SCENT: Scalable Compressed Monitoring of Evolving Multi-Relational Social Networks

YU-RU LIN
Arizona State University
K. SELÇUK CANDAN
Arizona State University
HARI SUNDARAM
Arizona State University
and
LEXING XIE
IBM T.J. Watson Research Center

We propose SCENT, an innovative, scalable spectral analysis framework for internet scale monitoring of multi-relational social media data, encoded in the form of tensor streams. In particular, a significant challenge is to detect key changes in the social media data, which could reflect important events in the real world, sufficiently quickly. Social media data have three challenging characteristics. First, data sizes are enormous – recent technological advances allow hundreds of millions of users to create and share content within online social networks. Second, social data are often multi-faceted (i.e., have many dimensions of potential interest, from the textual content to user metadata). Finally, the data is dynamic – structural changes can occur at multiple time scales and be localized to a subset of users. Consequently, a framework for extracting useful information from social media data needs to scale with data volume, and also with the number and diversity of the facets of the data. In SCENT, we focus on the computational cost of structural change detection in tensor streams. We extend compressed sensing (CS) to tensor data. We show that, through the use of randomized tensor ensembles, SCENT is able to encode the observed tensor streams in the form of compact descriptors. We show that the descriptors allow very fast detection of significant spectral changes in the tensor stream, which also reduce data collection, storage, and processing costs. Experiments over synthetic and real data show that SCENT is faster (17.7x–159x for change detection) and more accurate (above 0.9 F-score) than baseline methods.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; H.5.1 [**Information Interfaces and Representation**]: Multimedia Information Systems—*Evaluation/methodology*

General Terms: Experimentation, Measurement, Algorithms, Human Factors

Additional Key Words and Phrases: Social media, social network analysis, stream mining, multi-relational learning, tensor analysis

Author's address: Y.-R Lin, K. S. Candan and H. Sundaram, Arizona State University, Tempe, AZ 85281. L. Xie, IBM T.J. Watson Research Center, Hawthorne, NY 10532.

1. INTRODUCTION

Large volumes of social media data are being generated through highly popular social media platforms, such as Facebook, Twitter, Digg, and Flickr. For example, as of 2010, Facebook has 500 million users[1], and Twitter currently has a rate of 36,000 tweets per minute[2]. The nature of the data available in these platforms present many opportunities: data about individuals and the social structures to which they belong, are invaluable resources for understanding many multifaceted and complex social phenomena, from entertainment to politics to religion, that permeate every aspect of our daily lives. As many commercial/social/political institutions and movements rush to leverage the Web to improve their reach, online communication takes an increasingly significant role alongside (and in some cases surpassing) more traditional venues. In fact, today social media provide a key context for the rapid emergence and dissemination of cultural memes. Therefore, a critical understanding of the development and evolution of online social structures is increasingly important for educators, policy makers, as well as advertisement agencies and, thus, there is an increasing demand for analysis frameworks that can support applications, such as community discovery, that depend on information *latent in the social data* [Chi et al. 2006; Kolda and Sun 2008; Lin et al. 2008; Lin et al. 2009; Sun et al. 2006].

We note essential computational elements in recognizing nuanced patterns of bridging and linking among individuals and communities that occur through social media at different structural levels of interaction:

—*In social networks, user interactions and community interests are constantly evolving, often tracking real-world events.* Social media data is also multi-faceted: typically involving multiple types of relationships (e.g. friendship, co-commenting on a news story). Entities in social networks may also have different attributes, e.g. location, age, profession. The multi-dimensional and multi-relational nature of these interactions increases the complexity that the computational algorithms need to handle.

—*The datasets relevant to the analysis are enormous in size and diverse in form and content, and are growing and evolving rapidly.* The volume of the data and the speed with which the data changes pose significant challenges. Furthermore, a framework for extracting useful information from social media data needs to scale also against the number and diversity of the facets of the data.

Consequently, scalable frameworks, which are able to analyze voluminous social media data, are important to any significant technical advances in social media understanding. In this paper, we propose SCENT, an innovative, spectral analysis framework for Internet scale monitoring of multi-relational social media data, encoded in the form of tensor streams.

1.1 SCENT: *S*calable *C*ompressed *D*omain Analysis of *E*volvi*N*g *T*ensors

A scalable framework for managing voluminous user data in a form amenable for large scale data analysis is a key step to any significant technical advances in social

[1]http://www.facebook.com/press/info.php?statistics
[2]http://blog.twitter.com/2010/02/measuring-tweets.html

media understanding. The computational ceiling arising due to finite resources requires us to pursue innovative strategies to address the data scalability challenges. To facilitate very large scale data management and analysis, in this paper, we propose *SCENT*, *S*calable *C*ompressed *D*omain Analysis of *E*volvi*N*g *T*ensors framework for monitoring the evolution of multi-faceted (also called multi-relational) social network data resulting from users' continuous online interactions.

The key characteristics of social media data sets of urgent interest include the following: (a) voluminous, (b) multi-relational, (c) evolving, and (d) spectrally regular. As it is common in social media analysis, in SCENT, we model the social data in the form of tensor (multi-dimensional array) streams [Chi et al. 2006; Sun et al. 2006] and consider the problem of tracking the changes in the spectral properties of the tensor over time. Prior work, including [Sun et al. 2006], also considered this problem formulation and attempted to tackle the underlying computational complexity issues through incremental tensor decomposition techniques.

The problem with these existing approaches, however, is that (while being faster than regular tensor decomposition) even *incremental* tensor decomposition has exponential complexity [Sun et al. 2006]. To reduce the computational cost of detecting significant changes in the tensor streams, SCENT introduces an innovative *compressed sensing* mechanism that is able to encode the social data tensor streams in the form of compact descriptors. Compressive sensing (CS) is an emerging area of research in Information theory [Candès and Romberg 2007; Candès and Tao 2006; Candès and Wakin 2008], which shows that under certain conditions a signal can be faithfully reconstructed using fewer number of samples than predicted by the well-known Nyquist sampling theorem. We show that the descriptors allow very fast detection of significant spectral changes in the tensor stream, which also reduce data collection, storage, and processing costs.

While there has been work in other domains, such as audio, on change analysis and onset detection, there are key domain-specific differences between the techniques applicable in different domains. The major differences between traditional multimedia and social media data, especially within the context of change detection, include: (a) at each time instance, the relevant data forms a (multi-relational) graph – as opposed to a vector or matrix of intensity values; (b) the graph is very large. Therefore, change detection in social media data requires techniques that can efficiently detect structural changes in very large graphs.

In this paper, we focus on the problem of *change detection*, a key step in understanding the development and evolution patterns in multi-relational social media data. Our contributions include the following:

(1) *Compressive sensing for tensors*: The first contribution of this paper is to extend the recently developed CS theory to tensor stream analysis. CS theory has been primarily used in the analysis of 1D and 2D continuous time signals. Furthermore, we note that (as we discuss in Section 3) the basic compressed sensing theory assumes the availabilities of (a) a sparse data basis and (b) a constant time random sensing mechanism, neither of which generally holds in social media tensors.

(2) *Compressed tensor encoding*: We show how to create and use *random sensing* ensembles to transform a given tensor into a compressed representation that

implicitly captures the spectral characteristics of the tensor (i.e. the so called core tensor coefficients [Tucker 1966]). The length of this compressed representation is only $O(S \cdot \log N/S)$, where N is the size of data tensor and S is a small approximation rank.

(3) *Efficient change detection*: We also show that if the goal is to only identify the points of significant change (as opposed to discovering the values of the spectral coefficients at each time instance), the random sensing ensembles can be created more cheaply. Moreover, changes can be detected by comparing tensors in their compressed representations with logarithmic space cost.

(4) *Tensor coefficient recovery*: We propose three recovery strategies to incrementally obtain core tensor coefficients either from the input data tensor or from the compressed representation. These strategies can be used in different situations based on the availability of data and resources.

(5) *Systematic evaluation*: We systematically study the efficiency and effectiveness of our proposed framework on both synthetic and real data sets. On synthetic datasets, we study the SCENT performance over different data sizes, dimensionalities, and evolution scenarios. This provides a general recommendation for applications with different computational resources available. On real data sets, we demonstrate the efficiency of SCENT on monitoring time-varying multi-relational social networks. The experimental results show that our SCENT monitoring procedure is able to maintain an approximated tensor stream with high accuracy (above 0.9 in terms of F1-score), low errors (under 1.1 relative to baseline tensor decomposition in real-world datasets) and low time cost (17X–159X faster for change detection).

1.2 Organization

The rest of this paper is organized as follows. Section 2 introduces tensor analysis and the problem of spectral tracking. Section 3 presents the proposed framework, compressed sensing of tensors. Section 4 reports experiments. Section 5 reviews the related work and Section 6 presents our conclusion and future work.

2. SPECTRAL TRACKING FOR THE EVOLUTION OF SOCIAL NETWORKS

We model the social data in the form of tensor (multi-dimensional array) streams. This section provides key notations and minimal background on tensor representation (Section 2.1) and anlysis (Section 2.2). These will allow us to formally state the problem of tracking significant changes occurring in the social media data stream (Section 2.3). For a more comprehensive review on tensors, we refer readers to [Kolda and Bader 2009].

2.1 Tensor Representation of Social Data

A tensor is a mathematical representation of a multi-way array. The *order* of a tensor is the number of modes (or ways). A first-order tensor is a vector, a second-order tensor is a matrix, and a higher-order tensor has three or more modes. We use $\mathbf{x}$ to denote a vector, $\mathbf{X}$ denote a matrix, and $\mathcal{X}$ a tensor. Figure 1 provides an example order-3 tensor representing the three-way relations of users, topics, and keywords. Each entry (i, j, k), for example, could represent the number of times

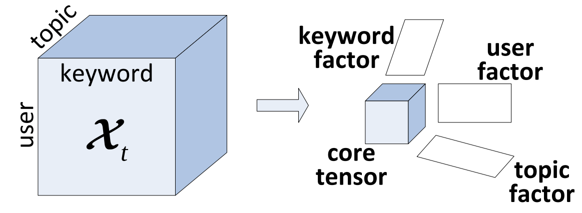

Fig. 1: Social network representation via a 3-mode data tensor $\mathcal{X}_t$ at time t, which can be summarized by a core tensor and factors for each mode (Tucker decomposition).

