d$
$\mathcal{I}^{(d)}$	dimension index set along mode d
$\{\mathcal{I}\}$	collection of dimension sets for all modes
$\mathcal{X}(\{\mathcal{I}\})$	sub-tensor induced by $\{\mathcal{I}\} \equiv (\mathcal{I}^{(1)}, \cdots, \mathcal{I}^{(M)})$

Table 1: Definitions of symbols

Definition 3.1. (Content-based network tensor) *A content-based network tensor is a tensor* $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times J_1 \times \cdots \times J_M}$ *with two sets of modes:* N *network modes of dimensionality* $I_1, \ldots, I_N$ *and* M *content modes of dimensionality* $J_1, \ldots, J_M$. *The element* $x_{i_1 i_2 \cdots i_N j_1 j_2 \cdots j_M}$ *in* $\mathcal{X}$ *is the joint weight of the network dimensions* $i_1 i_2 \cdots i_N$ *and content dimensions* $j_1 j_2 \cdots j_M$.

For example, if we represent the message body of an email using the well-established vector space model (i.e., bag-of-words approach), then a data corpus consisting of emails between many users can be modeled as a third order tensor of sender, recipient, and keyword modes. Here sender and recipient are the network modes and keyword is the content mode, as shown in Figure 1. The (i_1,i_2,j) element of the tensor is a variable that indicates whether person i_1 sent an email to person i_2 which contained the keyword j. This variable may be weighted (e.g., similar to the TFIDF score in information retrieval) or simply a binary indicator. Our methods can deal with both; unless otherwise noted, we use the latter.

For storing the tensors, we use the coordinate format as proposed in [6]. By storing only the non-zero entries along with their indices, we need space for just $(M+1)P$ elements, where P is the number of non-zeros and M is the number of modes.

4 Problem Overview

After introducing the data model, we now present the problems addressed in the paper. The goals are

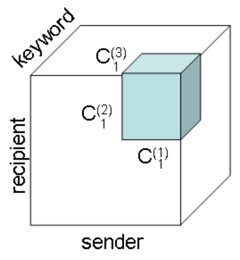


Figure 1: A content-based network tensor with two network modes (sender and recipient) and one content modes (keyword).

- **Content-based network analysis**: identify patterns from large content-based social networks;
- **Context-sensitive graph visualization**: present the patterns in the network and content modes for intuitive data exploration.

These two parts are naturally complementary to each other, as summarized in Figure 5 and further explained in Section 6. In particular, the patterns from network analysis provide a core structure for hierarchical and context-sensitive graph visualization. Likewise, the graph visualization helps better convey the patterns to users through an intuitive presentation.

4.1 Content-based Network Analysis We analyze the content-based network tensors using a two-level process. In this section we provide an overview of the overall process, and introduce some necessary notation. Section 5 explains our methodology in detail.

First, given a content-based network tensor, how to analyze the content and network modes in an efficient and robust manner? In this case, we need to model each mode of the tensor. In this paper, we focus on clustering applications.

Second, once the clusters on all modes are identified, how to efficiently identify the correlations across different modes? In this case, we need to correlate those clusters for cross-mode patterns. For example, in the email tensor, we want to associate groups of people (communities) and groups of keywords (concepts) to find topic-specific community patterns.

Single-Mode clustering The first, crucial step in discovering clusters (i.e., groups of dimensions) in each mode is to apply the Tucker decomposition on the original network tensor $\mathcal{X}$. As explained in Section 2.3, this serves a dual purpose: (i) similar to SVD, it discovers the appropriate coordinates for each mode, that capture the main correlations present in the data, allowing us to examine each mode independently of the others, and (ii) it reduces the number of dimensions we have to deal with, thus significantly speeding up pairwise similarity computations, which are necessary for clustering. Next, we apply a clustering algorithm on each of the factors $\mathbf{U}^{(d)}$.

The end result is a set of clusters (groups of dimensions) on each mode of the tensor.

Definition 4.1. (Dimension cluster, clustering) *A mode-d* dimension cluster $\mathcal{C}_i^{(d)}$ *is a subset of all dimensions in mode-d and a mode-d* dimension clustering $\mathbf{C}^{(d)}$ *is a partition of all dimensions of mode-d into K_d dimension clusters.*

Formally, $\mathbf{C}^{(d)}$ is a label vector that represents a mapping from each dimension i_d, for $1 \leq i_d \leq I_d$, to a cluster label $\mathbf{C}^{(d)}(i_d)$ where $1 \leq \mathbf{C}^{(d)}(i_d) \leq K_d$. The set of all dimensions assigned to the i-th label is $\mathcal{C}_i^{(d)} \equiv \{i_d \mid \mathbf{C}^{(d)}(i_d) = i\}$. The number of clusters in $\mathbf{C}^{(d)}$ is denoted by K_d. The cluster sizes, i.e., number of dimensions in $\mathcal{C}_i^{(d)}$, for $1 \leq i \leq K_d$, are denoted by $|\mathcal{C}_i^{(d)}|$.

Cross-mode clustering First, we introduce notation to describe sub-tensors, which are formed by selecting a subset of dimensions along each mode.

