Believe It Today or Tomorrow? Detecting Untrustworthy Information from Dynamic Multi-Source Data

Houping Xiao* Yaliang Li* Jing Gao* Fei Wang† Liang Ge‡ Wei Fan§
Long H. Vu¶ Deepak S. Turaga¶

Abstract

A vast ocean of data is collected every day, and numerous applications call for the extraction of actionable insights from data. One important task is to detect untrustworthy information because such information usually indicates critical, unusual, or suspicious activities. In this paper, we study the important problem of detecting untrustworthy information from a novel perspective of correlating and comparing multiple sources that describe the same set of items. Different from existing work, we recognize the importance of time dimension in modeling the commonalities among multiple sources. We represent dynamic multi-source data as tensors and develop a joint non-negative tensor factorization approach to capture the common patterns across sources. We then conduct a comparison between source input and common patterns to identify inconsistencies as an indicator of untrustworthiness. An incremental factorization approach is developed to improve the computational efficiency on dynamically arriving data. We also propose a method to handle data sparseness. Experiments are conducted on hotel rating, network traffic flow, and weather forecast data that are collected from multiple sources. Results demonstrate the advantages of the proposed approach in detecting inconsistent and untrustworthy information.

1 Introduction

With the emergence of social media and massively deployed mobile devices, people can freely share their information or opinions anytime, anywhere, and about anything. For example, how about the service of a hotel, is the movie exciting, or in which restaurant the food is delicious? Such user-generated contents are helpful for other people. However, unavoidably some information may be misleading or untrustworthy. Therefore, it is extremely important to detect and eliminate such wrong information.

Multi-source Untrustworthy Information Detection. Without any supervision or label data, it is difficult to distinguish trustworthy and untrustworthy information in an unsupervised manner. In this paper, we propose to detect untrustworthy information from the novel perspective of multi-source comparison. This is motivated by the following observation in real life: People will check multiple platforms or information sources to obtain a better idea about certain services. For example, people may investigate from Orbitz, Priceline, and TripAdivsor to book a hotel. The information from one single source could contain noisy, unreliable, or even wrong information, but by comparing information from multiple sources, we can learn which piece of information is more likely to be trustworthy. A piece of information shared by many sources is more likely to be true while a highly inconsistent or conflicting piece could be suspicious. Thus, an effective solution to the task of untrustworthiness detection is through multi-source comparison for outlying behavior or patterns that are inconsistent across sources.

The key challenge for multi-source untrustworthiness detection is that information from multiple sources is not directly comparable. One reason is that the users who create the information are different across multiple sources. For example, when we consider each review platform as an information source, each user may only express his opinion on one platform. However, different sources may possess similar groups and group behavior. For example, users may form certain groups based on their travel preference, and each group may give similar ratings at all platforms. If a hotel obtains unusually high or low ratings from a certain group at one particular platform compared with other platforms, we might want to investigate the user-generated information of this hotel for signs of spam or other suspicious activities. Therefore, we developed techniques to group users and compare group-level user behavior across sources to detect inconsistency as potential untrustworthy information [9, 10].

Importance of Time Dimension. The limitation of existing approaches is that they only consider static data rather than dynamic data. However, multi-source data can continuously arrive with constantly changing distributions. In the multi-platform review example, the user ratings about items change over time as the hotel quality may evolve. After a hotel conducts major renovation, we can expect its rat-

*Department of Computer Science and Engineering, SUNY Buffalo, {houpingx, yaliangl, jing}@buffalo.edu.
†Department of Computer Science and Engineering, University of Connecticut, fei_wang@uconn.edu.
‡Google, lge@google.com.
§Baidu Big Data Lab, fanwei03@baidu.com.
¶IBM T.J. Watson Research Center, {lhvu, turaga}@us.ibm.com.

ings improve and a low rating may be considered anomalous, but during the renovation, users may complain about the noise and give lower ratings, so a high rating may be suspicious. Therefore, user rating patterns evolve over time and we must incorporate the time factor into modeling user group and group behavior to capture untrustworthiness accurately.

One may divide data into snapshots and conduct separate modeling on each snapshot for trustworthiness analysis, but this simple approach fails to consider the temporal connections. Some patterns can only be detected when modeling data along time dimension, for example, there may be seasonal patterns in the hotel quality and corresponding user ratings when the number of tourists changes at different seasons. Therefore, user groups and group-level behavior must be analyzed across both platforms and time dimensions for an effective comparison to detect untrustworthy information.

Summary of the Proposed Approach. In this paper, we propose a novel framework to detect inconsistent information from multi-source dynamic data. The basic idea is to represent each data source as a tensor (user, item, time) and conduct joint tensor factorization to extract the common subspaces shared across sources. The tensor factorization results in a projection from multiple sources onto the common subspaces so that the multi-source information can be compared to detect inconsistencies. Specifically, a core tensor is extracted from each source to represent user group behavior across time clusters and apply consensus core tensor to represent consistent behavior across sources. We then quantify the inconsistency degree based on the difference between each source's core tensor and the consensus core tensor. We further extend the proposed algorithm to handle streaming data by developing an incremental factorization method and discuss practical solutions to solve missing data challenges.

Applications and Experiments. Throughout the paper, we use the task of detecting untrustworthy information from multi-platform user ratings as an example application. In the experiments, we demonstrate the effectiveness of the proposed approach on hotel rating data collected from Orbitz, Priceline, and TripAdivsor. The same setting applies to other online recommendation systems, such as movie, product or restaurant ratings. Moreover, the intermediate output (consensus core tensor) of the proposed algorithm can not only detect inconsistency but also facilitate the fusion of multiple data sources, which is validated in our experiments on network traffic flow and multi-platform weather forecast tasks. Experimental results demonstrate that the proposed method is able to detect inconsistent behavior in network traffic flow and weather forecast analysis. By comparing with baseline static approaches, we illustrate the importance of incorporating time dimension in untrustworthy information detection. We also show the advantage of the proposed incremental version over offline version in time and space complexity. Beyond the experimented datasets, the proposed framework can be applied to many applications in discovering untrustworthy information which may indicate critical points, events or activities, and thus can benefit the security and safety of the cyber and physical world.

To sum up, the contributions of this paper include:

- We propose the novel concept of multi-source inconsistency detection as an effective way to identify untrustworthy information. We model multi-source data as multiple tensors to recognize the importance of time factor and capture the impact of both source and time on inconsistency detection.
- We propose a framework that group both users and timestamps simultaneously. Based on tensor factorization, offline and incremental version of solutions are presented to fit different application cases. We also propose the technique to deal with sparse data.
- Experiments on both real world applications and synthetic data sets show that the proposed framework can find items which receive untrustworthy information.