Table I: Description of notations.

Symbol	Description
$\mathbf{x}$	a vector (boldface lower-case letter)
$\mathbf{X}$	a matrix (boldface capital letter)
$\mathcal{X}$	a tensor (boldface script letter)
$I_1, ..., I_M$	the dimensionality of mode 1, ..., M
$\mathbf{U}_i\|_{i=1}^N, \mathcal{X}_i\|_{i=1}^N$	a sequence of N matrices or tensors
$\|\mathcal{X}\|$	the norm of a tensor $\mathcal{X}$
$\|\mathbf{x}\|_1, \|\mathbf{x}\|_2$	the l_1-norm or l_2-norm of a vector $\mathbf{x}$
Φ, Ψ	basis matrices

the user i submitted an entry on topic j with keyword k. Table I presents the key notations used in this work. Please see Appendix A for more details on tensors.

2.2 Tensor Analysis

Matrix data is often analyzed for its *latent semantics* using a matrix decomposition operation known as the *singular value decomposition* (SVD). The analogous analysis operation on a tensor is known as *tensor decomposition* [Kolda and Bader 2009]. CP/PARAFAC [Carroll and Chang 1970; Harshman 1970] and Tucker decomposition [Tucker 1966] are the two most popular tensor decomposition variants (see [Kolda and Bader 2009] for a detailed review). In this paper, we use the Tucker decomposition to obtain the spectral coefficients (and the basis matrices) of a given tensor.

Definition 2.1 *Tucker decomposition.* A Tucker decomposition of $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_M}$ yields a *core tensor* $\mathcal{Z}$ of specified size $R_1 \times ... \times R_M$ and *factor matrices* $\mathbf{U}_m|_{m=1}^M \in \mathbb{R}^{I_m \times R_m}$ such that

$$\mathcal{X} \approx \mathcal{Z} \prod_{m=1}^{M} \times_m \mathbf{U}_m^T, \tag{1}$$

i.e., the reconstruction error $e = \|\mathcal{X} - \mathcal{Z} \prod_{m=1}^M \times_m \mathbf{U}_m^T\|$ is minimized. The right-hand side denotes a tensor multiplies a sequence of matrices: $\mathcal{Z} \times_1 \mathbf{U}_1 ... \times_M \mathbf{U}_M$ where the symbol $\times_d$ denotes the *mode-d product* (see Definition A.3). The Tucker decomposition approximates a tensor as a smaller core tensor times the product of matrices spanned by the first few left singular vectors in each mode. Typically, the factor matrices $\mathbf{U}_m|_{m=1}^M$ in Tucker decomposition are assumed to be orthogonal (the assumption does not hold for CP decomposition).

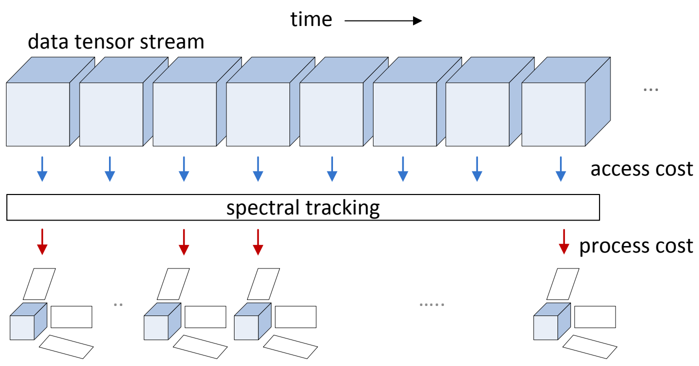

Fig. 2: Problem overview: Our goal is to develop a fast spectral tracking mechanism that reduces the processing cost.

Figure 1 also depicts the Tucker decomposition of the order-3 tensor representing the three-way relations of *users*, *topics*, and *keywords*. In this case, each factor matrix describes one distinct facet of the data: i.e., *users*, *topics*, and *keywords* clusters; the core tensor on the other hand indicates the strength (e.g., amount of correlation) of the relationships among the three facets.

2.3 Spectral Tracking

Figure 2 illustrates a social data stream in tensor form. In this example, the data stream is represented as a sequence of tensors, each representing a snapshot of the social network. Spectral tracking of this stream involves identifying when significant changes occur in the data stream and updating the analysis appropriately. Therefore, the problem of spectral tracking can be formalized as follows.

"Given a sequence of tensors $\mathcal{X}_t|_{t=1}^{T} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$, compute the core tensor and factor matrices, $\mathcal{Z}_t$ and $\mathbf{U}_{m;t}|_{m=1}^{M}$ such that the *mean reconstruction error*

$$e_T = \frac{1}{T}\sum_{t=1}^{T} \|\mathcal{X}_t - \widetilde{\mathcal{X}}_t\|, \tag{2}$$

where $\widetilde{\mathcal{X}}_t \approx \mathcal{Z}_t \prod_{m=1}^{M} \times_m \mathbf{U}_{m;t}^T$, is minimized."

The tracking process involves (1) fetching the data tensors for each time (access cost) and (2) determining whether there is a significant structural change at the time (process cost). Given the reconstructed tensor stream $\widetilde{\mathcal{X}}_t|_{t=1}^{T}$, significant changes in the core tensor are often seen as indicators of the structural (i.e., correlational) changes in the data [Kolda and Sun 2008; Sun et al. 2006]. However, obtaining the core tensor through the decomposition process is expensive. While it is possible to incrementally maintain the decomposition, the time complexity of the incremental maintenance itself is $O(\sum_{i=1}^{M} R_i I_i^2) + O(\sum_{i=1}^{M} I_i \prod_{j=1}^{M} I_j)$; i.e., exponential in the number of modes, or data facets, M. Here R_i denotes the specified rank of the i-th factor matrix, whereas I_i denotes the dimensionality of the data tensor along the i-th mode.

Therefore, a key step in efficient spectral tracking is to minimize the computational cost required in this process.

3. COMPRESSIVE SENSING OF TENSORS

In this paper, we argue that *significant spectral changes can be detected and structural properties can be tracked without having to rely on incremental decomposition maintenance*. We present an efficient spectral tracking framework in this section. After introducing the basic compressed sensing theory (Section 3.1), we present how to extend the theory to tensor analysis (Section 3.2), to develop algorithms for change detection and tensor recovery (Section 3.3 and 3.4).

3.1 Compressive Sensing

Recent developments in information theory have shown that under certain conditions, sparse signals (i.e., signals with few spectral coefficients) can be recovered from a very small number of samples or measurements [Candès and Wakin 2008]. Let $\mathbf{v} \in \mathbb{R}^n$ be an n-dimensional vector (i.e., a signal). Mathematically, sampling (or sensing) of this vector can be described in the form of a matrix product, $\mathbf{y} = \Phi\mathbf{v}$, with a $k \times n$ sensing matrix Φ. The k-dimensional vector $\mathbf{y}$ denotes the k samples one obtains from the data vector $\mathbf{v}$; $\mathbf{v}$ is called *s-sparse* if it has s non-zero entries. CS asserts that an *s-sparse* signal $\mathbf{v}$ can be recovered from $\mathbf{y}$ if the sensing matrix, Φ, satisfies the *restricted isometry property* (RIP); i.e., Φ should be such that

$$(1 - \delta_s)\|\mathbf{u}\|_2^2 \leq \|\Phi\mathbf{u}\|_2^2 \leq (1 + \delta_s)\|\mathbf{u}\|_2^2 \tag{3}$$

holds simultaneously for *all* s-sparse vectors $\mathbf{u}$ and a sufficiently small value δ_s (here $\|\mathbf{u}\|_2 = \sqrt{\sum_i u_i^2}$ is the l_2-norm of vector $\mathbf{u}$). Intuitively, RIP means that all subsets of s columns from Φ are *nearly orthogonal* (but the columns of Φ cannot be exactly orthogonal since there are more columns than rows). This result is especially useful because of the following two observations:

—*Existence of a convenient sensing matrix*: It has been shown that $k \times n$ *random matrices* whose rows are chosen randomly from a suitable distribution satisfying RIP with overwhelming probability, provided that $k \geq C \cdot s \cdot \log(n/s)$, where C is some small positive constant[Candès and Tao 2006; Candès and Wakin 2008]. This implies that it is easy to construct a sensing matrix with small number of samples.

—*Sufficiency of sparsity in an alternative transform space*: The result also holds for a suitably transformed version of $\mathbf{v}$ that is s-sparse. Let Ψ be a transformation matrix such that the coefficient vector $\mathbf{w} = \Psi^{-1}\mathbf{v}$ is s-sparse; then, the sampling process can be formulated as $\mathbf{y} = \Phi\mathbf{v} = \Phi\Psi\mathbf{w}$. The CS theory states that, if Φ is a random matrix satisfying RIP, then in most cases the product $\mathbf{A} = \Phi\Psi$ will also satisfy RIP [Baraniuk et al. 2008]. This implies that most real-world signals (whose coefficients tend to be sparse in some spectral domain) can be efficiently sampled using CS. Note that in CS literature, Ψ often refers to an orthornormal basis (such as a wavelet basis); however, other proper bases can be chosen as long RIP is satisfied, as we shall discuss later.

Furthermore, it has been shown that if we are given a k-dimensional sampling vector $\mathbf{y} = \Phi\mathbf{v}$ (or equivalently $\mathbf{y} = \Phi\Psi\mathbf{w} = \mathbf{A}\mathbf{w}$), the coefficient vector $\mathbf{w}$ can be recovered exactly, with high probability, by solving the minimization problem:

$$\min_{\mathbf{x}} \|\tilde{\mathbf{w}}\|_1 \;\text{ subject to } \mathbf{y} = \mathbf{A}\tilde{\mathbf{w}},$$

where $\|\mathbf{v}\|_1 = \sum_i |v_i|$ is the l_1-norm of $\mathbf{v}$. Here the role of l_1 minimization is to search the sparsest $\tilde{\mathbf{w}}$ that satisfies the constraint $\mathbf{y} = \mathbf{A}\tilde{\mathbf{w}}$, and $\mathbf{w} = \tilde{\mathbf{w}}$ with high probability.