Definition 4.2. (Dimension set and sub-tensor) *A dimension index set is $\mathcal{I}^{(d)}$ is a subset of all indices i_d, for $1 \leq i_d \leq I_d$, along mode d of tensor $\mathcal{X}$. Its cardinality is denoted by $|\mathcal{I}^{(d)}|$. We denote an ordered collection of index sets, one for each mode, by $\{\mathcal{I}\} \equiv (\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(d)}, \ldots, \mathcal{I}^{(M)})$. Given such a collection, the sub-tensor $\mathcal{X}(\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(d)}, \ldots, \mathcal{I}^{(M)}) \equiv \mathcal{X}(\{\mathcal{I}\})$ is the $|\mathcal{I}^{(1)}| \times \cdots \times |\mathcal{I}^{(M)}|$ tensor formed by selecting the corresponding subsets of dimensions along each mode d, for $1 \leq d \leq M$.*

Note that a dimension cluster $\mathcal{C}_i^{(d)}$ as defined above is an dimension index set. A combination of different dimension clusters, one from each mode, defines a cross-mode cluster and its associated sub-tensor.

Definition 4.3. (Cross-mode cluster) *A set of dimension clusters $\{\mathcal{C}\} = (\mathcal{C}_{i_1}^{(1)}, \ldots, \mathcal{C}_{i_M}^{(M)})$ defines a* cross-mode cluster *and its associated sub-tensor $\mathcal{X}(\{\mathcal{C}\})$.*

The shaded area in Figure 1 is a sub-tensor induced by dimension clusters $\mathcal{C}_1^{(1)}$, $\mathcal{C}_1^{(2)}$, and $\mathcal{C}_1^{(3)}$.

4.2 Context-sensitive Graph Visualization After obtaining all the patterns, how to visualize and explore social networks in the context of different network and content clusters? We argue this is as important as the network analysis, since the intuitive and scalable presentation of the patterns is a key to user understanding.

5 Content-based Network Analysis

We now present the network analysis on content-based social network tensors. First, we introduce a high-order dimensionality reduction process using tensor decomposion. Second, we leverage the low-dimensional representation along each mode for clustering. Third, we utilize input tensor to find dense connections across clusters from different modes. Figure 2 shows the overall process using the email tensor.

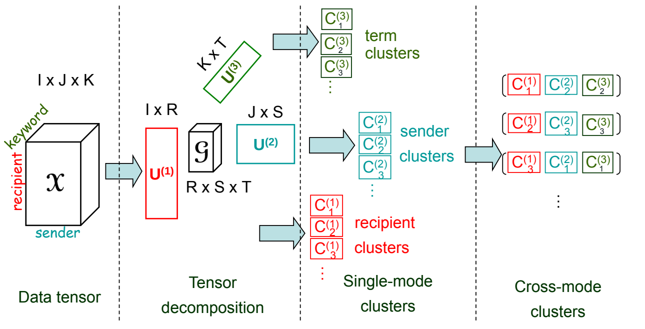


Figure 2: Network analysis through tensor decomposition

5.1 Dimensionality reduction through Tensor decomposition

First, we present how to use tensor decomposition as a dimensionality reduction process. Second, we address how to improve the scalability through biased sampling.

One simple way of analyzing the i-th mode of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is to matricize $\mathcal{X}$ along the i-th mode, i.e., construct the matrix $\mathbf{X}_{(i)} \in \mathbb{R}^{I_i \times (I_1 \times \cdots \times I_{i-1} \times I_{i+1} \times \cdots \times I_N)}$. Then, we model the row vectors of $\mathbf{X}_{(i)}$. As shown in the example of Figure 3,the matricization of $\mathcal{X} \in \mathbb{R}^{I \times J \times 3}$ along the recipient mode gives us I recipient vectors of size $J \times 3$.

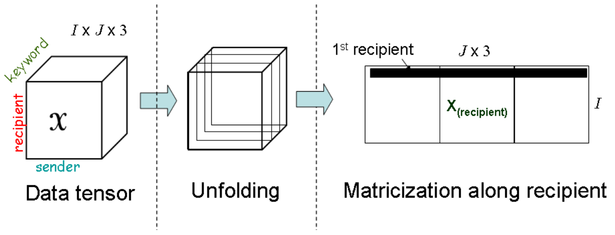


Figure 3: Tensor matricization generates extremely high dimensional vectors (e.g., the dark row vector in $x_{recipient}$). Consequently, direct modeling through the matricization for each mode is expensive and prone to overfitting.

There are two main issues with this simplistic approach:

- *Computationally expensive:* The cost for modeling this extremely high dimensional vectors can be high.
- *Overfitting:* Due to the high dimensionality, the sample size (i.e., the number of row vectors) is small and prone to overfitting.

Tensor Dimensionality Reduction To alleviate these problems, we apply tensor decomposition to reduce the dimensionality of all modes, in order to facilitate robust and efficient cluster analysis. More specifically, we adopt the *Tensor decomposition* with two different algorithms HOSVD and HOOI (Tucker) [1].

After the decomposition, the factor matrices $\mathbf{U}^{(i)} \in \mathbb{R}^{I_i \times R_i}$ for $i = 1, \ldots, M$ provide low-dimensional subspaces on each mode. For example, as shown in Figure 2, the recipient subspace is an $I \times R$ factor matrix.

Compared to the simplistic matricization, the space saving on the i-th mode is in the order of $O(\prod_{n \neq i} I_n / R)$. Depending on the subsequent analysis, the computational saving can also be tremendous.

Biased Sampling When the original tensor is too big to be decomposed entirely, we perform biased sampling on the dimensions based on the marginal norm distribution along each mode. A similar sampling algorithm has been used for Tensor CUR decomposition [12]. The pseudo-code for the sampling is shown in Algorithm 1. After that, we apply a tensor decomposition on the small sub-tensor induced by the sampled dimensions to find the subspaces for each mode.