2 Methodology

In this section, we give the formal definition of the task, and introduce the proposed factorization based framework.

2.1 Problem Formulation and Framework We discuss the setting based on hotel rating application, but as shown in experiments, it can be applied to other cases. Suppose we have M sources, each of which could be a platform or a domain, to collect rating data. In s-th source, the rating data can be represented as a tensor with three dimensions: user, item, and time. Let $\mathcal{X}^s \in \mathbb{R}^{N_s \times K \times T_s}$ denote the data from s-th source, where N_s is the number of users who give rating information, K is the number of items we are interested in, and T_s is the number of timestamp we collect. $\mathcal{X}^s_{ijk}$ represents the rating information from i-th user for j-th item at time k within s-th source. Note that N_s and T_s are different across sources, while the set of items is always the same. The goal of our task is to conduct information trustworthiness analysis to identify the unreliable ones.

The key challenge for multi-source trustworthiness analysis is that the information from multiple sources is not directly comparable. For instance, users' ratings differ across sources due to sparsity or alignment issues. However, the underlying user groups' rating behavior should be similar. The behavior of group level evolves along time. Besides, timestamps should also be partitioned into groups such that users in the same time group share similar rating distributions over items. Based on non-negative tensor factorization, we proposed a framework to simultaneously cluster user sets and timestamps on all M sources. In this framework, we extract a common subspace represented by a core tensor for each source. Since the core tensors of all sources share the

same dimensions, it is feasible to compare them to find the untrustworthy information among the original tensors. Next, we present our two-stage framework.

Stage I: Joint Tensor Factorization

We assume that the users and timestamps can be partitioned into C groups and D clusters. The idea of this stage is to factorize $\mathcal{X}^s \in \mathbb{R}^{N_s \times K \times T_s}$ in the following way:

$$\mathcal{X}^s \approx \mathcal{G}^s \times_1 U^{s,1} \times_2 U^{s,2} \times_3 U^{s,3}, \tag{2.1}$$

where

- $\mathcal{G}^s \in \mathbb{R}^{C \times K \times D}$ is the core tensor, which represents the latent behavior of each user group on each time cluster for each item. $\mathcal{G}^s_{ijk}$ represents the rating of j-th item from i-th user group at k-th time cluster.
- $U^{s,1} \in \mathbb{R}^{N_s \times C}$ denotes the partition of N_s users into C groups, where $U^{s,1}_{ij} \geq 0$ and $\sum_{j=1}^{C} U^{s,1}_{ij} = 1$. Each entry $U^{s,1}_{ij}$ indicates the probability of i-th user belonging to j-th groups.
- $U^{s,2} \in \mathbb{R}^{K \times K}$ is an identity matrix, denoting the items we are interested in.
- $U^{s,3} \in \mathbb{R}^{T_s \times D}$ denotes the partition of T_s timestamps into D clusters, where $U^{s,3}_{ij} \geq 0$ and $\sum_{j=1}^{D} U^{s,3}_{ij} = 1$. Each entry $U^{s,3}_{ij}$ indicates the probability of i-th timestamp belonging to j-th clusters.
- Tensor multiplication is defined as: for each i, j, k, $\mathcal{X}^s_{ijk} \approx \sum_u \sum_v \sum_w \mathcal{G}^s_{uvw} U^{s,1}_{iu} U^{s,2}_{jv} U^{s,3}_{kw}$.

As $U^{s,l}, l = 1, 2, 3$ are all non-negative, $\mathcal{G}^s$ also becomes non-negative. We conduct a **J**oint **N**on-**N**egative **T**ensor **F**actorization (**JNNTF**) on $\mathcal{X}^s$, $s = 1,2,\cdots,M$, and propose to calculate $\mathcal{G}^s$ and $\{U^{s,l}\}$, where $l = 1, 2, 3$ and $s = 1, 2, \cdots, M$, as solutions to the following optimization problem:

$$\min_{\{\mathcal{G}^s, \{U^{s,i}\}, \mathcal{G}^*\} \geq 0} \sum_{s=1}^{M} \left(\mathcal{L}(\mathcal{X}^s) + \alpha \Omega(\mathcal{G}^s, \mathcal{G}^*) \right). \tag{2.2}$$

Here, each tensor can be factorized into factors $\mathcal{G}_s$ and $\{U^{s,l}\}$. The first part of the objective function, $\mathcal{L}(\mathcal{X}^s) = \|\mathcal{X}^s - \mathcal{G}^s \times \{U^{s,l}\}\|_F^2$, measures the factorization error of each tensor. The second part, $\Omega(\mathcal{G}^s, \mathcal{G}^*) = \|\mathcal{G}^s - \mathcal{G}^*\|_F^2$ where $\mathcal{G}^* = 1/M \sum_{s=1}^{M} \mathcal{G}^s$ that captures the consensus information across multiple sources, is a regularization term proposed to learn the consensus information. α is a regularization parameter. The larger α is, the more penalty is added.

Stage II: Inconsistency Score Calculation

Based on the obtained consensus, we calculate inconsistency score for each item. The higher the inconsistency score is, the more untrustworthy the information about this item might be. Let $S_k \in \mathbb{R}^M$ be the difference vector between $\mathcal{G}^s$ from all sources and the consensus tensor $\mathcal{G}^*$ for k-th item. As majority of the items should be normal, the median of all items' difference vectors can represent platform bias. Thus the Inconsistency Score I_k for each item k is calculated as:

$$I_k = \|S_k - S_{\text{median}}\|^2. \tag{2.3}$$

The higher this score is, the more likely that the item receives inconsistent information from multiple sources.

In the following, we present a solution to Eq. (2.2) based on JNNTF in Section 2.2. In Section 2.3, we propose an incremental version of the solution to handle streaming data. Computation complexity analysis for both offline and incremental algorithms is given in Section 2.4. We further tackle the challenge of missing values in Section 2.5.

2.2 Joint Non-Negative Tensor Factorization The key component of the proposed framework is to solve the optimization problem in Eq. (2.2). In this section, we derive the solution for this problem and present an offline algorithm. We have four sets of variables to update. When fixing three sets of them, the objective function is a quadratic function with respect to the remaining one. Therefore, we can iteratively update $\mathcal{G}^s$ or $\{U^{s,l}\}$. The updating rules are summarized in Algorithm 1, and the specific procedure of deducing these rules is discussed as follows.