3.2 Sparse Random Sensing Tensors

Suppose Φ is a random sensing matrix satisfying RIP (Equation 3); standard CS involves computing the inner product of data vector with rows of Φ. We extend this to tensors as follows: Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_M}$, we

(1) construct K random tensors $\mathcal{R}_k|_{k=1}^{K}$ of the same size as $\mathcal{X}$, and
(2) obtain a sensing vector $\mathbf{y}$ with each element $\mathbf{y}_k = \langle \mathcal{X}, \mathcal{R}_k \rangle$.

As discussed in Section 3.2, to efficiently encode data $\mathcal{X}$ into a compressed representation $\mathbf{y}$, we need to establish (a) a sparse data basis for $\mathcal{X}$ and (b) a constant time random sensing mechanism.

The above procedure has an equivalent counterpart in matrix form – we construct a $K \times N$ random sensing matrix Φ, where $N = \prod_{m=1}^{M} I_m$. Φ comprises vec($\mathcal{R}_k$) in each row, and then obtain $\mathbf{y} = \Phi\mathbf{x}$, where $\mathbf{x}$=vec($\mathcal{X}$). Here, "vec($\cdot$)" is the vectorization operation which unfolds the tensor into a vector.

In practice, sensing matrices that satisfy the RIP property are quite easy to generate [Candès and Tao 2006]. In our work, we consider a *sparse ensemble* suggested by Baraniuk et al. [Baraniuk et al. 2008] which established that RIP holds with high probability when the $K \times N$ matrix Φ is drawn according to (as long as $K \geq C \cdot s \cdot \log(N/s)$ for all s-sparse vectors, where C is some positive constant):

$$\Phi_{kj} = \sqrt{\frac{3}{n}} \begin{cases} +1 & \text{with probability} & 1/6 \\ \;\;0 & \quad\quad\ldots & 2/3 \\ -1 & \quad\quad\ldots & 1/6 \end{cases} . \tag{4}$$

The advantage of such ensemble is that the sensing matrix Φ (i.e. the set of random tensors $\mathcal{R}_k$'s) will only have about 1/3 non-zero entries.

To construct sparse transformation of data, let us consider a tensor decomposition of tensor $\mathcal{X}$; i.e., $\mathcal{X} \approx \mathcal{Z} \prod_{m=1}^{M} \times_m \mathbf{U}_m^T$ for $\mathcal{Z} \in \mathbb{R}^{R_1 \times \ldots \times R_M}$ and $\mathbf{U}_m|_{m=1}^{M} \in \mathbb{R}^{I_m \times R_m}$. While $\mathcal{Z}$ itself will be dense, the number of coefficient in $\mathcal{Z}$ is much smaller than the size of $\mathcal{X}$; i.e., $\mathcal{Z}$ can be seen as the set of sparse coefficients of $\mathcal{X}$. Note that, the vector $\mathbf{y}$ can be written as

$$\mathbf{y} = \Phi\Psi\mathbf{z}, \tag{5}$$

where $z = \text{vec}(\mathcal{Z})$ and Ψ is a matrix representation of $\prod_{m=1}^{M} \times_m \mathbf{U}_m$. Thus, we can see $\mathbf{y}$ as a sensing vector (measurements) for not only the data tensor $\mathcal{X}$, but also for the core tensor $\mathcal{Z}$ with respect to the transformed matrix (basis) Ψ comprising of a particular set of factor matrices $\mathbf{U}_m|_{m=1}^{M}$. With the following lemma, it is easy to show that RIP (Equation 3) holds for $\Phi\Psi$ and the core tensor $\mathcal{Z}$ can be recovered through compressive sensing.

LEMMA 3.1. *If the factor matrices $\mathbf{U}_m|_{m=1}^{M}$ are orthogonal, the matrix Ψ constructed based on Equation 5 is also orthogonal.*

Table II: SCENT CS-based change detection.

```
Input:
    New data tensor $\mathcal{X}_t$
    Past sensing vector $\mathbf{y}_{t-i}$
    The set of random tensors $\mathcal{R}_k|_{k=1}^{K}$
Output:
    Spectral change at time $t$ relative to time $t-i$
Change-Detection:
1.  Sense $\mathbf{y}_t$ from $\mathbf{y}_{k;t} = \langle \mathcal{X}_t, \mathcal{R}_k \rangle$
2.  Compute $\delta = \|\mathbf{y}_t - \mathbf{y}_{t-i}\| / \|\mathbf{y}_{t-i}\|$. If $\delta > \tau_y$, output change
```

See Appendix B for the proof of Lemma 3.1.

Accordingly, with the sparse transformed representation $\mathcal{Z}$ of data $\mathcal{X}$, the length of the compressed sensing vector $\mathbf{y}$ is only $O(S \cdot \log N/S)$, where N is the size of data tensor and S is a small approximation rank (the size of core tensor). Also see Appendix B for more detailed discussion.

3.3 CS-based Change Detection

Significant changes in the data are reflected in significant changes in the core tensor coefficients [Kolda and Sun 2008; Sun et al. 2006]; thus significant changes in the tensor stream can be said to occur between time instances t and $t-i$ (for some $i \geq 1$) if $\|\Delta\mathcal{Z}\| = \|\mathcal{Z}_t - \mathcal{Z}_{t-i}\| = \|\mathbf{z}_t - \mathbf{z}_{t-i}\| > \tau_z$, for some positive value τ_z. Because $\mathbf{z} = (\Phi\Psi)^{-1}\mathbf{y}$, we can write $\Delta\mathbf{z} = (\Phi\Psi)^{-1}\Delta\mathbf{y} = (\Phi\Psi)^{-1}(\mathbf{y}_t - \mathbf{y}_{t-i})$. This means that, assuming that $\mathbf{P}^{-1} = (\Phi\Psi)^{-1}$ exists, we can detect changes in the core tensor simply by using the sensing vectors $\mathbf{y}_t$ and $\mathbf{y}_{t-i}$. While, in general, $\mathbf{P}^{-1}$ may not exist, the Moore-Penrose pseudo-inverse $\mathbf{P}^{+} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$ is guaranteed to exist[3] and is unique[4]. Thus, we approximate $\Delta\mathbf{z}$ as $\Delta\mathbf{z}^* = \mathbf{P}^{+}(\mathbf{y}_t - \mathbf{y}_{t-i})$. The transformed matrix Ψ can be chosen from the tensor decomposition of an arbitrary data tensor $\mathcal{X}_{t'}$ so that for any $\mathbf{z}_t$ and $\mathbf{z}_{t-i}$, Ψ does not change and hence $\mathbf{P}$ remains constant. Consequently, we have $\|\Delta\mathbf{z}^*\| \propto \|\Delta\mathbf{y}\|$ and the relative change in $\mathbf{y}$ can be computed by $\delta = \|\mathbf{y}_t - \mathbf{y}_{t-i}\| / \|\mathbf{y}_{t-i}\|$. This enables us to develop a compressive sensing based spectral change detection algorithm, as listed in Table II.

Cost. Let L be the number of non-zero entries in data tensors[5] $\mathcal{X}_t$ and $\mathcal{X}_{t-i}$. The computational cost for detecting a spectral change in a given time instance only involves the random sensing process; i.e., K inner products on the non-zero entries in both data and random tensors, which is only $O(KL)$. Note that the set of random tensors used for sensing only need to be constructed *once*; the same set of sensing tensors can be reused for each time instance.

[3]For arbitrary matrices, the existence of the Moore-Penrose pseudo-inverse may be established from the singular value decomposition (SVD). See e.g., [Golub and Van Loan 1996].

[4]If a matrix $\mathbf{A}$ is invertible, its inverse and pseudoinverse are the same; moreover the pseudoinverse of a pseudoinverse is the original matrix.

[5]Tensors that encode social media are known to be relatively sparse [Kolda and Sun 2008]

3.4 Core Tensor Recovery

In this section, we present three complementary core tensor recovery strategies. We focus on the core tensor coefficients as the factor matrices can be obtained from the latest performed tensor decomposition and can be folded together with a core tensor to approximate the original data. For the sampling vector $\mathbf{y} = \Phi\Psi\mathbf{z}$, where Φ is the sensing matrix and Ψ is a matrix representation of the factor matrixes of the data tensor, the core tensor $\mathcal{Z}$ (or $\mathbf{z}$) can be recovered by: (a) tensor decomposition based recovery, (b) factor-driven recovery , and (c) CS-recovery. By leveraging with change detection procedure, these strategies can be used in different situations based on the availability of data and resources.

Tensor decomposition based recovery. As we mentioned earlier, the conventional mechanism to recover the core tensor is to either carry out full tensor decomposition or maintain the decomposition incrementally. *SCENT* can reduce the cost by relying to them only when significant changes are detected in the tensor stream as described in Section 3.3.

Factor-driven recovery. In factor matrix driven recovery, the core tensor is recovered using the data tensor and the factor matrices. The transformation $\mathbf{z} = \Psi^T\mathbf{x}$ which would give the core tensor coefficients from the data values and the matrix representation Ψ of $\prod_{m=1}^{M} \times_m \mathbf{U}_m$ is equivalent to the tensor operation $\mathcal{Z} = \mathcal{X} \prod_{m=1}^{M} \times_m \mathbf{U}_m$. The computational cost of this process is $O(N)$ where $N = \prod_{m=1}^{M} I_m$, but it relies on the availability of the full data tensor as well as the factor matrices. Note that unless a significant change has been detected (in which case tensor will be decomposed), the set of factor matrices can be picked from the most recent decomposition of an earlier data tensor; i.e., Ψ remains unchanged, and tensor decomposition is needed only when significant spectral changes are detected through compressive sensing.