5.2 Single-mode clustering

The goal here is to cluster the dimensions along a specific mode, i.e., we want to find the cluster labels $\mathbf{C}^{(d)}$ (see Definition 4.1). The overall clustering utilizes both the factor matrices and the core tensor. First, we need to construct low-dimensional points. Then we can apply any clustering algorithm on them[2].

We need to handle each dimension in a different way, depending on whether a specific dimension is used in the tensor decomposition (i.e., is included in the sample) or not. To facilitate presentation, we call the dimensions used in the tensor decomposition *training dimensions* and the rest of the dimensions *testing dimensions.*

For example, if the i-th dimension in mode-d is used in the decomposition, the low-dimensional point is the

[1]Details about tensor algorithms can be found in a recent survey [25].

[2]In the experiments, we use k-means, but other clustering algorithms can also be used.

```
Algorithm 1: TensorSampling(tensor $\mathcal{X} \in \mathbb{R}^{I_1\times\cdots\times I_M}$, sample sizes in $J_d|_{d=1}^{M}$)
   Output: sampled tensor $\mathcal{Y} \in \mathbb{R}^{J_1\times\cdots\times J_M}$
1  for $d = 1$ to $M$ do
       // marginalize along each mode
2      $\mathbf{z} = \mathrm{rownorm}(\mathbf{X}_{(d)})$
3      Select the $J_d$ dimensions with largest entries in $\mathbf{z}$
4  Construct the subtensor $\mathcal{Y}$ induced by the selected dimensions.
5  ________________________________
6  SubAlgorithm rownorm(matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$)
7  for $d = 1$ to $M$ do
       // compute the norm of the $i$-th row
8      $\mathbf{z}(d) = \|\mathbf{a}_{d:}\|$
9  return $Vz$
```

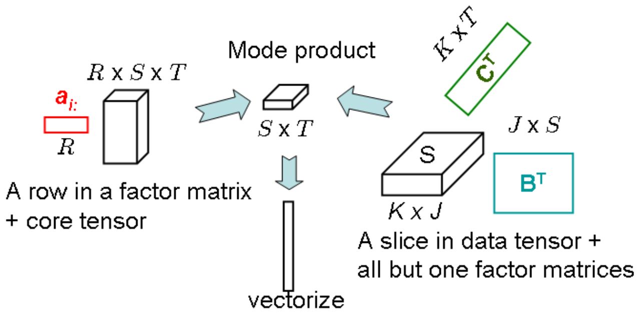
mode product of the core tensor $\mathcal{G}$ and the i-th row of factor matrix $\mathbf{U}^{(d)}$. Formally, the low-dimensional point $\mathbf{z} \in \mathbb{R}^{R_1\cdots R_{d-1}R_{d+1}\cdots R_M}$ is

$$\mathbf{z} = \mathrm{vectorize}(\mathcal{G} \times_d \mathbf{u}_{i:}^{(d)}). \tag{5.2}$$

If the i-th dimension in mode-d is not used in the decomposition, we need to approximate it using all the other factor matrices except the $\mathbf{U}^{(d)}$. Let $\mathcal{S}$ denote the slice of the i-th dimension along mode-d of the tensor $\mathcal{X}$. Then, the low-dimensional point $\mathbf{z} \in \mathbb{R}^{R_1\cdots R_{d-1}R_{d+1}\cdots R_M}$ can be computed as

$$\mathbf{z} = \mathrm{vectorize}(\mathcal{S} \times \{\mathbf{U}^{(j)\mathsf{T}}\}_{j\neq d}). \tag{5.3}$$

A pictorical view of the email tensor is shown in Figure 4.



$\mathbf{a}_{i:}$
R
R x S x T
Mode product
S x T
K x T
C^T
J x S
S
B^T
K x J
A row in a factor matrix + core tensor
A slice in data tensor + all but one factor matrices
vectorize

Figure 4: The constructions of the low-dimensional points for training and testing dimensions

The equivalence of Equation 5.2 and Equation 5.3 for Tucker tensor is shown by the following lemma.

```
Algorithm 2: Single-Mode-Cluster(cluster mode $d$, Data tensor $\mathcal{X}$, factor matrice $\{\mathbf{U}^{()}\}$, core tensor $\mathcal{G}$)
1  foreach dimension $i$ in the $d$-th mode do
2      if dimension $i$ is a training dimension then
           // Mode-product of core and $\mathbf{U}^{(d)}$
3          $\mathbf{z} = \mathrm{vectorize}(\mathcal{G} \times_d \mathbf{u}_{i:}^{(d)})$
4      else
5          Find the slice $\mathcal{S}$ of dimension $i$ along the $d$-th mode
           // Project on all but the $d$-th factor matrics
6          $\mathbf{z} = \mathrm{vectorize}(\mathcal{S} \times \{\mathbf{U}^{(j)\mathsf{T}}\}_{j\neq d})$
       // store the low-dimensional point
7      $\mathbf{z}_{i:} = \mathbf{z}$
8  cluster the rows of $\mathbf{Z}$
9  return cluster assignments
```

Lemma 5.1. *Given a Tucker tensor $\mathcal{X} = \mathcal{G}\times\{\mathbf{U}\}$ where $\mathbf{U}^{(n)}$ is orthogonal for $n = 1, \ldots, M$,*

$$\mathcal{X} \times_1 \mathbf{U}^{(1)\mathsf{T}} \cdots \times_{d-1} \mathbf{U}^{(d-1)\mathsf{T}} \times_{d+1} \mathbf{U}^{(d+1)\mathsf{T}} \cdots \times_M \mathbf{U}^{(M)\mathsf{T}}$$
$$= \mathcal{G} \times_d \mathbf{U}^{(d)}$$

Proof. Follows from the orthogonality of $\mathbf{U}^{(n)}$.