Algorithm 1 Offline JNNTF

Input: $\mathcal{X}^i \in \mathbb{R}^{N_s \times K \times T_s}, i = 1, \cdots, M$, C, D, and α.
Output: Inconsistency scores: $I \in \mathbb{R}^K$.

```
Initialize {U^{s,l}},G*, and G^s, s = 1, ··· , M; l = 1, 2, 3;
while convergence criterion is satisfied do
    for s ← 1 to M do
        for l ← 1 to 3 do
            Calculate B^{s,l} based on Eq. (2.4) to (2.6);
            Unfold X^s and B^{s,l} on l-th order;
            Calculate T^{s,l} and Q^{s,l};
            U^{s,l}_{ij} ← U'^{s,l}_{ij} sqrt(Q^{s,l}_{ij} / (U'^{s,l} T^{s,l})_{ij});
        end for
        Construct Ū^s; and vectorize G^s, G* and X^s;
        g^s_q ← g'^s_q sqrt( (Ū^s x^s + g*)_q / ((Ū^s (Ū^s)^T + I) g'^s)_q );
        Update g*;
    end for
end while
Calculate inconsistency score I_k according to Eq. (2.3).
return I_k, for k = 1, ··· , K
```

In the following, we derive the updating rule for $U^{s,l}$. The derivation of updating rules for the other variables follow similar procedure. First, we define the following three tensors for s-th source:

$$\mathcal{B}^{s,1} = \mathcal{G}^s \times_1 I_1 \times_2 U^{s,2} \times_3 U^{s,3}; \tag{2.4}$$

$$\mathcal{B}^{s,2} = \mathcal{G}^s \times_1 U^{s,1} \times_2 I_2 \times_3 U^{s,3}; \tag{2.5}$$

$$\mathcal{B}^{s,3} = \mathcal{G}^s \times_1 U^{s,1} \times_2 U^{s,2} \times_3 I_3, \tag{2.6}$$

where I_1, I_2, and I_3 are identity matrices of order C, K, and D. Then we can rewrite $\mathcal{L}(\widetilde{\mathcal{X}}^s)$ as follows:

$$\begin{aligned}\mathcal{L}(\widetilde{\mathcal{X}}^s) &= \sum_{i,j,k}\left(\widetilde{\mathcal{X}}^s_{ijk}-\mathcal{X}^s_{ijk}\right)^2\\ &= \sum_{i,j,k}\left(\sum_u U^{s,1}_{iu}\sum_{v,w}\mathcal{G}^s_{uvw}U^{s,2}_{jv}U^{s,3}_{kw}-\mathcal{X}^s_{ijk}\right)^2\\ &= \sum_{i,p}\left(\sum_u U^{s,l}_{iu}B^s_{up}-(X^s_{(l)})_{ip}\right)^2\\ &= \|U^{s,l}B^s_{(l)}-X^s_{(l)}\|_F^2. \end{aligned} \tag{2.7}$$

$X^s_{(l)}$ and $B^s_{(l)}$ are two matrices unfolded from $\mathcal{X}^s$ and $\mathcal{B}^s$ on l-th order separately, and $p=(j-1)\times T_s+k$. If we derive the loss term involved in $U^{s,l}$ from Eq. (2.2), we can obtain:

$$\begin{aligned}\mathcal{L}(U^{s,l}) &= \mathcal{L}(\widetilde{\mathcal{X}}^s)\\ &= tr\left(U^{s,l}B^{s,l}_{(l)}(B^{s,l}_{(l)})^T(U^{s,l})^T\right)\\ &\quad -2\alpha tr\left(U^{s,l\,T}X^s_{(l)}(B^{s,l}_{(l)})^T\right).\end{aligned} \tag{2.8}$$

With known inequalities $a\le(a^2+b^2)/2b, \forall a,b>0$ and $z\ge 1+\log z, \forall z>0$, we have the following properties:

$$tr\left(U^{s,l}T^{s,l}(U^{s,l})^T\right)\le\sum_{i,j}\frac{(U'^{s,l}T^{s,l})_{ij}(U^{s,l}_{ij})^2}{(U^{s,l}_{ij})'}, \tag{2.9}$$

$$tr\left((U^{s,l})^TQ^{s,l}\right)\ge\sum_{i,j}U'^{s,l}_{ij}Q^{s,1}_{ij}\left(1+\log\frac{U^{s,l}_{ij}}{U'^{s,l}_{ij}}\right) \tag{2.10}$$

where $T^{s,l}=B^{s,l}_{(l)}\left(B^{s,l}_{(l)}\right)^T$, and $Q^{s,l}=X^s_{(l)}\left(B^{s,l}_{(l)}\right)^T$. Further we have:

$$\begin{aligned}Z(U^{s,l},U'^{s,l}) &= \sum_{i,j}\frac{(U'^{s,l}T^{s,l})_{ij}(U^{s,l}_{ij})^2}{U'^{s,l}_{ij}}\\ &\quad -2\alpha\sum_{i,j}U'^{s,l}_{ij}Q^{s,l}_{ij}(1+\log\frac{U^{s,l}_{ij}}{U'^{s,l}_{ij}}).\end{aligned}$$

$$Z(U^{s,l},U'^{s,l})\ge\mathcal{L}(U^{s,l}),\ Z(U^{s,l},U^{s,l})=\mathcal{L}(U^{s,l}). \tag{2.11}$$

According to the definition of auxiliary function as defined in [8], $Z(U^{s,1},U'^{s,1})$ is an auxiliary function of $\mathcal{L}(U^{s,1})$. The first order derivative of the auxiliary function is:

$$\frac{\partial Z(U^{s,l},U'^{s,l})}{\partial U^{s,l}_{ij}}=2\frac{(U'^{s,l}T^{s,l})_{ij}U^{s,l}_{ij}}{U'^{s,l}_{ij}}-2Q^{s,l}_{ij}\frac{U'^{s,l}_{ij}}{U^{s,l}_{ij}}. \tag{2.12}$$

By setting $\partial Z(U^{s,l},U'^{s,l})/\partial U^{s,l}_{ij}=0$, we can obtain the updating rule for $U^{s,l}$:

$$U^{s,l}_{ij}=U'^{s,l}_{ij}\sqrt{Q^{s,l}_{ij}/(U'^{s,l}T^{s,l})_{ij}}. \tag{2.13}$$