CS-recovery. Let $\mathbf{P} = \Phi\Psi$ be a $K \times S$ matrix[6]. It has been shown that the core tensor coefficient $\mathbf{z}$ can be recovered from constrained l_1-minimization [Candès and Romberg 2007]:

$$\min_{\mathbf{z}} \|\tilde{\mathbf{z}}\|_1 \text{ subject to } \mathbf{y} = \mathbf{P}\tilde{\mathbf{z}}. \tag{6}$$

When the data tensor is large, we have $K \geq S$ and this means that we no longer have an *undersampled* situation given a fixed basis Φ; thus, $\mathbf{z}$ can be recovered unambiguously by l_2-minimization:

$$\min \|\mathbf{y} - \mathbf{P}\tilde{\mathbf{z}}\|_2 \tag{7}$$

and the solution $\mathbf{z}^*$ is obtained from $\mathbf{z}^* = \mathbf{P}^+\mathbf{y}$, where as mentioned earlier, $\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$ is the Moore-Penrose pseudo-inverse. Note that, the computational cost of CS-recovery is dominated by computing $\mathbf{P}$ which is $O(KSN)$ where $K \times S$ is the size of $\mathbf{P}$, N is the data size defined above. While computing $\mathbf{P}$ is relatively expensive, this can be done offline (i.e. when a significant change is detected and the factor matrices are being updated), and the online phase involves computing core tensor coefficient given $\mathbf{P}$.

[6]Here we use a different variable $\mathbf{P}$ and $\mathbf{z}$ instead of $\mathbf{A}$ and $\mathbf{w}$ due to the special construction of $\mathbf{P}$ based on tensor decomposition. See Appendix B for more detailed explanation.

Table III: Core tensor recovery.

```
Encoding: Given data tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_M}$
1.    Construct the random tensors $\mathcal{R}_k|_{k=1}^{K}$ of the same size as $\mathcal{X}$
2.    Sense $\mathbf{y}$ from $\mathbf{y}_k = \langle \mathcal{X}, \mathcal{R}_k \rangle$

Decoding: Given sensing vector $\mathbf{y}$, random tensors $\mathcal{R}_k|_{k=1}^{K}$, factor matrices $\mathbf{U}_m|_{m=1}^{M}$
(a)   Tensor decomposition (given the data tensor $\mathcal{X}$)
      Construct the core tensor by decomposing $\mathcal{X}$
(b)   Factor-driven recovery (given the data tensor $\mathcal{X}$)
      Construct the core tensor from $\mathcal{Z} = \mathcal{X} \prod_{m=1}^{M} \times_m \mathbf{U}_m$
(c)   CS-recovery (given the sensing vector $\mathbf{y}$)
      1. Construct the matrix $\mathbf{P} = \Phi\Psi$
      2. Compute core tensor coefficient $\mathbf{z} = \mathrm{vec}(\mathcal{Z})$ from $\mathbf{z} = \mathbf{P}^{+}\mathbf{y}$
```

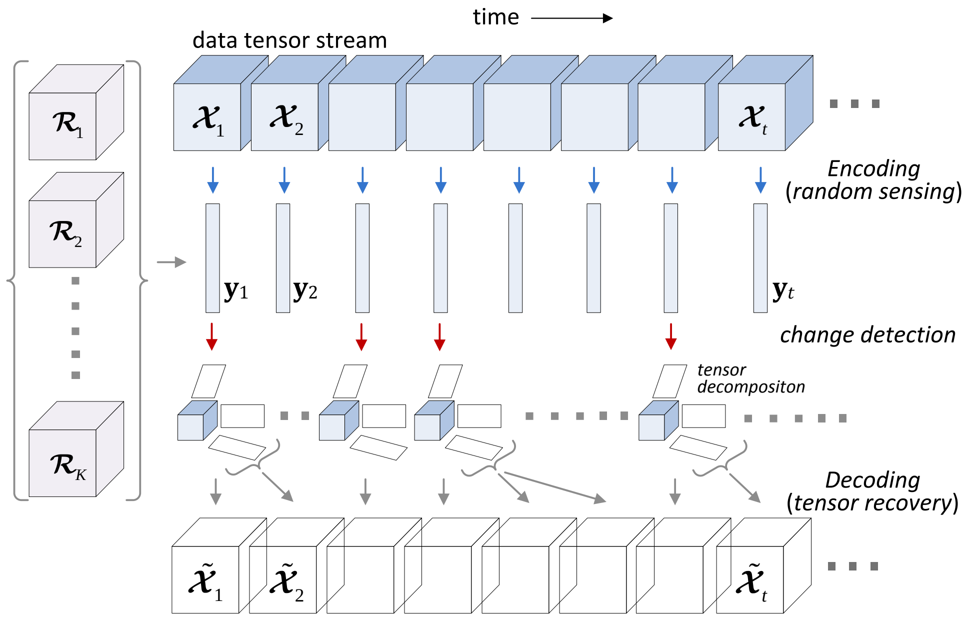

Fig. 3: Our SCENT framework reduces the process cost by triggering tensor decomposition based on change detection.

We summarize these strategies in Table III (also see Appendix C for more details about the three strategies). The overall encoding (sensing) and decoding (recovery) process is illustrated in Figure 3.

4. EXPERIMENTS

We evaluated SCENT on both synthetic data and real-world data sets with the goal of studying the performance in terms of (a) change detection accuracy, (b) recovery error, and (c) cost running time. We consider these under different (synthetic and real) data sets and parameter settings. In the following we first describe the datasets, the baseline methods and the performance metrics used in the evaluations. We then report and discuss the results.

Table IV: Real-world datasets.

Dataset	*Digg*	*DBLP*
dimensions	2575 users $\times$ 51 topics $\times$ 8596 keywords	78 conferences $\times$ 6226 authors $\times$ 6251 keywords
density	3×10^{-6}	1×10^{-6}
timestamps	27 (days)	14 (years)

4.1 Datasets

Digg dataset. From the Digg dataset described in [Lin et al. 2009], we extracted a sequence of third order tensors to represent the user-topic-keyword relation. Each tensor represents the data of one day (from August 1 to August 27, 2008). Each (i, j, k)-entry in the tensor represents that a user i submits a story of topic j with a keyword k. In this dataset, the average number of stories submitted per day is 4076. Consequently, the data density per tensor, i.e. fraction of non-zero entries of a tensor, is about 3×10^{-6}.

DBLP dataset. From the DBLP data, we generate a sequence of conference-author-keyword tensors. Each (i, j, k)-entry in a tensor represents that an author j published a paper with keyword k in conference i. Each tensor represents papers published in a year (from 1995 to 2008). In this dataset, the average number of papers per year is 3180, corresponding to the mean number of non-zero entries of a tensor. The data density per tensor is about 10^{-6}.

The properties of tensors in both real datasets are listed in Table IV. As can be seen, the data tensors in both datasets are relatively sparse.

Synthetic datasets. We design a simulation to generate synthetic tensor streams as input data. The simulation, with ground truth available, helps get a more comprehensive picture about the performance of SCENT in handling diverse structure change scenarios. We generated a set of tensor streams (i.e. sequences of tensors) with characteristics that have been observed in prior research, such as power-low distribution [Barabási et al. 2002] and densification over time [Leskovec et al. 2005]. The simulation is controlled by a set of parameters (see Table VII in Appendix D for the list of parameters). Specifically, we created tensors of different sizes (from 10^3 to 10^6), data densities (from 10^{-4} to 10^{-1}), and number of modes (from 3 to 6), with fixed Poisson parameter $\lambda = 10$ (average timesteps between two changes) and tensor stream length $T = 200$. We used a relatively large drift rate ($\sim 5\%$; data deviation between two change events) and a high change frequency ($\lambda = 10$) to test SCENT in stress conditions. The details of the synthetic data generation are provided in Appendix D.

4.2 Change Detection Methods

We compare the following monitoring procedures, each of which estimates change, Δ, and reports if Δ exceeds a threshold τ.

(1) Naïve-monitor – detects the change based on the difference in consecutive data tensors, i.e. $\Delta = \|\mathcal{X}_t - \mathcal{X}_{t-1}\| / \|\mathcal{X}_{t-1}\|$.

(2) Basic-monitor – detects change based on the difference in reconstructed data tensors across time, i.e., $\Delta = \|\widetilde{\mathcal{X}}_t - \widetilde{\mathcal{X}}_{t-1}\| / \|\widetilde{\mathcal{X}}_{t-1}\|$ where is the approximated

tensor of $\mathcal{X}$ obtained from standard Tucker tensor decomposition.

(3) DTA-monitor – Similar to the basic-monitor except the approximated tensors are obtained by an incremental tensor analysis called DTA [Sun et al. 2006]. At each time t, DTA utilizes the decomposition results of $\mathcal{X}_{t-1}$ to compute $\widetilde{\mathcal{X}}_t$. The forgetting factor in DTA is set to be 0.1 for all the experiments as suggested by the authors[7].

(4) SCENT-monitor – as described in Section 3.3, SCENT detects change based on the difference in the random sensing of data tensors, i.e. $\Delta = \|\mathbf{y}_t - \mathbf{y}_{t-1}\|/\|\mathbf{y}_{t-1}\|$, where $\mathbf{y}$ is the random sensing of tensor $\mathcal{X}$. The length of sensing measurements K is controlled by the sampling constant C and core tensor size. We fixed the approximation rank to be 5 in each mode, for all methods described below.

4.3 Recovery Methods

While DTA-monitor relies on conventional tensor decomposition, the SCENT-monitor uses the following CS-based recovery strategies to maintain the approximated tensor streams:

—SCENT-c (piecewise-constant recovery) – performs full tensor decomposition whenever a change is detected using CS-based approach and holds the decomposition results until the next change point.

—SCENT-f (factor-driven recovery) – performs full tensor decomposition whenever a change is detected, and uses the decomposition factor matrices to perform factor-driven recovery until next change point.

—SCENT-p (CS-recovery through $\mathbf{P}$ matrix) – performs full tensor decomposition whenever a change is detected, and uses the decomposition factor matrices to update the $\mathbf{P}$ matrix and to perform the CS-recovery[8].

4.4 Performance Metrics

We use three metrics to evaluate the efficiency of SCENT:

—*Detection accuracy* (F1): We evaluate change detection accuracy using the F1-score based on precision and recall: $F_1 = 2 \cdot \frac{\text{precision}\cdot\text{recall}}{\text{precition}+\text{recall}}$.

—*Error ratio*: We assess the coefficient recovery performance by comparing the mean reconstruction error (Equation 2) against the mean reconstruction error of the Basic-monitor scheme.

—*Running time* (CPU time in seconds).

The experiments are repeated 10 times under each of the different settings and the average performance results are reported.

[7]When the forgetting factor is set to be 0, DTA outputs results as from standard Tucker tensor decomposition. We have used DTA with forgetting factor 0 to implement Basic-monitor.