We assume that the data tensor can be precisely represented as the mode products between the core and factor matrices. If $\mathcal{X} \approx \mathcal{G} \times \{\mathbf{U}\}$ (instead of strict equality), the equivalence of Equation 5.2 and Equation 5.3 becomes approximate as well. In terms of computation, Equation 5.2 is much more efficient than Equation 5.3, which implies that constructing low-dimensional points for training dimensions is much easier than for the testing dimensions. Finally, the algorithm is shown in Algorithm 2. The core tensor size is an input parameter, which is typically set to be much smaller than the size of the data tensor.

5.3 Cross-mode clustering After identifying the clusters within each mode, we correlate the clusters across different modes. The goal is to explain a specific cluster by the related clusters from the other modes. For example in the email tensor, a recipient cluster can be explained by the fact that they receive emails from specific sender clusters about specific topics (i.e., clusters of keywords) In other words, a cluster becomes clearer only in the context of related clusters from other modes.

With this idea in mind, we want to find the dense cross-mode clusters (see Definition 4.3). Typically, dense cross-mode clusters indicate high correlations of a subset of dimensions across multiple modes. Or more formally, we want to find the M-tuples $\{\mathcal{C}\} \equiv \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(M)}\}$ that represent the interesting cross-mode clusters. The subtensor induced by the cross-mode cluster $\{\mathcal{C}\}$ is denoted as $\mathcal{X}(\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(M)})$. We may omit a mode $\mathcal{C}^{(d)}$ when all dimensions in mode d are included in the cross-mode cluster.

Given the definition of cross-mode cluster, the goal here is to find the densest cross-mode clusters. The density of $\{\mathcal{C}\}$ is defined as the density of its induced subtensor $\mathcal{X}(\{\mathcal{C}\}) \in \mathbb{R}^{K_1 \times \cdots \times K_M}$, or formally, $\| \mathcal{X}(\{\mathcal{C}\}) \| / \prod_i K_i$. This serves as a simple but effective measure of the "interestingness" of a cross-mode cluster. More specifically, in this paper, we focus on identifying the dense cross-mode clusters defined by a rectangular sub-tensor.

A simple solution is to enumerate all the clusters along each mode, to compute the density for all the cross-mode clusters, and to identify the densest ones. However, this is not efficient because computing the density for a cross-mode cluster is an expensive operation. Without indexing, the worst cost for every such operation is linear with respect to the number of non-zeros in the tensor.

A better way is to leverage the fact that the tensor norm can be incrementally computed. We can sequentially scan over the non-zeros in the tensor; update the corresponding norms of the sub-tensor induced by cross-mode clusters; finally, pick those with the highest density. Note that the temporary norm counters will only be created when needed, in order to avoid unnecessary storage for empty cross-mode clusters.

6 Context-sensitive Graph Visualization

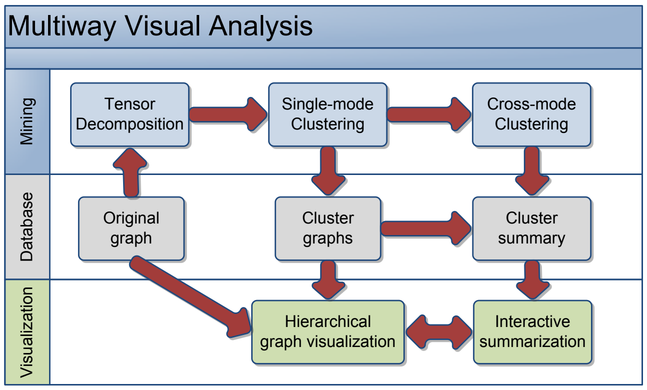


Figure 5: The interaction between mining, data, and visualization layers.

So far, we have focused on how to mine a content-based network for single-mode and cross-mode clusters as shown in the top layer of Figure 5. However, all this information needs to be presented to the users in a meaningful, clear and intuitive ways.

To address that, we now turn on ways to tackle scalability from a cognitive perspective. In particular, we propose to use hierachical graph visualization coupled with single-mode and cross-mode clustering to achieve this goal, as shown in Figure 5. The key observation is that visualization and mining components reinforce each other to facilitate intuitive understanding of the data.

We give an overview of the visual analysis process next (see Figure 5). Finally, we give several examples of visualizations on real data, which clearly demonstrate the power of our visualizations, combined with our proposed content-based analysis.

6.1 Hierarchical Graph Visualization

To maintain the user's mental map, visualizations for huge graphs need to compute a layout, which strikes a balance between the general and the detailed information being displayed. To achieve this, our huge graph visualization method works as follows.

Visualize clusters recursively: Clusters at multiple levels of the graph are generated by using our proposed content-sensitive tensor clustering. More specifically, we apply single-mode clustering recursively on the network dimension (specifically, on the recipient mode) to construct hierarchical clusters, which will be used for visualization.

Both the original graphs and the clusters are stored in a database and are dynamically loaded level by level at runtime. For original graphs, we store both nodes and edges. For clusters, we only store the cluster nodes, while cluster edges are dynamically computed by querying the database. Due to space limitation, the details about database schema and index are not presented in the paper. Keep in mind that the data layer in Figure 5 is designed to support real-time interaction for visualization.

Each cluster is visualized as a rectangle, which contains a subset of key nodes (sub-clusters or leaf nodes) filtered by centrality scores within that cluster. We use adaptive thresholding on each level to determine the appropriate nodes to display. In all visualizations in the paper, we intensionally set the threshold high to fit multiple screenshots in one page. For interactive use, we typically display a lot more nodes (in the order of a hundred at a given level).