Similarly, the updating rules for $\mathcal{G}^s$ can be derived as:

$$g^s_q\leftarrow g'^s_q\sqrt{\frac{(\bar{U}^s\mathbf{x}^s+\mathbf{g}^*)_q}{((\bar{U}^s(\bar{U}^s)^T+I)\mathbf{g}'^s)_q}}. \tag{2.14}$$

When updating the core tensor $\mathcal{G}^s$, the index q is defined as: $q=i+(j-1)*N_s+(k-1)*N_sK$. $\mathbf{x}^s$ is the vectorized form of $\mathcal{X}^s$ with $\mathbf{x}^s_q=\mathcal{X}^s_{ijk}$. $\mathbf{g}^*$ and $\mathbf{g}'^s$ are the vectorized forms of $\mathcal{G}^*$ and $\mathcal{G}'^s$ separately. $\bar{U}^s=[\bar{\mathbf{u}}^s_1,\bar{\mathbf{u}}^s_2,\cdots,\bar{\mathbf{u}}^s_N]$ with $\bar{\mathbf{u}}^s_q(p)=U^{s,1}_{iu}U^{s,2}_{jv}U^{s,3}_{kw}$ and $p=u+(v-1)*N_s+(w-1)*N_sK$. Here, we show that Algorithm 1 can improve the solution gradually in the following lemma:

LEMMA 2.1. *Following the updating rules in Algorithm 1 (shown in Eq. (2.13) to (2.14)), the objective function in Eq. (2.2) decreases monotonically.*

Proof. At each iteration, we show the Hessian matrix of the auxiliary function with respect to $U^{s,l}$ is convex, which is,

$$\frac{\partial^2 Z(U^{s,l},U'^{s,p})}{\partial U^{s,l}_{ij}\partial U^{s,l}_{kl}}=\delta_{ik}\delta_{jp}H_{ij}, \tag{2.15}$$

where, $H_{ij}=2(U'^{s,l}T^{s,l})_{ij}/U'^{s,l}_{ij}+2Q^{s,l}_{ij}U'^{s,l}_{ij}/(U^{s,l}_{ij})^2$. Therefore, $Z(U^{s,l},U'^{s,l})$ is a convex function with respect to $U^{s,l}$ and updating $U^{s,l}$ (Eq. (2.13)) gives the global minimum of $Z(U^{s,l},U'^{s,l})$ at each iteration step. Namely, $\mathcal{L}(U'^{s,l})=Z(U'^{s,l},U'^{s,l})\ge Z(U^{s,l},U'^{s,l})\ge\mathcal{L}(U^{s,l})$. Then, Eq. (2.13) results in monotonic decrease of $\mathcal{L}(U^{s,l})$.

Similarly, we can prove that Eq. (2.14) also result in a monotonic decrease of $\mathcal{L}(\widetilde{\mathcal{X}}^s)$.

2.3 Incremental JNNTF In streaming data applications, data continuously arrive. Let $\mathcal{X}^{s,t}$ denote the data obtained from the s-th source at time t. In this section, we propose a two-step incremental algorithm shown in Algorithm 2 to process streaming data. Its computational and storage complexity is analyzed in Section 2.4.

At the first step, we try to obtain optimal projection matrices by solving the following optimization function:

$$\min_{\{U^{(s,i,T)}\}}\sum_{s=1}^{M}\|\mathcal{X}^{s,T}-\mathcal{G}^{s,T-1}\times\{U^{(s,l,T)}\}\|_F^2. \tag{2.16}$$

According to the procedure of solving optimization in Section 2.3, it is easy to derive the updating rules for $\{U^{(s,l,T)},\ s=1,\cdots,M,\ l=1,2,3\}$. Suppose that the solutions at time t are $\{U^{(s,l,t)}_o,s=1,\cdots,M,l=1,2,3\}$.

The second step is to update core tensor $\mathcal{G}^{s,T}$ at time T by solving the following optimization problem:

$$\min_{\{\mathcal{G}^{s,T}\}}\sum_{s=1}^{M}\sum_{t=1}^{T}\|\mathcal{X}^{s,t}-\mathcal{G}^{s,T}\times\{U^{(s,t,l)}_o\}\|_F^2+\alpha\|\mathcal{G}^{s,T}-\mathcal{G}^{*(T-1)}\|_F^2. \tag{2.17}$$

Similarly, the following equation holds:

$$\begin{aligned}\mathcal{L}(\mathcal{G}^{s,T}) &= \sum_{t=1}^{T}\|\mathcal{X}^{s,t}-\mathcal{G}^{(T)}_s\times\{U^{(s,i,t)}_o\}\|_F^2+\alpha\|\mathcal{G}^{(T)}_s-\mathcal{G}^{*(T-1)}\|_F^2\\ &= \sum_{t=1}^{T}\|x^{s,t}-(U^{s,t}_o)^Tg^{s,T}\|_F^2+\alpha\|g^{s,T}-g^{*(T-1)}\|_F^2,\end{aligned}$$

where $x^{s,t}$, $g^{s,T}$, and $g^{*(T-1)}$ are vectorized from $\mathcal{X}^{s,t}$, $\mathcal{G}^{s,T}$, and $\mathcal{G}^{*(T-1)}$ separately. $U^{s,t}_o\in\mathbb{R}^{N_sKT_s\times CKD}$ is a matrix built from $\{U^{(s,l,t)}_o,l=1,2,3\}$. Note that,

$$\begin{aligned}\mathcal{L}(\mathcal{G}^{s,T}) \quad \leq \quad & \sum_{t=1}^{T}\sum_{l}\frac{(U_o^{s,t}(U_o^{s,t})^T g'^{s,T})_l (g^{s,T})_l^2}{(g'^{s,T})_l^2} \\ & -\sum_{t=1}^{T}\sum_{l}(U_o^{s,t}X^{s,t})_l (g'^{s,T})_l (1+\log\frac{(g^{s,T})_l}{(g'^{s,T})_l}) \\ & -2\alpha\sum_{l}(g'^{s,T})_l (g^{*(T-1)})_l (1+\log\frac{(g^{s,T})_l}{(g'^{s,T})_l}) \\ & +\alpha\sum_{l}\frac{(g'^{s,T})_l (g^{s,T})_l^2}{(g'^{s,T})_l}. \end{aligned} \tag{2.18}$$