[8]Because the offline computation of $\mathbf{P}$ is relatively expensive, in this paper we focus on the online phase of CS-recovery and report its performance on the synthetic datasets.

Table V: Average running time (in seconds) and error ratio. In both datasets, all methods have similar recovery error ratios. However, the time costs (both detection and recovery) of SCENT-monitor are relative low.

Detection	*Digg*	*DBLP*	**Recovery**	*Digg*		*DBLP*	
	time	time		time	error	time	error
Basic	0.810	1.700	Basic	0.810	1	1.700	1
DTA	0.875	1.688	DTA	0.875	1.003	1.688	1.001
SCENT	0.020	0.011	SCENT-c	0.202	1.040	1.040	1.020
			SCENT-f	0.338	1.011	1.011	1.010

Table VI: Summary of performance improvement of SCENT-monitors with respect to the Basic-monitor.

	Detection	**Recovery**	
		SCENT-c	SCENT-f
Gain (speed-up) in Digg	40.4X	299%	144%
Loss (error increase) in Digg	N/A	3.95%	1.97%
Gain (speed-up) in DBLP	159.3X	139%	11.51%
Loss (error increase) in DBLP	N/A	1.09%	0.94%

4.5 Experiments with Digg and DBLP Data

Figure 4 shows the efficiency of SCENT-monitors on Digg and DBLP datasets, compared with baseline methods Basic- and DTA- monitor. We use $C = 1/4$ for SCENT-monitors. Since in these datasets, the ground truth about the abrupt changes is not available, for this experiment we only report the recovery error ratio and the running times. Figure 4 (a) shows the running time for the Digg dataset – the recovery step for SCENT-monitor is required only when changes are detected, i.e. only in 5 of the 27 days: day 3, 11, 16, 17 and 25. Figure 4 (b) shows the error ratio in Digg dataset – the error ratios appear to slightly increase right before change detection; the error ratio of SCENT-f remains low throughout the experiment. Figure 4 (c) shows the running time for the DBLP dataset – the recovery step of SCENT-c is required only when significant changes are detected in '98, '02, '03 and '07. Figure 4 (d) shows the corresponding error ratio – once again error ratios of SCENT schemes remain low (<1.1).

Table V shows the average running times and error ratios. A summarization of performance gain versus loss is given in Table VI. In addition to having low average error ratios in both datasets, the time costs of SCENT-monitors (SCENT-c and SCENT-f) are relatively low. The SCENT-monitors perform 40$\sim$159X faster in change detection and up to 3X faster in recovery, with an error penalty of 1$\sim$4% on average (ref. Table VI). Due to factor-driven recovery between two detected changes, SCENT-f has a lower error ratio than SCENT-c, but a higher running time.

The experiments show that, since tensor decomposition takes significantly more time, a fast change detection mechanism is critical in reducing the overall execution costs.

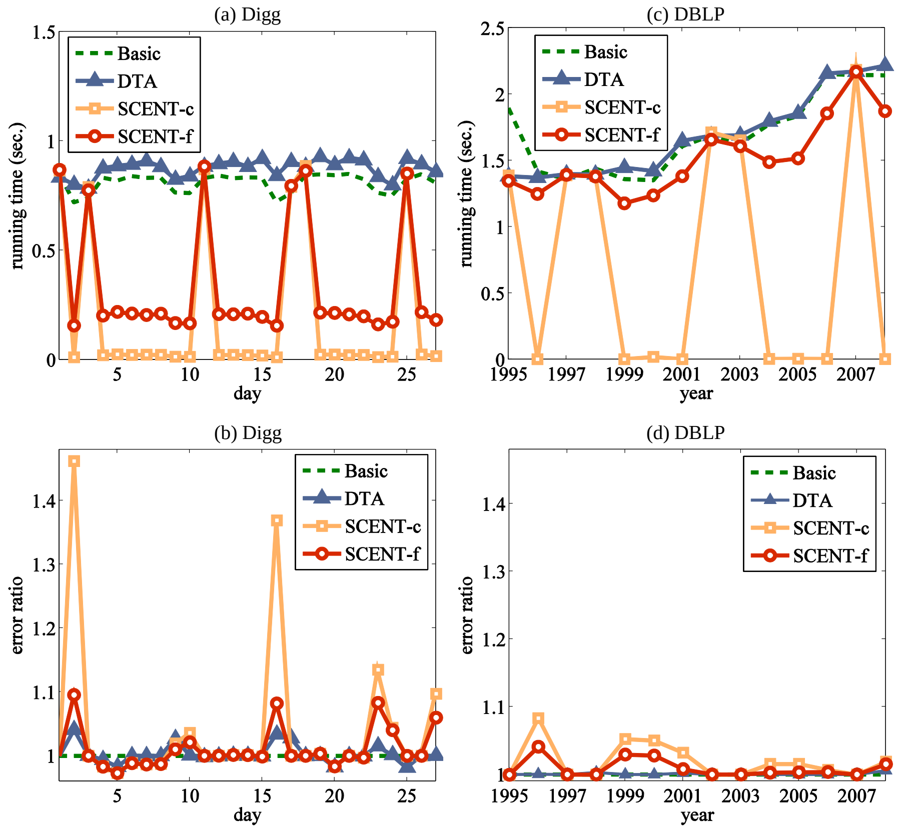

Fig. 4: Efficiency comparison based on Digg and DBLP datasets. (a,c) Running time in Digg and DBLP datasets: The Basic- and DTA- monitor perform full or incremental tensor decomposition at each time frame. For SCENT-c and SCENT-f, full tensor decomposition is required only when changes are detected: see the peaks in the running time curves. For SCENT-c, since there is no maintenance, the time costs between two change points include only the detection costs; for SCENT-f, however, the time includes detection time plus factor-driven recovery time at each instance. Note that there is a significant difference in SCENT-f costs between Digg and DBLP data sets. (b,d) The error ratios of SCENT-monitors remain low for most of the tensors; the errors appear to slightly increase right before change detection. Note: we use a sampling constant $C = 1/4$ for SCENT-monitors. The average performances are reported in Table V and Table VI.

4.6 Synthetic Data Experiments

The goal of the experiments with synthetic data is to observe the detection accuracy based on the ground truth and to observe the performance as a function of key parameters.

As can be seen in Figure 5 (a), in terms of change detection accuracy, despite relying on the whole data set, the Naïve-monitor performs the worse: i.e., structural changes in the social media cannot be estimated directly from the data. The other three methods show comparable performance. Note also that the detection accuracy drops slightly in the case of large densification rates; again, the largest drop is for the Naïve-monitor.

As mentioned earlier, in these experiments we used a relatively large ∼5% drift

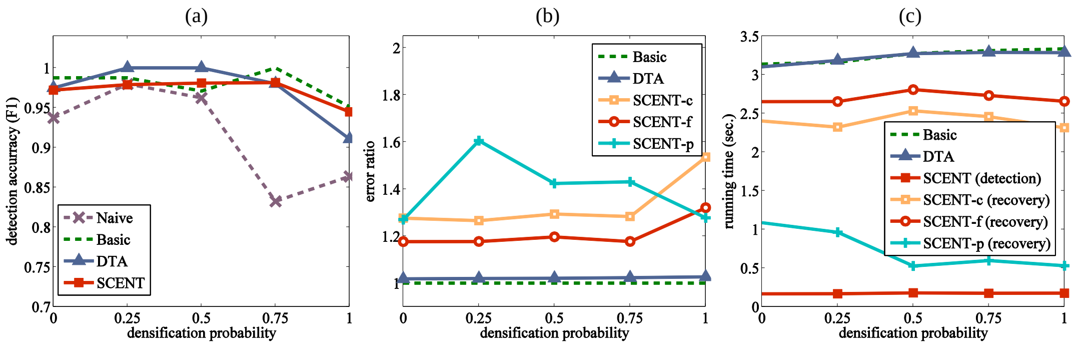

Fig. 5: Efficiency comparisons with baseline methods. (a) detection accuracy (F1-score; higher is better), (b) recovery error ratio (lower is better), and (c) run time (in seconds; lower is better). Note that for baseline methods (Basic- and DTA-monitor), the detection time and recovery time are the same (for Naíve-monitor, the error ratio and running time do not have meaning; hence it does not appear in (b) and (c)); $C = 1/4$ for the SCENT-monitors.

rate and a high frequency $\lambda = 10$ of abrupt changes to observe the recovery error in stress conditions. Figure 5 (b) shows that, the DTA-monitor is the best in terms of the recovery error relative to the Basic monitor. This is expected as both perform similar tensor decompositions. The SCENT alternatives perform well despite the significant drift rate (the error ratio is between 1.2 and 1.5 relative to the Basic-monitor). Figure 5 (c) shows that SCENT-monitor has much lower running time relative to alternatives. For change detection, SCENT is about 17.7X faster than the baseline methods. CS-based processing also helps reduce the recovery time for SCENT-c and SCENT-f by up to $\sim$25% despite the high frequency of abrupt changes: the fact that SCENT-c recovery is much higher than the detection cost indicates that most of the cost comes from the decomposition cost when changes are detected; the fact that SCENT-c and SCENT-f are close to each other indicates that the factor-driven recovery adds only a minimal overhead. Also note that the online phase of the SCENT-p recovery (with $\mathbf{P}$ matrix computed offline) is 68%$\sim$76% faster than the baseline methods.

Scalability. In Figure 6, we study how the computational time of SCENT varies with changes in (a) tensor size (total number of entries of a tensor), (b) tensor density (ratio of non-zero elements in a data tensor), and (c) tensor order (number of tensor modes. The running time increases sub-linearly with the increase of tensor size (ref. Figure 6 (a)). Figure 6 (b) shows that running time increases roughly linearly with tensor density. Note that in both Figure 6 (a) and (b), the x-axes are in log-scale. Figure 6 (c) shows the detection running time remains constant and the recovery running time increases almost linearly with the number of tensor modes. This is important in that the core tensor size, and hence the cost of decomposition, increases exponentially with the number of modes. Our SCENT method reduces the overall process cost tremendously by taking advantage of a fast change detection algorithm. The results are in agreement with Section 3.3.

