# HARVESTER: Principled Factorization-based Temporal Tensor Granularity Estimation

Ravdeep S Pasricha* Uday Singh Saini † Nicholas D. Sidiropoulos ‡ Fei Fang §
Kevin Chan ¶ Evangelos E. Papalexakis ‖

**Abstract**

Given a tensor that captures temporal data, such as (user, item, time), the way that we set the granularity of the "time" mode can *make or break* our analysis of the data. If we set the granularity to be extremely fine, we end up with a very sparse and high-rank tensor which is essentially incompatible with what virtually all tensor decomposition models expect, i.e., tensors with low-rank structure, which can be expressed in some form of factorization.

Traditionally, this problem has been *avoided* by setting the granularity of the "time" to a "reasonable" aggregation (say hourly or daily intervals), an approach which has certainly served tensor analysis of temporal methods well so far. However, such an approach requires *tedious* trial-and-error experimentation across a number of such fixed aggregations, where typically the one that provides the most sensible results is retained, and furthermore it is *arbitrary*, since the optimal aggregation over time need not necessarily be uniform. In our work, we directly tackle this problem.

We introduce HARVESTER, the first principled factorization-based approach which seeks to identify the best temporal granularity of a given tensor. Unlike existing methods which follow a greedy approach, HARVESTER leverages multiple aggregated views of the tensor, and a carefully-designed optimization problem, in order to uncover an aggregation of a tensor which has a "good" structure for factor analysis or a downstream task.

We extensively evaluate HARVESTER on synthetic and real data, and demonstrate that it consistently produces tensors of very high quality, compared to the state-of-the-art, across the board for a number of different popular quality measures that have been used by the community.

## 1 Introduction

Tensor decomposition methods have been used to find latent structures in multi-modal data in a wide variety of applications like web link analysis [13], social network analysis [4], health care data analysis [11] and many more. When dealing with data that is temporal in nature, the granularity of data plays a crucial role in determining its usability in any data mining or machine learning algorithms. In applications like web mining, social networks or computer network analysis, if the data is collected at very fine temporal granularity the resulting tensor can be extremely sparse and noisy, in addition to being very high-dimensional. Unfortunately, tensors of that form are typically not amenable to popular and widely-used tensor decompositions[1], which seek to identify low-rank latent structure in the data.

There have been recent works which modify the decomposition algorithm in order to accommodate temporal irregularities, such as imposing smoothness [8] or burstiness [9] in the temporal factor or, especially in the streaming case, defining a "continuous tensor model" which accommodates incoming data in an "any-time" fashion by implicitly aggregating locally over time [15]. Instead of customizing, and thus restricting, the decomposition algorithm, in this work we seek to pre-process the tensor data in order to allow for flexible downstream analysis. Note that extreme aggregation hides away the temporal information (and the model identifiability boost that comes with it), while no aggregation leaves us with very sparse and possibly noisy data, not amenable to low-rank modeling. Hence the optimal level of aggregation lies somewhere in between, and our goal is to find it.

To circumvent the problem of having a long tensor in temporal mode and no exploitable structure, the most intuitive and widely-used approach is to aggregate the temporal mode using certain *fixed window sizes*. For example converting milliseconds to seconds, seconds to minutes, minutes to hours, hours to days, and so on. This solution can work, and has been serving tensor analysis of temporal data well for many years; however, there are a few issues with it, which we seek to mitigate in our work. First, it requires a copious amount of trial-and-error experimentation to evaluate what aggregation level works the best for the given data and tensor decomposition method. Second, and more importantly,

*University of California Riverside. Email: rpasr001@ucr.edu
†University of California Riverside. Email: usain001@ucr.edu
‡University of Virginia. Email: nikos@virginia.edu
§Carnegie Mellon University. Email: feifang@cmu.edu
¶DEVCOM Army Research Laboratory Email: kevin.s.chan.civ@army.mil
‖University of California Riverside. Email: epapalex@cs.ucr.edu

[1] We use the terms "decomposition" and "factorization" interchangeably.

tensor decomposition on these fixed size aggregated tensors may result in finding "good" latent structures however there might exist more "natural" aggregation which does not follow any fixed or arbitrary window size aggregations and might provide latent structures that are missed because of fixed window aggregation. To tackle this problem, [19, 20] introduced the problem *Trapped Under Ice* and provided greedy solutions IceBreaker and IceBreaker++, which iterate over the temporal mode and decide to aggregate two timestamps together or not based on certain utility functions. However, the greedy nature of those methods is prone to suboptimal solutions and requires a lot of recursive iterations, which hurts scalability, as we demonstrate in our experiments.

In this work, we introduce a principled way of finding an optimal aggregation of the temporal mode. We take inspiration from [2, 3] which essentially tackles the inverse problem to ours: given multiple views of an aggregated tensor in different modes (temporal and non-temporal), estimate the disaggregated tensor. Motivated by that approach, we propose Harvester where we leverage multiple aggregated views of a tensor, derived from the tensor in the original granularity in the temporal mode, and define a decomposition-based approach for recovering the best granularity.

Our contributions in this paper are as follows:

- **Novel Problem Formulation**: We provide a new and novel way to formulate the problem in terms of a factorization problem which can be solved using optimization techniques.
- **Efficient & Scalable Algorithm**: We propose Harvester, which solves the formulated problem in alternating fashion using non-negative multiplicative updates.
- **Experimental Evaluation**: We demonstrate the effectiveness of Harvester by performing extensive experiments on synthetic and real datasets.

## 2 Problem Formulation

**2.1 Preliminary Definitions** Tensors are multidimensional arrays (of order higher than two, i.e., indexed by three or more indices) which are used to model data that are multi-aspect in nature.