Let $Z(g^{s,T}, g'^{s,T})$ denote the right hand side of the Eq. (2.18). Then, the first order derivative of $Z(g^{s,T}, g'^{s,T})$ with respect to $g^{s,T}$ should be:

$$\begin{aligned}\frac{\partial Z(g^{s,T}, g'^{s,T})}{\partial g^{s,T}} \quad = \quad & \sum_{t=1}^{T}\sum_{l}\frac{(U_o^{s,t}(U_o^{s,t})^T g'^{s,T}_l + \alpha/T g'^{s,T}_l}{g'^{s,T}_l} 2g_l^{s,T} \\ & -2\sum_{l}[(U_o^{s,t}x^{s,t})_l + \alpha/T g_l^{*(T-1)}]\frac{g'^{s,T}_l}{g_l^{s,T}}. \end{aligned}$$

Since $g'^{s,T}_l$ could be any point, we replace it with $g_l^{s,T-1}$. Then, let the right hand side equal to 0; we can obtain the updating rules for $g_l^{s,T}$:

$$g_l^{s,T} \leftarrow g_l^{s,T-1}\xi, \tag{2.19}$$

where $\xi = \sqrt{\frac{\sum_{t=1}^{T}[(U_o^{s,t}x_l^{s,t} + \alpha/T g_l^{*(T-1)}]}{\sum_{t=1}^{T}[(U_o^{s,t}(U_o^{s,t})^T g^{s,T-1})_l + \alpha/T g_l^{s,T-1}]}}$. Assume $V^{s,T} = \sum_{t=1}^{T} U_o^{s,t}x^{s,t}$, and $H^{s,T} = \sum_{t=1}^{T} U_o^{s,t}(U_o^{s,t})^T$. The Eq. (2.19) could be rewritten as:

$$g_l^{s,T} \leftarrow g_l^{s,T-1}\sqrt{\frac{V_l^{s,T} + \alpha g_l^{*(T-1)}}{(H^{s,T}g^{s,T-1})_l + \alpha g_l^{s,T-1}}}. \tag{2.20}$$

Updating Eq. (2.20) implies that we do not need to compute new $V^{s,T}$ and $H^{s,T}$ every time. Instead, we only need to calculate $U_o^{s,T}x^{s,T}$ and $U_o^{s,T}(U_o^{s,T})^T$, and add them to the $V^{s,T-1}$ and $H^{s,T-1}$ obtained at previous timestamps.

2.4 Computational Complexity Analysis In this section, we analyze computational complexity for both offline and incremental algorithms. Assume that N is the maximum number of users among all sources, K is the number of items in each source, C is the number of groups in each source, and D is the number of timestamp group in each source. Assume we have $\mathcal{X}^{s,t} \in \mathbb{R}^{N_s \times K \times T_{s,t}}$ at each time t. We consider both algorithms' behaviors from time 1 to T.

Time Complexity. For offline algorithm, we need to implement Algorithm 1 at each timestamp t. Let the maximum number of steps taken to converge at each timestamp t is T_t. The dominating computation in Algorithm 1 is step 10 constructing $U^{s,t}$ for each source, which has the complexity $\mathcal{O}(NK^2CD\sum_{s,t}\sum_{i=1}^{t}T_{s,i})$. For the incremental algorithm, the dominating computation happens in step 3. This step takes $\mathcal{O}(CK^2DN\sum_{s,t}T_{s,t})$. This implies that the incremental algorithm is more efficient than the offline one.

Algorithm 2 Incremental JNNTF

Input: $\mathcal{X}^{s,T} \in \mathbb{R}^{N_s \times K \times T'_s}$, $\mathcal{G}^{s,T-1}, \mathcal{G}^{*(T-1)} \in \mathbb{R}^{C\times K\times D}$; $V_s^{s,(T-1)} \in \mathbb{R}^{C\times K\times D}$ and $H^{s,T-1} \in \mathbb{R}^{CKD\times CKD}$.
Output: Inconsistency scores: $I \in \mathbb{R}^K$.
1: Solve the optimization problem in Eq. (2.17) using Eqs. (2.13) to obtain $\{U^{s,i,T}, s = 1, \ldots, M, i = 1, 2, 3\}$;
2: **for** $s \leftarrow 1$ to M **do**
3: Construct $U^{s,T} \in \mathbb{R}^{CKD\times N_s KT'_s}$;
4: Vextorize $\mathcal{X}^{s,T}$, $\mathcal{G}^{s,T-1}$, and $\mathcal{G}^{*(T-1)}$ into $X^{s,T}$, $G^{s(T-1)}$, and $G^{*s(T-1)}$ separately;
5: Update $V^{s,T}$: $V^{s,T} = V^{s,T-1} + U^{s,T}X^{s,T}$;
6: Update $H^{s,T}$: $H^{s,T} = H^{s,T-1} + U^{s,T}(U^{s,T})^T$;
7: Update $g_l^{s,T} \leftarrow g_l^{s,T-1}\sqrt{\frac{V_l^{s,T}+\alpha g_l^{*(T-1)}}{(H^{s,T}g^{s,T-1})_l+\alpha g_l^{s,T-1}}}$, then vectorize $G^{s,T}$ into tensor $\mathcal{G}^{s,T}$;
8: **end for**
9: Update $\mathcal{G}^{*(T)} = \frac{1}{M}\sum_{s=1}^{M}\mathcal{G}^{s,T}$;
10: Calculate inconsistency score I_k according to Eq. (2.3).
11: **return** I_k, for $k = 1, \cdots, K$

Storage Complexity. At time T, the storage complexity of Algorithm 1 is $\mathcal{O}(MNK\sum_{s,t}T_{s,t})$. Referred to the input section in Algorithm 2, the storage complexity should be $\mathcal{O}(MNK\sum_{s=1}^{M}T_{s,T})$. Since our framework is proposed to handle large datasets, the number of users' and timestamps' clusters are much smaller than the number of users and timestamps. The storage complexity of Algorithm 2 is $\mathcal{O}(MNK\sum_{s=1}^{M}T_{s,T})$ that is less than that of Algorithm 1.