We design an energy based graph layout algorithm similar to the Kamada and Kawai method [21] which is carefully tuned to determine an optimal layout for the

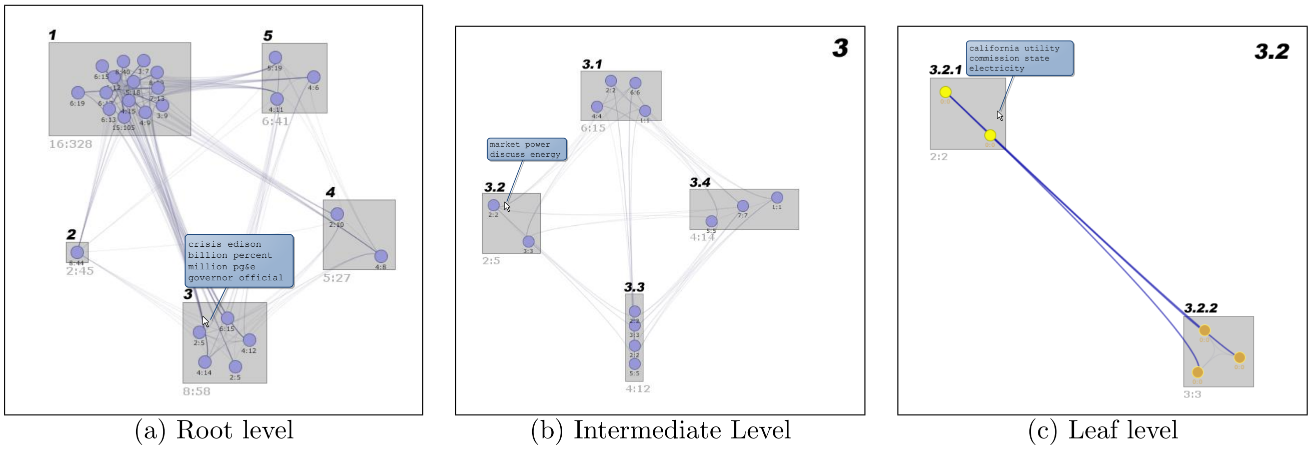


Figure 6: Hierarchical graph visualization for Enron data: It illustrates one navigation path from left to right. The bubbles show the keywords of the content dimension in a dense cross-mode cluster. The light yellow color in (c) indicates the key people clusters and dark yellow indicate the people who directly connect to them.

hierarchical graph at the selected level. The details of the visualization algorithm are beyond the scope of the paper.

Interactive summarization Hierarchical graph visualization does provide a clean layout of the nodes (users and user clusters) for navigation. However, one challenge still remains, i.e., how to guide the navigation on this hierarchical graph. To address that, we need to have intuitive and succinct summaries of the clusters that are displayed in the visualization.

The key observation is that a set of users in the network can be typically described by how they interact with each other within the content modes. That is, the best summary of s network cluster is the corresponding content cluster. For example, a group of users belong to the same cluster because they talk to each other using common keywords. Those keywords will be a good summary for this cluster. To implement this observation, for a network cluster, we show its corresponding dense cross-mode cluster on the content modes.

6.2 Case-study: visual mining on Enron data

Figure 6 plots a navigation path on the Enron email graph. The use case is the following: Starting from the root level (Figure 6a), we display the top-level clusters (gray rectangles) and their key child clusters (purple circles). When the user mouses over a cluster (or a child cluster), the keywords from the corresponding dense cross-mode cluster will pop up. Users can zoom into a cluster (Figure 6b) by clicking on it. At a given level, all the keywords from the ancestor clusters are excluded, since the context is already given.

For example as shown in Figure 6, starting from the root level, the user may traverse through cluster 3 ("crisis,pg&e..."), cluster 3.2 ("market, power...") and cluster 3.2.1 ("california..."). At the leaf level, the user can mouse over a leaf node to show the adjacent network clusters (which are highlighted in Figure 6c). In fact, both networks clusters consist of the key persons who plotted the energy trading strategy causing the California crisis.

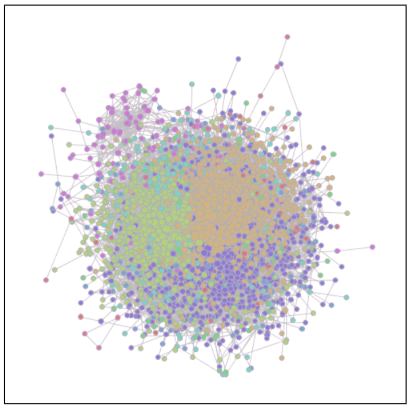


Figure 7: Traditional graph visualization

With the help of both single-mode and cross-mode clusters and the hierarchical graph visualization, we are able to interactively explore data, demonstrating the power of this hybrid approach for facilitating user understanding.

Another benefit is that the hierarchical graph visualization provides a much more intuitive representation of the data, compared to traditional visualization (e.g., Figure 7)

7 Experimental Evaluation

In this section, we evaluate the proposed methods quantitatively using real datasets.

The goal is to answer the following questions:

- How scalable are the proposed methods on different datasets and parameter settings?
- What is the performance break-down on different parts of the analysis?
- How stable are the clusters when we vary the training dimensions?

First, we introduce the datasets and experiment settings. Second, we report the results of various experiments.