5. RELATED WORK

Stream mining and time-series analysis. There is abundant literature on stream mining and time-series analysis, such as efficient frequent-item mining based

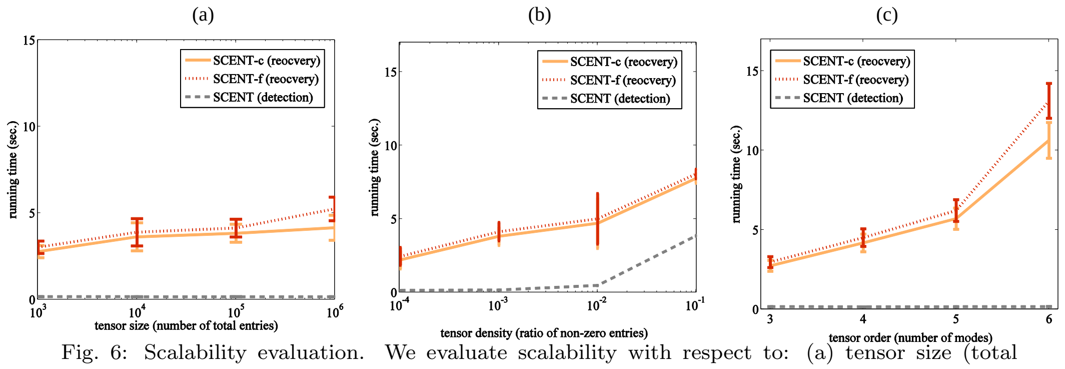

Fig. 6: Scalability evaluation. We evaluate scalability with respect to: (a) tensor size (total number of entries of a tensor), (b) tensor density (ratio of non-zero elements in a data tensor), and (c) tensor order (number of tensor modes. For both detection and recovery, the running time increases sub-linearly with tensor size and roughly linearly with tensor density and order. Note that the x-axes in (a) and (b) are in log-scale.

on statistical synopses and other approximation techniques [Cormode and Hadjieleftheriou 2009], anomaly or novelty detection [Yang et al. 1998; Dasgupta and Forrest 1996], and clustering different types of data streams [Aggarwal and Yu 2005; Aggarwal 2003; Chen and Liu 2009; Aggarwal and Yu 2010]. There has also been work using online or incremental supervised learning on data streams to address the "concept drift" problem [Domingos and Hulten 2000; Wang et al. 2003]. Most of these proposed methods focus on time-series of scalar values, categorical or homogeneous network data as opposed to our multi-relational network data.

Online news and event tracking. The problem of tracking online news or events has been studied as part of the Topic Detection and Tracking (TDT) task [Allan et al. 1998; Kumaran and Allan 2004] which seeks to identify and organize news events in a continuous stream (e.g. a stream of broadcast news stories). With the abundance of text corpora, many of these approaches rely on natural language processing techniques [Brants et al. 2003; Kumaran and Allan 2004], document clustering [Allan et al. 1998], or LDA-based models [Blei et al. 2003; Blei and Lafferty 2006; Wang and McCallum 2006]. The problem has recently attracted considerable research interests due to the extraordinary popularity, growth, and influence of online social media. There has been work on tracking events in blogs [Balasubramanyan et al. 2009; Leskovec et al. 2009]. Besides textual features, there has been work utilizing a variety of context features in social media, such as authors, tags, hyperlinks, times and locations, to improve event mining from social media sources. For example, Becker et al. [Becker et al. 2009] identified events (e.g., President Obama's inauguration) by combining the variety of social media features based on a few domain-specific similarity metrics and a weighted ensemble clustering. Many of these approaches concern offline process of data and disregard the online scalability issues. In complementary, our scalable online framework can be applied to this line of work in that we consider a more general rich setting where keywords are on one of multiple facets and an event is manifested as long as a significant change occurs in the correlations of multiple facets.

Tensor analysis. Tensor decompositions have been used in a variety of applications in data mining. Chi et al. [Chi et al. 2006] applied the high-order singular value decomposition (HOSVD, a version of the Tucker decomposition) to extract dynamic structural changes as trends of the blogosphere. Sun et. al [Sun et al. 2006] proposed methods for dynamically updating a Tucker approximation, with applications ranging from text analysis to network modeling. Although tensors are power tools for representing and mining higher order data, the analysis based on tensor decomposition are usually expensive. To tackle large-size tensor decomposition, Kolda and Sun [Kolda and Sun 2008] have proposed a scalable Tucker decomposition for sparse tensors. A biased sampling method has been proposed in CUR tensor decomposition [Drineas and Mahoney 2007], which reduce tensor dimensions based on the marginal norm distribution along each mode.

Applications of compressed sensing. The desirable features of CS has drawn significant research interests in signal processing, and has been recently extended to many applications, ranged from computer vision to networked data analysis [Haupt et al. 2008]. For example, Bajwa et al. [Bajwa et al. 2006] applies CS to identify sparse channels in order to optimize the wireless communication systems.

Graph mining, compression and sampling. Latent Semantic Indexing (LSI) [Deerwester et al. 1990] employs SVD to discover the latent topics or factors in the document-term matrix. On the Web, methods based on ranking algorithms [Brin and Page 1998; Kleinberg 1999], spectral clustering or graph partitioning [Shi and Malik 2000] and probability mixture model (such as PLSI [Hofmann 1999]) have been proposed for web data reduction. In addition, time-varying graphs have been analyzed through, for example, evolutionary clustering [Lin et al. 2008]. Also, existing works in lossless social graph compression [Chierichetti et al. 2009] are able to reduce the storage complexity to 3-4 bits per edge. The graph sampling work has a different goal in that it seek to reduce the data in a way that preserves statistical and topological properties of the overall network [Leskovec and Faloutsos 2006; Rusmevichientong et al. 2001]. Most algorithms leverage connectivity and the degree of activity as indicators of importance and filter the graphs in a way that maintains only important nodes and edges [Leskovec and Faloutsos 2006].

Many of existing social network analysis and mining approaches do not consider the impact of temporal dynamics on the analysis quality and computational cost. In this paper, we focus on identifying the significant change points when a costly tensor computation would be necessary: Through compressive sensing, our SCENT framework provides a strong indication of the change to the core tensor. In between significant changes, the spectral properties are tracked using relatively cheaper alternatives with good recovery properties.

6. CONCLUSION AND FUTURE WORK

We proposed the SCENT framework for monitoring the evolution of multi-relational social network data from users' continuous online interactions. To track spectral characteristics in multi-relational data, prior work relies on tensor decomposition, which is computationally expensive for internet scale data management and analysis. The key idea in this paper was that significant spectral changes in multi-relational networks can be quickly detected before tensor decomposition, and the fast change detection can further speed up the tensor stream approximation. We leveraged tensor based analysis with the recent established CS theory to develop the SCENT framework, which involved: (a) encoding the observed tensor streams in the form of compact descriptors through sparse random sensing, and (b) incrementally obtaining core tensor coefficients either from the input data tensor or from the compact descriptors. The experimental results showed that our SCENT monitoring procedure was able to maintain an approximated tensor stream with high accuracy (above 0.9 in terms of F1-score), low errors (under 1.5 relative to baseline tensor decomposition) and low time cost (17X–159X faster for change detection).

ACKNOWLEDGMENTS

We thank Jimeng Sun for the providing us with the DTA source code; thank Nan Cao for helping prepare the DBLP dataset; thank Mahsa Mojtahedi for the help with the experiment resources.

REFERENCES

Aggarwal, C. 2003. A framework for diagnosing changes in evolving data streams. In *Proc. of ACM SIGMOD 2003*. ACM, 586.

Aggarwal, C. and Yu, P. 2005. Online analysis of community evolution in data streams. In *Proc. of SDM 2005*. Society for Industrial Mathematics, 56.

Aggarwal, C.C., Z. Y. and Yu, P. 2010. On clustering graph streams. In *Proc. of SDM 2010*.

Allan, J., Papka, R., and Lavrenko, V. 1998. On-line new event detection and tracking. In *Proc. of ACM SIGIR 1998*. ACM, 37–45.

Bajwa, W., Haupt, J., Sayeed, A., and Nowak, R. 2006. Compressive wireless sensing. In *Proc. of the 5th intl. conf. on Information processing in sensor networks*. ACM, 142.

Balasubramanyan, R., Lin, F., Cohen, W., Hurst, M., and Smith, N. 2009. From episodes to sagas: Understanding the news by identifying temporally related story sequences.

Barabási, A., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., and Vicsek, T. 2002. Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications 311,* 3-4, 590–614.

Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M. 2008. A simple proof of the restricted isometry property for random matrices. *Constructive Approximation 28,* 3, 253–263.

Becker, H., Naaman, M., and Gravano, L. 2009. Event identification in social media. In *Proc. of the ACM SIGMOD Workshop on the Web and Databases (WebDB09)*. Citeseer.

Blei, D. and Lafferty, J. 2006. Dynamic topic models. In *Proc. of ICML 2006*. ACM, 120.

Blei, D., Ng, A., and Jordan, M. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research 3*, 993–1022.

Brants, T., Chen, F., and Farahat, A. 2003. A system for new event detection. In *Proc. of ACM SIGIR 2003*. ACM, 330–337.

Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30,* 1-7, 107–117.

Candès, E. and Romberg, J. 2007. Sparsity and incoherence in compressive sampling. *Inverse Problems 23*, 969–985.

Candès, E. and Tao, T. 2006. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory 52,* 12, 5406–5425.

Candès, E. and Wakin, M. 2008. People hearing without listening: An introduction to compressive sampling. *IEEE Signal Processing Magazine 25,* 2, 21–30.

Carroll, J. and Chang, J. 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika 35,* 3, 283–319.

Chen, K. and Liu, L. 2009. He-tree: a framework for detecting changes in clustering structure for categorical data streams. *VLDB J. 18,* 6, 1241–1260.

Chi, Y., Tseng, B., and Tatemura, J. 2006. Eigen-trend: trend analysis in the blogosphere based on singular value decompositions. In *Proc. of the ACM CIKM*. ACM, 68–77.

Chierichetti, F., Kumar, R., Lattanzi, S., Mitzenmacher, M., Panconesi, A., and Raghavan, P. 2009. On compressing social networks. In *Proc. of SIGKDD 2009*. ACM, 219–228.

Cormode, G. and Hadjieleftheriou, M. 2009. Finding the frequent items in streams of data. *Communications of the ACM 52,* 10, 97–105.