2.5 Handling Missing Data In many real world applications, the input tensor is hardly complete. There are usually many missing entries. To tackle this challenge, we propose an objective function on the available data entries to handle sparse tensors. Assume we have triple-elements set $\mathcal{K}^s = \{(i,j,k) : \mathcal{X}^s_{ijk} \text{ is available}\}$ as input. Therefore, the optimization problem (Eq. (2.2)) should be rewritten as:

$$\begin{aligned}\min_{\{\mathcal{G}^s, \{U^{s,l}\}, \mathcal{G}^*\}\geq 0} & \sum_{s=1}^{M}\sum_{(i,j,k)\in\mathcal{K}^s}\left(\mathcal{X}^s_{ijk} - (\mathcal{G}^s \times \{U^{s,l}\})_{ijk}\right)^2 \\ & +\alpha\sum_{(i,j,k)\in\mathcal{K}^s}\left(\mathcal{G}^s_{ijk} - \mathcal{G}^*_{ijk}\right)^2. \end{aligned} \tag{2.21}$$

Updating $U^{s,l}$. We first unfold $\mathcal{X}^s$ into $X^s_{(l)}$ on i-th dimension where $l = 1, 2, 3$, and assume the binary set $\mathbf{K}^s = \{(p,k) : (X^s_{(l)})_{pk} \text{ is available}\}$ exist. Then we have:

$$\min_{U^{s,l}\geq 0}\sum_{(p,k)\in\mathbf{K}^s}\left((U^{s,l}B^s_{(l)})_{pk} - (X^s_{(l)})_{pk}\right)^2. \tag{2.22}$$

The procedure of obtaining $U^{s,l}$ is similar to Section 2.2. Let's denote $\mathbf{B}^{s,l}_{(l)}$ as the corresponding matrix for $B^{s,l}_{(l)}$, where k-th column is deleted if $(p,k) \notin \mathbf{K}^s$. Updating rules are:

$$U^{s,l}_{pk} \leftarrow U'^{s,l}_{pk} \sqrt{\mathbf{Q}^{s,l}_{pk} / (U'^{s,l}\mathbf{T}^{s,l})_{pk}}, \tag{2.23}$$

where $\mathbf{T}^{s,l} = \mathbf{B}^{s,l}_{(l)} \left(\mathbf{B}^{s,l}_{(l)}\right)^T$, and $\mathbf{Q}^{s,l} = \mathbf{X}^s_{(l)} \left(\mathbf{B}^{s,l}_{(l)}\right)^T$.

Updating $\mathcal{G}^s$. We vectorize $\mathcal{X}^s$ into x^s and derive the objective function involving entries of $\mathcal{G}^s$. The optimization function in Eq. (2.2) should be rewritten as:

$$\min_{g^s} \sum_{s=1}^{M} \sum_{l \in \mathbf{C}^s} (x^s_l - (g^s \bar{\mathbf{U}}^s)_l)^2 + \sum_j (g^s_j - g^*_j)^2. \tag{2.24}$$

where $\mathbf{C}^s = \{l : x^s_l \text{ is available}\}$. Similarly, we denote $\bar{\mathbf{U}}^s$ as the corresponding matrix for U^s, where l-th column is deleted when $l \notin \mathbf{C}^s$. The updating rules are:

$$g^s_l \leftarrow g'^s_l \sqrt{\frac{(\bar{\mathbf{U}}^s \mathbf{x}^s + \mathbf{g}^*)_l}{((\bar{\mathbf{U}}^s (\bar{\mathbf{U}}^s)^T + I)\mathbf{g}'^s)_l}}. \tag{2.25}$$

This approach only involves available entries in the computation, so it can greatly improve efficiency on sparse data.

3 Experiments

In this section, we show the experimental results that validate the proposed framework for trustworthiness analysis from the following four aspects: (1) We apply the framework to three real world applications: hotel rating analysis, network traffic flow analysis and weather prediction integration, and case studies show how the proposed approach detects meaningful alerts on untrustworthy items. (2) Synthetic data sets are adopted to conduct quantitative evaluation, showing the benefit of incorporating time factor. (3) Both offline and incremental versions of methods are implemented for streaming application scenario, and the results show that incremental algorithm runs much faster while the performance is almost the same. (4) Experiments show that by considering the sparseness of data, running time can be highly reduced by the technique we proposed to deal with missing values.

3.1 Real World Applications

We implement the proposed method on three real data sets, and show the experimental results in this section.

3.1.1 Hotel Rating

Hotel rating data sets are crawled for Las Vegas (LV) and New York City (NYC) from three popular travel websites: Orbitz, Priceline, and TripAdvisor. Among these websites, 111 common hotels are identified for Las Vagas, and 210 common hotels for New York City. We crawled all the ratings for these common hotels over three websites from January to December 2013 and all the rating information is normalized into the same scale $[0, 1]$.

Result Analysis. Figure 1 shows the inconsistency score distributions for hotels in LV and NYC. Most hotels receive low inconsistency scores, which confirms our assumption that the majority of items are consistent. Moreover, a few

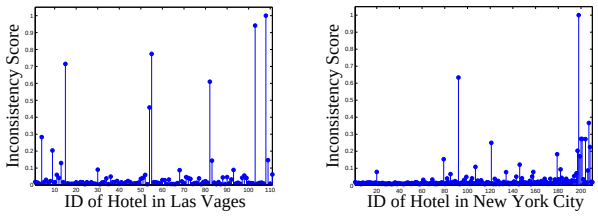


Figure 1: Inconsistency score distribution of hotels

hotels receive significantly high inconsistency scores, indicating that there exists inconsistency among the information of those hotels from multiple websites. Next we investigate these hotels to check whether the findings are meaningful by conducting two case studies on hotels at LV and NYC respectively.

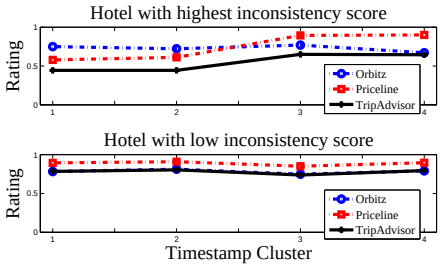


Figure 2: Case study: rating pattern of hotels in NYC

Case Study. In New York City data set, we choose the hotel with the highest inconsistency score and a random hotel with low inconsistency score, which are shown in Figure 2. In Figure 2, we can clearly observe that for the normal hotel (low inconsistency score), the trend of rating patterns for all websites are consistent, and the differences among them are the platform bias. For the hotel with high inconsistency score, the rating patterns are only consistent for Priceline and TripAdvisor, but for Orbitz, the trend of rating patterns at timestamp 3 and 4 are different from others'. This indicates the occurrence of some untrustworthy information.

3.1.2 Network Traffic Flow

The network traffic flow data set, collected from an enterprise network containing half million hosts, is used to detect anomalous hosts that generate suspicious traffic flow on weekdays. The behavior of each host evolves during a month. Each week can be considered as one timestamp. We collect three data sources, each of which consists of traffic flow of four week from a month. Data is normalized before fed into the proposed JNNTF.