We use two real datasets and one synthetic dataset: *SmallBlue* is a 3rd-order (sender, recipient, term) tensor of size $3,679 \times 3,679 \times 1000$ which is constructed based on the social communication data from IBM's social network analysis suite, SmallBlue [14]. To construct the tensor, we take 3,679 users who installed the SmallBlue software and parse the terms by keeping the top-1000 frequent words. The density is 0.0341% (percentage of nonzeros). *Enron* is also a 3rd-order (sender,recipient,term) tensor of size 500×500×500. We parse the Enron corpus [37] by filtering the stopwords and keeping the top-500 users and keywords. The density is 1.7%. *Random* is a set of random tensors with varying density, tensor order, and dimensionality. We consider the following performance metrics: 1) *CPU time* is the elapsed wall-clock time for the computation. This determines how efficiently we can solve the problem. 2) *Distance stability* is the average distance changes (in percentage) of all pairs of dimensions before and after sampling. 3) *Cluster stability* is measured by two metrics: precision and recall with respect to cluster label changes with and without sampling.

7.1 Computational Performance on tensor decomposition

The cost of tensor decomposition depends on the properties of the data tensors. In particular, there are several data-dependent parameters: 1) density (percentage of nonzeros), 2) tensor order, 3) dimensionality of each mode, and 4) core tensor size. We illustrate their impact using synthetic, random tensors, comparing both HOSVD and the Tucker decomposition. Figure 8 illustrates the computational performance when varying different data-dependent parameters. Across all parameter settings, HOSVD is consistently faster than Tucker because Tucker typically requires more than one iterations until convergence. Note that for all experiments, HOSVD and Tucker give very similar approximation accuracy (relative difference less than 5%), so we omitted that for space.

Density: Figure 8(a) shows the CPU time (y-axis) vs. the density (x-axis). Here we vary the density from .001 to 1 percent, while fixing size of the random to $500 \times 500 \times 500$ and the size of the core tensor to $10 \times 10 \times 10$. CPU time for both HOSVD and Tucker increases sub-linearly with density since x-axis increases exponentially, while y-axis increases roughly linearly.

Order: Figure 8(b) shows the CPU time (y-axis) vs. the tensor order (x-axis). We vary tensor order from 3 to 5 by fixing the data tensor dimensionality to 50 along each mode, with overall density .1%, and setting the core tensor size to $10 \times 10 \times 10$. As expected, the CPU time increases super-linearly with respect to the tensor order for both HOSVD and Tucker. This is due to the exponential increase of the tensor sizes (i.e., actual number of non-zeros) with respect to the tensor order.

Dimensionality: Figure 8(c) shows the CPU time (y-axis) vs. the tensor dimensionality (x-axis). We vary tensor dimensionality along each mode from 100 to 1,000, while fixing the order to 3, the density to .1%, and the core tensor size to $10 \times 10 \times 10$. CPU time increases linearly with dimensionality (tensor size).

Core-size: In Figure 8(d), as we increase the result size (core tensor size), the CPU time increases roughly linearly for both HOSVD and Tucker. We vary the core tensor size from $5\times5\times5$ to $25\times25\times25$, while fixing the tensor of size $500\times500\times500$ with the density .001%.

7.2 Scalability and stability on clustering

The main cost of clustering can be divided into two parts: computing the low-dimensional representation and finding the per-mode clusters.

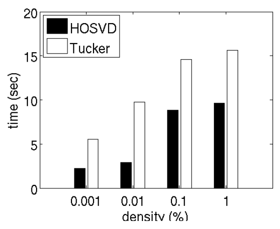
Low-dimensional representation: As shown in Section 5.1, we use tensor decomposition to obtain low-dimensional representation. In order to scale up to large datasets, we use biased sampling to reduce the cost of tensor decomposition. Here we evaluate both stability and speed of this sampling step.

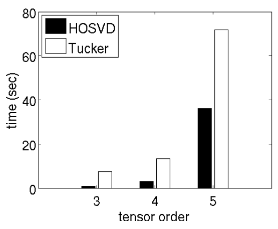
We vary the sampling probability from 5% to 30% on the dimensions of a given mode[3]. We define *distance stability* as

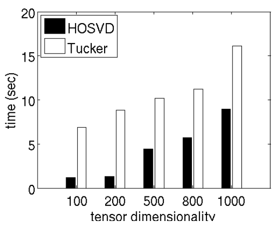
$$1 - \frac{\sum_{\text{pair } i,j} \|d(D(i), D(j)) - d(\tilde{D}(i), \tilde{D}(j))\|}{\sum_{\text{pair } i,j} \|d(D(i), D(j))\|}$$

where $D(i)$ ($\tilde{D}(i)$) is the low-dimensional point for dimension-i without (with) sampling. Intuitively, this measures the relative deviation of the distances over all pairs of dimensions. Ideally, we want to achieve high

[3]We present the results for sampling on the sender mode. We omit the similar results for the other modes.


(a) Density


(b) Tensor order


(c) Dimensionality

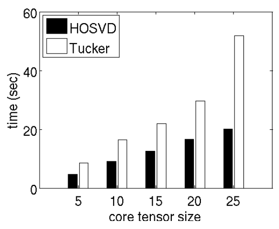
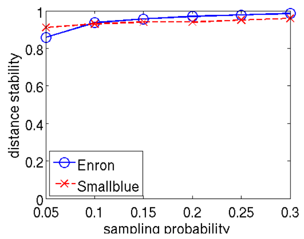

(d) Core size

Figure 8: Computational performance when varying data-dependent parameters. Default settings are: tensor order $M = 3$, dimensionality $n = 500$ per mode, core tensor size 10^3.

stability, because variations in distance will impact the overall clustering results.