Dasgupta, D. and Forrest, S. 1996. Novelty detection in time series data using ideas from immunology. In *Proc. of the Intl. Conf. on Intelligent Systems.* Citeseer, 82–87.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *J. of the American Society for Information Science 41,* 6, 391–407.

Domingos, P. and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 80.

Drineas, P. and Mahoney, M. 2007. A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear algebra and its applications 420,* 2-3, 553–571.

Golub, G. and Van Loan, C. 1996. *Matrix computation.* The Johns Hopkins University Press.

Harshman, R. 1970. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA working papers in phonetics 16,* 1, 84.

Haupt, J., Bajwa, W., Rabbat, M., and Nowak, R. 2008. Compressed sensing for networked data. *IEEE Signal Processing Magazine 25,* 2, 92–101.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *Proc. of ACM SIGIR 1999.* ACM Press New York, NY, USA, 50–57.

Kleinberg, J. M. 1999. Authoritative sources in a hyperlinked environment. *J. ACM 46,* 5, 604–632.

Kolda, T. and Bader, B. 2009. Tensor decompositions and applications. *SIAM Review 51,* 3, 455–500.

Kolda, T. G. and Sun, J. 2008. Scalable tensor decompositions for multi-aspect data mining. In *Proc. of ICDM 2008.* 363–372.

Kumaran, G. and Allan, J. 2004. Text classification and named entities for new event detection. In *Proc. of ACM SIGIR 2004.* ACM, 304.

Leskovec, J., Backstrom, L., and Kleinberg, J. 2009. Meme-tracking and the dynamics of the news cycle. In *Proc. of ACM SIGKDD 2009.* ACM, 497–506.

Leskovec, J. and Faloutsos, C. 2006. Sampling from large graphs. In *Proc. of ACM SIGKDD 2006.* ACM, New York, NY, USA, 631–636.

Leskovec, J., Kleinberg, J., and Faloutsos, C. 2005. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proc. of ACM SIGKDD 2005.*

Lin, Y.-R., Chi, Y., Zhu, S., Sundaram, H., and Tseng, B. L. 2008. Facenet: A framework for analyzing communities and their evolutions in dynamics networks. In *Proc. of WWW 2008.* ACM Press.

Lin, Y.-R., Sun, J., Castro, P., Konuru, R., Sundaram, H., and Kelliher, A. 2009. Metafac: Community discovery via relational hypergraph factorization. In *Proc. of SIGKDD 2009.*

Rusmevichientong, P., Pennock, D., Lawrence, S., and Giles, C. 2001. Methods for sampling pages uniformly from the world wide web. In *AAAI Fall Symposium on Using Uncertainty Within Computation.* 121–128.

Shi, J. and Malik, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22,* 8, 888–905.

Sun, J., Tao, D., and Faloutsos, C. 2006. Beyond streams and graphs: dynamic tensor analysis. *Proc. of ACM SIGKDD 2006*, 374–383.

Tucker, L. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika 31,* 3, 279–311.

Wang, H., Fan, W., Yu, P., and Han, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proc. of ACM SIGKDD 2003.* ACM, 226–235.

Wang, X. and McCallum, A. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proc. of ACM SIGKDD 2006.* ACM, 433.

Yang, Y., Pierce, T., and Carbonell, J. 1998. A study of retrospective and on-line event detection. In *Proc. of ACM SIGIR 1998.* ACM, 28–36.

APPENDIX

A. TENSORS

This sections provide the key notations and main concepts used in this paper. For a more comprehensive discussions on tensors, we refer readers to Kolda and Bader's review [Kolda and Bader 2009].

A tensor is a mathematical representation of a multi-way array. The *order* of a tensor is the number of modes (or ways). We use $\mathbf{x}$ to denote a vector, $\mathbf{X}$ denote a matrix, and $\mathcal{X}$ a tensor. A sequence of tensors $\mathcal{X}_1$ ... $\mathcal{X}_n$ is called a *tensor stream*, if n is a positive integer that increases with time. The *dimensionality* of a mode is the number of elements (entities) in that mode. We use I_d to denote the dimensionality of mode d. E.g., the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ has 3 modes with dimensionalities of I_1, I_2 and I_3, respectively. The (i_1,i_2,i_3)-element of a third-order tensor is denoted by $\mathcal{X}_{123}$. Indices typically range from 1 to their capital version, e.g. i_1=1,...,I_1. Specifically for a matrix $\mathbf{X}$, we use $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ to extract the i-th row and the j-th column of $\mathbf{X}$. We provide the definitions of *tensor norm*, *matricization or unfolding*, and *tensor product* in Appendix A.

Definition A.1 *Tensor norm.* The norm of an M-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_M}$ is $\|\mathcal{X}\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} ... \sum_{i_M=1}^{I_M} x^2_{i_1 i_2 ... i_M}}$.

Definition A.2 *Mode-d matricization or unfolding.* Matricization is the process of reordering the elements of an M-way array into a matrix. The mode-d matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_M}$ is denoted by $\mathcal{X}_{(d)}$, i.e. unfold$(\mathcal{X}, d) = \mathcal{X}_{(d)} \in \mathbb{R}^{I_d \times \prod_{q \in \{1,...,M\}, q \neq d} I_q}$. Unfolding a tensor on mode d results in a matrix with height I_d and its width is the product of dimensionalities of all other modes. The inverse operation is denoted as $\mathcal{X}$ = fold$(\mathcal{X}_{(d)}) \in \mathbb{R}^{I_1 \times ... \times I_M}$. Similarly, we can define a *vectorization* operation $\mathbf{x}$=vec$(\mathcal{X})$, which linearizes (unfolds) the tensor into a vector.

Definition A.3 *Mode-d product.* The mode-d matrix product of a tensor $\mathcal{X}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_d}$ is denoted by $\mathcal{X} \times_d \mathbf{U}$ and results in a tensor of size $I_1 \times ... \times I_{d-1} \times J \times I_{d+1} \times ... \times I_M$. Elementwise, we have $(\mathcal{X} \times_d \mathbf{U})_{i_1 i_2 ... i_{d-1} j i_{d+1} ... i_M} = \sum_{i_d=1}^{I_d} x_{i_1 i_2 ... i_M} u_{j i_d}$. In general, a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_M}$ can multiply a sequence of matrices $\mathbf{U}_i|_{i=1}^{M} \in \mathbb{R}^{I_i \times R_i}$ as: $\mathcal{X} \times_1 \mathbf{U}_1 ... \times_M \mathbf{U}_M$, which can be written as $\mathcal{X} \prod_{i=1}^{M} \times_i \mathbf{U}_i$ for clarity.

B. PROOF AND DISCUSSION

Proof of Lemma 3.1. In Equation 5. Ψ is a matrix representation of $\prod_{m=1}^{M} \times_m \mathbf{U}_m$. Specifically, let j be the linear index corresponding to a set of subscript values $(j_1, ..., j_M)$, and define a tensor $\mathcal{B}_j$ such that

$$\mathcal{B}_j = \mathcal{B}_{j_1 ... j_M} = \prod_{m=1}^{M} \times_m \mathbf{U}_m(:, j_m), \; 1 \leq j_m \leq R_m,$$

where $\mathbf{U}_m(:, j_m)$ is the j_m-th column of $\mathbf{U}_m$. Ψ is constructed such that $\Psi(:, j) =$ vec$(\mathcal{B}_j)$ $\forall 1 \leq j \leq S$, where $S = \prod_{m=1}^{M} \times_m R_m$. Let $\mathbf{u}^m_{i_m} = \mathbf{U}_m(:, i_m)$, $\mathbf{u}^m_{j_m} = \mathbf{U}_m(:, j_m)$, $\forall m \in [1, M]$, be two set of vectors. We have $\mathcal{B}_i = \mathcal{B}_{i_1 ... i_M} = \mathbf{u}^1_{i_1} \otimes \mathbf{u}^2_{i_2} \otimes ... \otimes \mathbf{u}^M_{i_M}$

and $\mathcal{B}_j = \mathcal{B}_{j_1 \ldots j_M} = \mathbf{u}^1_{j_1} \otimes \mathbf{u}^2_{j_2} \otimes \ldots \otimes \mathbf{u}^M_{j_M}$, where $\otimes$ denotes the tensor product operation (or outer product).

The inner product $\langle \mathcal{B}_i, \mathcal{B}_j \rangle = \langle \mathbf{u}^1_{i_1} \otimes \mathbf{u}^2_{i_2} \otimes \ldots \otimes \mathbf{u}^M_{i_M}, \mathbf{u}^1_{j_1} \otimes \mathbf{u}^2_{j_2} \otimes \ldots \otimes \mathbf{u}^M_{j_M} \rangle = \langle \mathbf{u}^1_{i_1}, \mathbf{u}^1_{j_1} \rangle \langle \mathbf{u}^2_{i_2}, \mathbf{u}^2_{j_2} \rangle \ldots \langle \mathbf{u}^M_{i_M}, \mathbf{u}^M_{j_M} \rangle$. Hence, $\langle \mathcal{B}_i, \mathcal{B}_j \rangle = 1$ when $i = j$ and 0 if $i \neq j$. Since $\Psi(:, j) = \text{vec}(\mathcal{B}_j)\ \forall 1 \leq j \leq S$, the columns of Ψ are mutually orthogonal. □

With Lemma 3.1, we can establish that RIP (Equation 3) holds for the pair $\Phi\Psi$ due to the *incoherence* [Candès and Romberg 2007] between Φ and Ψ, where Φ here is constructed by the sparse ensemble method and satisfies RIP [Baraniuk et al. 2008]. The verification of RIP in Equation 5 plays a crucial role in our method. In CS literature, the formulation of sampling process $\mathbf{y} = \Phi\mathbf{v} = \Phi\Psi\mathbf{x}$ usually considers the transformation matrix Ψ as an $n \times n$ orthonormal basis, where n is the length of signal. Consequently, the transformed vector $\mathbf{x} \in \mathbb{R}^n$ is required to be s-sparse (where $s < n$). In such case, the at most s nonzero entries in $\mathbf{x}$ can be effectively recovered by l_1-minimization [Candès and Romberg 2007] (as described in Equation 6). In our method, the transformation matrix $\Psi \in \mathbb{R}^{N \times S}$ (i.e. the matrix representation of $\prod_{m=1}^{M} \times_m \mathbf{U}_m$) is not an orthonormal basis and it directly transforms the data into a relatively small vector $\mathbf{z} \in \mathbb{R}^S$ (the vector representation of the core tensor $\mathcal{Z}$). Hence we need to verify that RIP holds for $\Phi\Psi$ in Equation 5 in particular to assure that $\prod_{m=1}^{M} \times_m \mathbf{U}_m$ is a proper basis and $\mathbf{z}$ is a concise representation in this basis ($\mathbf{z}$ is not necessarily sparse).