Result Analysis. Figure 3 shows inconsistency score distribution for the first 400 hosts. The minority of hosts receive significantly high inconsistency scores, indicating inconsistent traffic flow across multiple sources. We conduct investigation on these hosts to check whether their behavior is inconsistent. Figure 4 shows the traffic pattern of two

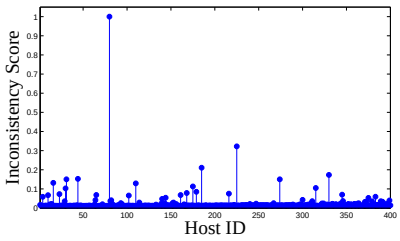


Figure 3: Inconsistent scores distribution of hosts

hosts. For the the normal hosts, the patterns of traffic flow for all sources are almost the same, indicating that its behavior is consistent across multiple sources. However, the trend of the host, whose inconsistency score is high, is only consistent between two sources. The pattern on the remaining source distinguishes dramatically from others' at the third timestamp, which indicates that it generates suspicious unusual traffic flow.

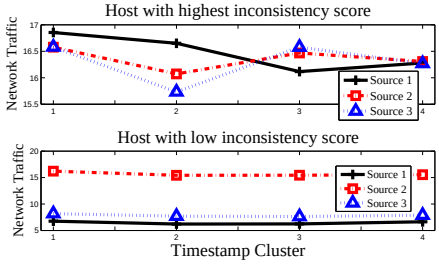


Figure 4: Case study: traffic flow pattern of hosts

3.1.3 Weather Forecast For this application task, we crawled the weather prediction information for 88 cities in United States from three sources: HAM weather (HAM), Wunderground (Wund), and World Weather Online (WWO). The crawling procedure last from Oct. 7, 2013 to Dec. 17, 2013, and the predicted high temperature is collected. As the temperature information has its own natural range, we don't normalize them, and the range is between -17F and 111F.

Result Analysis. Figure 5 shows the inconsistency score distribution for all cities, from which we can see most of them receive low inconsistency scores. It indicates that most cities are consistent with respect to high temperature across multiple sources. For cities receiving high inconsistency scores, there is inconsistency about weather prediction of those cities across multiple websites. Furthermore, to conduct a case study for justifying the ability of the proposed framework in detecting untrustworthy information, we pick the most inconsistent city and a normal city randomly. Figure 6 shows that for the normal city, although the trend of high temperature patterns deviate a little from each other, they are consistent across three sources. However, the most inconsistent city shows a different trend. There exist only consistent patterns between WWO and Wund, but for HAM, the trend of high temperature at timestamps 3 and 4 is totally different from the others. It implies the inconsistency on HAM's prediction compared with others'.

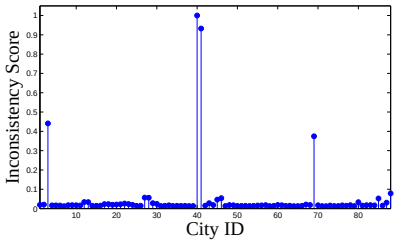


Figure 5: Inconsistency score distribution of cities

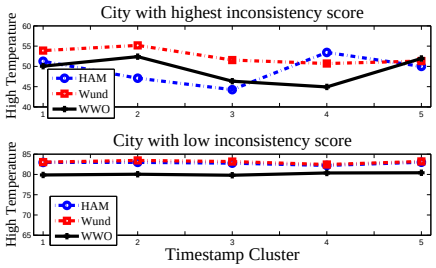


Figure 6: Case study: high temperature pattern of cities

3.2 Quantitative Evaluation In this section, we compare the proposed method with baseline methods. As synthetic data sets are adopted here, ground truth information is available and enables us to conduct quantitative analysis.

Data Generation. We generate synthetic data to simulate multi-platform rating data. The synthetic data are generated based on the observation that the behavior of latent user groups and timestamp clusters across multiple sources are consistent. We generate three sources and mandate three hidden user groups and three timestamp clusters. The basic idea is that when users from the same user group and timestamp cluster, their ratings over K items are drawn from the same distribution. Among the K items, we randomly choose n items ($n \leq K$) whose ratings are randomly changed and are treated as outliers receiving untrustworthy information.

Evaluation Metric. As we know which items should be detected as inconsistent ones, F-measure is adopted to evaluate the performance of all methods.

Baseline Methods. We compare the proposed model with the following baselines. The first one is Normalized Histogram Comparison (NHC). The second baseline method is Joint Matrix Factorization (JMF) from [10]. It assigns various users to latent groups by joint matrix factorization and retrieves the inconsistent items by comparing items at group level. The third baseline is Multi-Source Deep Belief Network (MSDBN) from [9], which is to learn a joint representation of ratings across multiple sources. Each source's rating is reconstructed based on the joint representation. Item's

inconsistency score is computed based on the difference between its actual rating and the reconstructed rating.

As all the above baselines don't take time into consideration, we run them on every time snapshot. Then, for each item, there are totally T (number of timestamps) inconsistency scores. The average of them is taken as the final output.

Performance Comparison. We vary the percentage of inconsistent items and total items to simulate different scenarios. Table 1 shows the performance comparison of our method and three baselines. NHC, whose performance is worst, fails to detect most of the inconsistent items. This method is vulnerable to noise and has limited discriminative ability, as it only compares statistics at high level. The performance of JMF improves over NHC mainly because it considers the user group-level information. MSDBN explores a much finer representation of common latent user groups than clustering, and thus its performance is better than JMF. However, compared with the proposed JNNTF, which has the perfect performance and reveals all the inconsistent items, some of which are still undetected by the baselines. This performance gap between baselines and JNNTF is mainly due to the fact that the baselines don't consider time information. The proposed JNNTF conducts tensor factorization on both user and timestamp dimensions, and thus can successfully detect inconsistent patterns hidden in user and timestamp clusters.

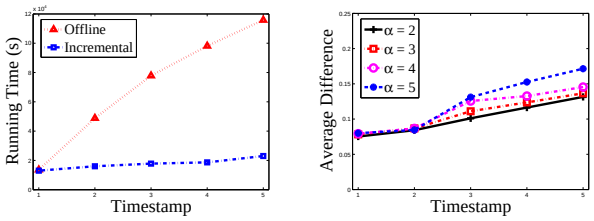
Table 1: F-measure comparison w.r.t. outlier percentage

	2%	3%	5%	6%	8%	10%
NHC	1.00	0.50	0.67	0.50	0.60	0.33
JMF	1.00	1.00	0.67	0.75	0.60	0.67
MSDBN	1.00	1.00	1.00	0.75	0.60	0.83
JNNTF	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

3.3 Streaming Scenario In this part, we evaluate the performance of the proposed approach on streaming data. In Section 2.3, we proposed an incremental version of our method, and further in Section 2.4, we claim that the incremental version (Algorithm 2) is much faster. We confirm this by the experimental results. Due to the page limitation, we only report the results on weather data set. We can observe similar patterns from the results on the other data sets.