Figure 9(a) shows the *distance stability* (y-axis) increases as the sampling probability (x-axis). Note that even with 5% sampling rate, both Smallblue and Enron give very high distance stability. Figure 9(a) shows the *distance stability* (y-axis) increases as the CPU-time (x-axis). Compared to full-tensor decomposition (shown with '*'), sampling can significantly reduce the computational cost without losing much distance stability.


(a) distance stability

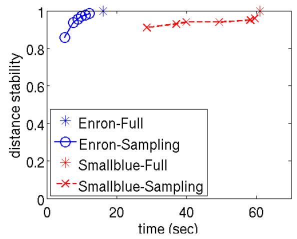

(b) CPU time

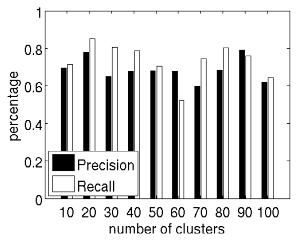
Figure 9: Low-dimensional representation construction: a) distance variation gradually increases as the sample size reduces, b) Sampling reduces the computational time significantly.

Clustering stability
We use the standard K-means clustering algorithm on the low-dimensional representation. Since the running time analysis of K-means is well-known, we omitted the speed evaluation. Also note that the cost of cross-mode clustering is negligible compared to the other parts, since it only requires a single scan over the nonzeros. Therefore, we omit the CPU time evaluation for cross-mode clustering.

Instead, we focus on studying *cluster stability* while varying the sampling probability. In order to scale to large datasets, we have to rely on sampling for efficiently building the low-dimensional representation. Therefore, to quantify how stable the clusters are, we use the standard precision and recall measures, defined as follows:

- *Precision*: Given two dimensions that belong to the same cluster using the sampled tensor, the precision is the probability that they also belong to the same cluster using the full tensor.
- *Recall*: Given two dimensions that belong to the same cluster using the full tensor, the recall is the probability that they also belong to the same cluster using the sampled tensor.

Figure 10 illustrates the precision and recall for both Enron and Smallblue datasets as a function of the number of clusters. Here the sampling probability is 5%. Across all settings on both datasets, precision and recall are high, which confirms that using sampled tensor can significantly reduce computational cost. The overall precision and recall of Smallblue is higher than Enron, because the distance stability of Smallblue is higher, as shown in Figure 9(a).


(a) Enron

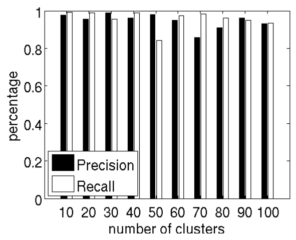

(b) Smallblue

Figure 10: Cluster stability in terms of precision and recall: Both precision and recall are consistently high across different datasets and cluster sizes. The cluster stability is better on Smallblue compared to Enron, because the distance stability of Smallblue is higher than Enron as shown in Figure 9(a). Sampling probability is 5% for both datasets.

8 Related Work

Tensor applications Tucker decompositions have been used in a variety of applications in data mining, and we highlight several distinctive examples here. Savas and Eldén [30] applied the HO-SVD (a version of the Tucker decomposition) to identifying handwritten digits. Acar et al. [2, 3] applied Tucker and other tensor decompositions to the problem of separating conversations in online chatrooms and Chi et al. [9] have applied it to the blogosphere. J.-T. Sun et al. [34] used Tucker for analyzing clickthrough data. J. Sun et al. [33, 32] have written a pair of papers on dynamically updating a Tucker approximation, with applications ranging from text analysis to environmental and network modeling. Kolda and J. Sun [26] introduce a scalable Tucker decomposition for sparse tensors.

Information retrieval and graph analysis In the field of traditional information retrieval (i.e., document-term or user-term relationships), Latent Semantic Indexing (LSI) [29] is based upon the observation that different words do not occur independently within documents. To that end, it models term dependencies using linear correlations and employs the singular value decomposition (SVD) to discover "latent" topics or factors in the document-term matrix.

Conversely, in the context of graphs (i.e., document-document or user-user relationships), similar ideas and decompositions applied to the adjacency (i.e., document-document) matrix, lead to spectral clustering [28] and graph partitioning [22, 10]. Also related are ranking algorithms, such as HITS [23] and PageRank [7]. The former operates on the result set of a traditional IR query, whereas the latter operates on the entire web graph. Both, however, employ similar fundamental mathematical concepts, to rank pages according to some notion of centrality or steady-state random walk probability.

Moving away from methods based on linear algebra, probability mixture models have been employed to analyze content and documents. Cohn and Hofmann [11] introduced an extension of probabilistic latent semantic indexing (PLSI), which tries to model both hyperlink and topic structure in a weh graph. Their method, however, models content associated with nodes (i.e., webpages) and not with source-destination pairs (i.e., edges, such as for example anchor text on hyperlinks). The author-recipient-topic (ART) model [27] employs latent Dirichlet allocation (LDA) to decompose the term distribution into three factors (corresponding to each component of the model's name). Finally, similar approaches have been employed to model information diffusion in social networks [31].

The TOPHITS method [4] is most similar in spirit to our work in this paper. It is based on multi-linear algebra techniques, which generalize the basic tools used in, e.g., LSI and HITS. However, TOPHITS focuses on an entirely different application, namely analyzing the relationships between websites (i.e., sets of webpages within the same domain), leveraging the anchor text of links among them.

Visualization Graph visualization is an active research area in information visualization. One major challenge in graph visualization is graph size [17]. Huge graphs pose several difficult problems. First, if the number of elements is large, high computational complexity tends to be a problem. Second, even if it is possible to place all the elements, the issue of readability or usability arises, because it becomes impossible to discern vertices and edges. Therefore it is important to study graph visualization technology for efficiently browsing and navigating huge graphs.