The proper transformation leads to two main advantages in our method. First, in the sensing step (Section 3.2 and 3.3), the length of the sensing vector $\mathbf{y} \in \mathbb{R}^K$, is determined by the choice of the $K \times N$ sensing matrix Φ (i.e. the set of random tensors $\mathcal{R}_k$'s). When Φ is constructed by the sparse ensemble method [Baraniuk et al. 2008], RIP holds for the random sensing matrix Φ as long as $K \geq C \cdot s \cdot \log(N/s)$ for all s-sparse vectors, where C is some positive constant. Based on Lemma 3.1, we can verify that RIP holds for the S-length vector $\mathbf{z}$ as long as $K \geq C \cdot S \cdot \log(N/S)$ random tensors are used. Hence, the length of the compressed sensing vector $\mathbf{y}$ is only $O(S \cdot \log N/S)$. Second, in the recovering process (Section 3.4), we show that when K is chosen to be larger than S in the sensing step, $\mathbf{z}$ can be recovered unambiguously by l_2-minimization (Equation 7), which is a more effective recovering process than l_1-minimization.

C. RECOVERY STRATEGIES

The core tensor recovery strategies described in Section 3.4 can be illustrated by Figure 7. By leveraging with change detection procedure, these strategies, corresponding to different ways the random sensing ensembles have been constructed, can be used in different situations based on the availability of data and resources.

D. EXPERIMENT DETAILS

Running time in Digg and DBLP data. In Figure 8, we study how the computational time of the three variants of our SCENT method vary with changes in data density. The original data density (number of non-zero elements in data tensors) in both real-world datasets can be found in Table IV. In these experiments, we randomly remove $10\% \sim 90\%$ elements from the original data tensors and report the running time of SCENT. As can be seen, the running time remains almost

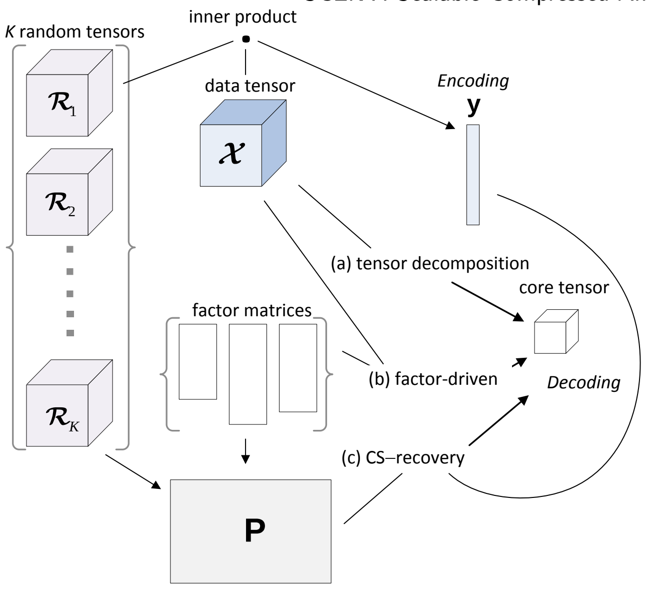

Fig. 7: Recovery strategies: (a) tensor decomposition based, (b) factor-driven, and (c) CS-recovery.

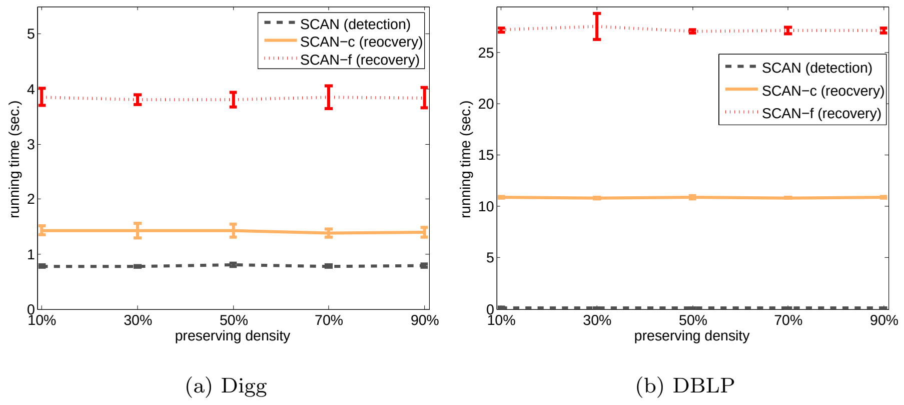

Fig. 8: Running time against data density. The running time remains roughly constant when randomly removing 10% ∼ 90% elements from the original data tensors in both real-world datasets.

constant on both datasets. This observation is also consistent with the scalability test in the synthetic data experiments (see Figure 6(b)) which shows that when the data density increases exponentially, the running time increases roughly linearly.

Length of sensing vector. Figure 9 shows the impact of the parameter, C, on SCENT in the synthetic data experiments: a larger value of C leads to higher accuracy in detection and lower error ratio, with only slightly increase in the detection time. The empirical results suggest that both the detection and recovery quality are reasonably well as long as $C \geq 1/4$.

Synthetic data settings. In the synthetic data experiments (ref. Section 4.1),

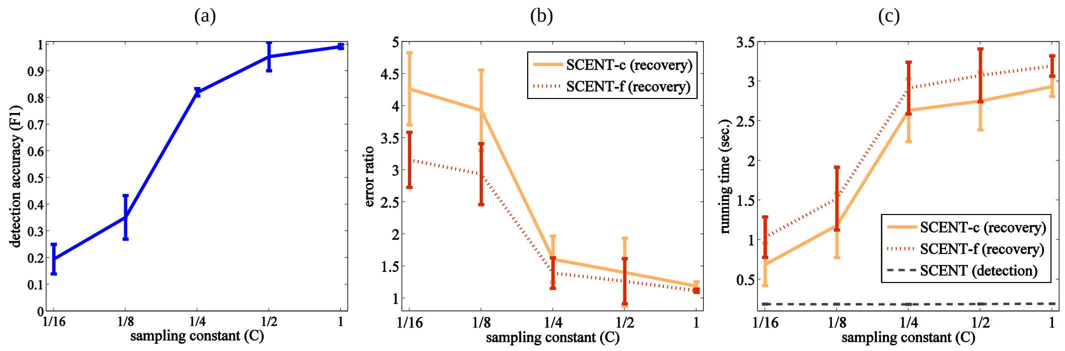

Fig. 9: (a) Detection accuracy (F1-score), (b) error ratio, and (c) detection running time (sec.) over different values of sampling constant C. The performance of SCENT increases with larger value of C (higher accuracy and lower error ratio), with slightly increase in running time.

Table VII: Synthetic data settings.

Fixed settings:
▷ tensor stream length (T): 200
▷ drift rate: $\sim 5\%$ (data deviation between two change events)
▷ change frequency (λ): 10 (average timesteps between two changes)
Varying settings:
▷ tensor size, density, and number of modes
▷ densification rate (κ): with probability κ, a non-zero entries is relocated through preferential attachment

we generated a set of tensor streams (i.e. sequences of tensors) with characteristics that have been observed in prior research. The simulation is controlled by a set of parameters listed in Table VII.

Data drift in Synthetic datasets. The synthetic data are generated as follows: each tensor stream $\mathcal{X}_t|_{t=1}^{T}$ is generated to simulate (a) *abrupt changes* and (b) *drifts*. Abrupt change events are generated based on Poisson distribution with parameter λ, the expected length of interval between changes. We model an abrupt change at t by generating a random tensor $\mathcal{X}_t$, where the non-zero entries are distributed at random according to a specified data density (i.e. the ratio of the number of non-zero entries to the total number of entries in a tensor). Drifts are small changes between consecutive tensors. Since it has been observed that social network evolution exhibits preferential attachment phenomena (see e.g. [Barabási et al. 2002]) and the network tends to densify over time [Leskovec et al. 2005], we model drifts through a *densification parameter*, κ – with probability κ, a non-zero entry is relocated through preferential attachment and with probability $1 - \kappa$, it is moved to a random location. The relocation process proceeds until the tensor norm deviates from the original at a specified rate.

We model data drift to simulate the preferential attachment phenomenon. Prior work on preferential attachment considers a unipartite or bipartite network. To implement the idea on multi-relational networks, we relocate a non-zero entry in a tensor based on the degree distribution along each mode of the tensor. The parameter, densification probability κ, controls to what extent the data drift is

Table VIII: Data drift model.

```
Given: data tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$, densification probability $\kappa$, drift rate $\delta$,
generate data drift $\mathcal{H}$ by the following process:
  $\mathcal{H} := \mathcal{X}$
  Repeat while $\|\mathcal{H} - \mathcal{X}\| / \|\mathcal{X}\| < \delta$
    Randomly pick a non-zero entry $x$ from $\mathcal{H}$
    Randomly pick a mode $m \in [1, M]$
    With probability $\kappa$, pick a zero entry $x'$ from $\mathcal{H}$ along mode-$m$ with
      probability proportional to
      $\frac{\sum_{i_1 \cdots i_{m-1} i_{m+1} \cdots i_M} h_{i_1 \cdots i_{m-1} i_m i_{m+1} \cdots i_M}}{\sum_{i_1 \cdots i_M} h_{i_1 \cdots i_M}}$
      and then swap the value of $x$ and $x'$
    With probability $1 - \kappa$, randomly pick a zero entry $x'$ from $\mathcal{H}$ along mode-$m$,
      and then swap the value of $x$ and $x'$
```

driven by preferential attachment. Another parameter, drift rate δ, controls how much the new data tensor deviates from the old one. The data drift model is given in Table VIII.

Received September 2010; revised March 2011; accepted Month Year