In Figure 7 (a), we show the running time comparison between offline and incremental versions of the proposed method. We can see that the incremental algorithm benefits from efficient computation techniques, and takes less running time compared with the offline method.

For both incremental and offline methods, we compute the average consistency scores over all items. Figure 7 (b) shows the difference between them. It can be seen that the difference is small, and it increases slowly as time goes on, which is caused by the error accumulation. Figure 7 (b) also shows that when we vary the parameter α, the difference



(a) (b)

Figure 7: Algorithm analysis: (a) running time comparison; (b) average difference of algorithm 2 with respect to α

will not change significantly. α controls the weight of the regularization term in Eq. (2.2). This indicates that the performance of the incremental method is quite close to the offline version, while the incremental method takes less computation resources and unsensitive to the parameter α.

3.4 Data Sparseness We test the proposed approach that handles missing data. We vary the sparseness percentage of the synthetic data set and evaluate the performance of JNNTF and the algorithm that does not consider missing data (JNNTF-MD). The running time comparison is shown in Table 2. Columns 2 and 3 record their running time respectively, and the last column calculates the ratio between them. As the sparseness ratios increase, the running time of JNNTF-MD keeps almost the same while that of the proposed JNNTF decreases, because JNNTF-MD explicitly eliminates missing entries from computation. It indicates that it is useful to consider sparseness in the algorithm.

Table 2: Running time w.r.t. sparseness percentage

percentage	JNNTF	JNNTF-MD	ratio
90%	461.383s	1445.306s	31.9%
70%	460.788s	1451.650s	31.7%
50%	722.393s	1444.786s	50.0%
30%	953.113s	1394.353s	68.4%
10%	1009.149s	1341.918s	75.2%

4 Related Work

The proposed framework is built on tensor factorization, which attracts much attention recently [1, 13, 19]. However, most existing approaches simply factorize a tensor of single source. Although some recent work introduces the concept of multi-source tensor factorization [18, 20], they propose to model the source as one dimension of the tensor. Different from these work, the goal of this framework is to conduct a multi-source joint tensor factorization, where sources are connected through a consensus core tensor.

Information trustworthiness analysis is a hot topic with many applications such as social networks (e.g., Twitter) [14], online auction websites (e.g., eBay) [17], and product reviews (e.g., Amazon) [15]. Most related work focuses on

supervised learning approaches to detect untrustworthy information on single source. The main drawback of this type of approaches is the time-consuming labeling labor. Thus, we introduce an unsupervised learning framework to estimate information trustworthiness across multiple sources.

Previous work in [9, 10] propose joint matrix factorization and deep learning approaches to estimate the local trustworthiness of information. However, both works fail to consider time, and thus lots of useful information along time dimension is lost. In our framework, we propose to extract the latent user groups and timestamp clusters simultaneously.

Another relevant field is anomaly detection, which has been studied extensively for years [7]. These studies detect anomalies based on how far their distances [12], densities [6], statistical distributions [2], or chances of isolations [16] deviate from the rest of the data. However, they focus on detecting anomalies from single data source.

On multi-source data, where each view (source) describes one aspect of objects, people developed techniques for classification and clustering. In multi-view learning, a joint model is learnt from both labeled and unlabeled data of multiple sources to get the final classifier [3, 5]. Multi-view clustering explores the joint analysis of multiple views to compute a global clustering solution [4, 11]. These methods conduct a joint inference of a global model rather than detecting untrustworthy information from multiple sources.

5 Conclusions

In this paper, we develop a joint tensor factorization framework to detect inconsistent information from multiple streaming data, meanwhile consider time dimension. We compute inconsistency degree of information by comparing the corresponding projected tensors and the consensus one. We further propose an incremental approach to dynamically conduct joint tensor factorization when data continuously arrive with a guarantee on performance and efficiency. To handle sparse data set with many missing entries, we adapt the formulation to focus on available data and eliminate the effect of missing data in updating rules. Experiments on three real-world applications with dynamic multi-source data including hotel ratings, network traffic flow, and weather forecast demonstrate the advantage of the proposed method.

6 Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments and suggestions, which help us tremendously improve the paper's quality. We also thank Lu Su for his discussions. This work was supported in part by the National Science Foundation under Grant NSF IIS-1319973. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *SDM*, pages 701–712, 2010.

[2] D. Agarwal. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *Proc. of ICDM*, pages 5–12, 2005.

[3] A. Argyriou, C. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In *NIPS*, pages 25–32, 2008.

[4] S. Bickel and T. Scheffer. Multi-view clustering. In *Proc. of ICDM*, pages 19–26, 2004.

[5] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of COLT*, pages 92–100, 1998.

[6] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proc. of SIGMOD*, pages 93–104, 2000.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41:15:1–15:58, 2009.

[8] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proc. of KDD*, pages 126–135, 2006.

[9] L. Ge, J. Gao, X. Li, and A. Zhang. Multi-source deep learning for information trustworthiness estimation. In *Proc. of KDD*, pages 766–774, 2013.

[10] L. Ge, J. Gao, X. Yu, W. Fan, and A. Zhang. Estimating local information trustworthiness via multi-source joint matrix factorization. In *Proc. of ICDM*, pages 876–881, 2012.

[11] D. Greene and P. Cunningham. A matrix factorization approach for integrating multiple data views. In *Proc. of ECML/PKDD*, pages 423–438, 2009.

[12] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: algorithms and applications. *The VLDB Journal*, 8(3-4):237–253, 2000.

[13] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[14] K. Lee, J. Caverlee, and S. Webb. Uncovering social spammers: social honeypots + machine learning. In *Proc. of SIGIR*, pages 435–442, 2010.

[15] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proc. of CIKM*, pages 939–948, 2010.

[16] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *Proc. of ICDM*, pages 413–422, 2008.

[17] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proc. of WWW*, pages 201–210, 2007.

[18] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *Proc. of VLDB*, pages 697–708, 2005.

[19] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proc. of ICML*, pages 792–799, 2005.

[20] J. Sun, S. Papadimitriou, and S. Y. Philip. *Tensor Analysis on Multi-aspect Streams*. Springer, 2007.