Over the past decades, much effort has been done for huge graph visualization [17, 15, 36, 18]. A common strategy for visualizing large graphs is to use clustering methods to find groups of related nodes in the large graph, then visualize the graph at multiple abstract levels. The node clustering approach has been taken by a number of graph drawing researchers [15, 36, 13]. Most existing cluster-based methods only use one clustering method, structure-based clustering or content-based clustering. Such methods may fail to reveal the complex relationships from multiple, combined aspects. Edge clustering approaches have also been proposed, such as [19] which relies on node hierarchies provided *a priori*. Finally, the work of [20] also recognizes both aspects of scalability (computational and cognitive) as challenges, and applies METIS recursively to facilitate interactive visualization and exploration of graphs which, however, do not involve any content modes.

In contrast to previous work, in this paper we focus on content-sensitive visualization through clusters from different modes, which provides more intuitive presentation than traditional data mining and graph visualization can provide on their own.

9 Conclusions

In this paper, we combine the power of data mining and information visualization, to provide a comprehensive methodology for the analysis and navigation of huge, content-specific social networks. In particular, we model content-based social network as tensors. We develop an analytic framework for finding single- and cross-mode clusters. Leveraging the cluster results, we present a hierarchical visualization to explore the data, at various levels of detail. We evaluate our methods on both synthetic and real datasets. The former include the popular Enron email dataset, and data from Smallblue, a social network analysis platform for business intelligence applications. The experiments demonstrate both the computational efficiency as well as the intuitive effectiveness of our proposed methodology.

Acknowledgement We are pleased to acknowledge Brett Bader and Tamara Kolda from Sandia National lab for providing the tensor toolbox[5] which makes our implementation and experiments an easy job.

References

[1] `http://www.research.ibm.com/smallblue/`.

[2] E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy, and B. Yener. Modeling and multiway analysis of chatroom tensors. In *ISI 2005*, pages 256–268, 2005.

[3] E. Acar, S. A. Çamtepe, and B. Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *ISI 2006*, pages 213–224, 2006.

[4] B. Bader and T. Kolda. The TOPHITS model for higher-order web link analysis. In *Workshop on Link Analysis*, 2006.

[5] B. W. Bader and T. G. Kolda. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Trans. Math. Softw.*, 32(4):635–653, 2006.

[6] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM J. Sci. Comput.*, 30(1):205–231, 2007.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Com. Net.*, 30(1–7):107–117, 1998.

[8] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. *Psychometrika*, 35:283–319, 1970.

[9] Y. Chi, B. L. Tseng, and J. Tatemura. Eigen-trend: trend analysis in the blogosphere based on singular value decompositions. In *CIKM*, pages 68–77, 2006.

[10] F. Chung. *Spectral Graph Theory.* AMS, 1997.

[11] D. Cohn and T. Hofmann. The missing link: A probabilistic model o fdocument content and hypertext connectivity. In *NIPS*, 2001.

[12] P. Drineas and M. W. Mahoney. A randomized algorithm for a tensor-based generalization of the svd. Technical Report YALEU/DCS/TR-1327, 2005.

[13] P. Eades and Q.-W. Feng. Drawing clustered graphs on an orthogonal grid. In *International Symposium on Graph Drawing*, pages 146–157, 2005.

[14] K. Ehrlich, C.-Y. Lin, and V. Griffiths-Fisher. Searching for experts in the enterprise: Combining text and social network analysis. In *GROUP*, 2007.

[15] Y. Frishman and A. Tal. Navigating clustered graphs using force-directed methods. *Journal of Graph Algorithms Applications*, 4(3):157–181, 2000.

[16] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[17] I. Herman, G. Melancon, and M. Marshall. Graph visualization and navigation in information visualization: a survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January-March 2000.

[18] I. Herman, G. Melancon, and M. Marshall. Multilevel graph layout on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1310–1317, November 2000.

[19] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. In *InfoVis*, 2006.

[20] J. F. R. Jr., H. Tong, A. J. Traina, C. Faloutsos, and J. Leskovec. GMine: A system for scalable, interactive graph visualization and mining. In *VLDB*, 2006.

[21] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, April 1989.

[22] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *J. Par. Distr. Comp.*, 48(1):96–129, 1998.

[23] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *JACM*, 46(5):604–632, 1999.

[24] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. In revision for SIAM Review.

[25] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review.*

[26] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM*, 2008.

[27] A. McCallum, A. Corrada-Emmanuel, and X. Wang. Topic and role discovery in social networks. In *IJCAI*, 2005.

[28] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[29] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. In *PODS*, 1998.

[30] B. Savas and L. Eldén. Handwritten digit classification using higher order singular value decomposition. *Pattern Recogn.*, 40(3):993–1003, 2007.

[31] X. Song, C.-Y. Lin, B. L. Tseng, and M.-T. Sun. Modelling and predicting personal information dissemination behavior. In *KDD*, 2005.

[32] J. Sun, S. Papadimitriou, and P. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *ICDM*, 2006.

[33] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *SIGKDD*, 2006.

[34] J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. CubeSVD: a novel approach to personalized web search. In *WWW*, pages 382–390, 2005.

[35] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3), 1966.

[36] C. Walshaw. A multilevel algorithm for force-directed graph drawing. *Journal of Graph Algorithms Applications*, 7(3):253–285, 2003.

[37] Enron database. `http://bailando.sims.berkeley.edu/enron_email.html`.