**Matricization of a tensor** is defined as unfolding or flatting a n-mode tensor in one of its modes to form a matrix. Consider a 3-mode tensor $\mathcal{X}$ of dimension $I \times J \times K$, which can be unfolded in three ways along each mode. Unfolding in first mode is given by $\mathbf{X_1}$ of dimension $I \times JK$ and similarly unfolding in other two modes is given by $\mathbf{X_2}$ and of $\mathbf{X_3}$ of dimension $J \times IK$ and $K \times IJ$ respectively.

The **n-mode product** of a tensor $\mathcal{X}$ of dimension $I_1 \times I_2 \cdots \times I_k \cdots \times I_n$ with a matrix $\mathbf{W}$ of dimension $I_k \times F$ is given by $\mathcal{Y} = \mathcal{X} \times_k \mathbf{W}$ where dimension of tensor $\mathcal{Y}$ is $I_1 \times I_2 \cdots \times I_{k-1} \times F \times I_{k+1} \cdots \times I_n$. This can also be extended to the matricization format as $\mathbf{Y_k} = \mathbf{W X_k}$.

The **CANDECOMP/PARAFAC decomposition** (henceforth referred to as CP) [7, 10] of a tensor $\mathcal{X}$ of size $I \times J \times K$ and (tensor) rank $F$ is a sum of $F$ rank-1 tensors (outer products of three vectors in the 3-D case) $\mathcal{X} \approx \sum_{r=1}^{F} a_r \circ b_r \circ c_r$, which is irreducible, in that an alternative decomposition cannot be found for $F' < F$. Here $\circ$ denotes the outer product and $a_r$, $b_r$, $c_r$ are vectors of dimension $I$, $J$, $K$ respectively. Another way of writing the above decomposition is $\mathcal{X} \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ where $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are factor matrices of dimension $I \times F$, $J \times F$ and $K \times F$ respectively such that each outer product of the $r^{th}$ column of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ forms one of the rank one tensors involved in the CP decomposition.

For more in-depth details on tensors and tensor decompositions we refer readers to the popular surveys in the literature [12, 22, 18].

**2.2 Problem Definition** Consider a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ with extremely fine raw temporal granularity, and which is very long and sparse/noisy in the temporal mode (in this example we consider the third mode as the temporal mode) which makes it ill-suited for compact modeling via CP decomposition. Our problem is:

> Given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and multiple views of the tensor $\mathcal{X}$, $\mathcal{Y}^1 \in \mathbb{R}^{I \times J \times K_1}$ and $\mathcal{Y}^2 \in \mathbb{R}^{I \times J \times K_2}$, where $K_2 < K_1 < K$. Find an aggregated tensor $\mathcal{Y} \in \mathbb{R}^{I \times J \times K^*}$ where $K^* < K_2 < K_1 < K$, such that its temporal mode latent factor matrix is an optimal low-rank compression of the original temporal mode latent factor, while respecting the temporal sequence of indices in that mode.

**Creating Views:** To give a concrete example, let's assume $\mathcal{X}$ is of size $I \times J \times 9$ and if we want to aggregate the third mode on some fixed window size, say 3, so $\mathbf{W}$ is of size $3 \times 9$ and takes the form:

$$\mathbf{W} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The resulting tensor will be of size $I \times J \times 3$, where the first 3 slices (1 to 3) are aggregated into one slice, the next 3 slices (4 to 6) into a second slice and last three slices (7 to 9) into the third slice.

## 3 Proposed Method

Consider a three mode tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ where $I$, $J$ denote the dimensions of the first two modes and $K$ that of the third mode which is temporal in nature and

has fine granularity, making tensor $\mathcal{X}$ extremely sparse and having no exploitable latent structure. The CP decomposition of tensor $\mathcal{X}$ for a rank $F$ is given by:

$$\mathcal{X} \approx [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C}]\!] \tag{3.1}$$

Where $\mathbf{A} \in \mathbb{R}^{I \times F}$, $\mathbf{B} \in \mathbb{R}^{J \times F}$ and $\mathbf{C} \in \mathbb{R}^{K \times F}$.

Let us assume there exists an oracle tensor $\mathcal{Y}$, which has perfect latent structure, which is created by the third mode-product of the tensor $\mathcal{X}$ with a matrix $W_{opt}$ which has special structure as specified in Section 2.2

$$\mathcal{Y} = \mathcal{X} \times_3 \boldsymbol{W_{opt}} \tag{3.2}$$

$W_{opt}$ is of size $K_{\text{opt}} \times K$ so the dimension of $\mathcal{Y}$ is $I \times J \times K_{\text{opt}}$, such that $K_{\text{opt}} << K$. The CP decompositon of oracle tensor $\mathcal{Y}$ is given by

$$\mathcal{Y} \approx [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_{opt}}]\!] \tag{3.3}$$

where matrix $\mathbf{C_{opt}}$ is of size $K_{\text{opt}} \times F$

The key assumption we make is that $\mathbf{C}$ can be obtained by some dimensionality-expanding linear transformation of the unknown (latent) $\mathbf{C}_{\text{opt}}$. This is tantamount to assuming that there exists low rank matrix structure in the temporal mode.

$$\mathbf{C} = \mathbf{M}\mathbf{C}_{\text{opt}} \tag{3.4}$$

where $\mathbf{M} \in \mathbb{R}^{K \times K_{\text{opt}}}$ and $\mathbf{M}$ is a tall matrix.

We create $m \geq 2$ views of tensor $\mathcal{X}$ as follows:

$$\mathcal{Y}^1 = \mathcal{X} \times_3 \boldsymbol{W_1} = [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_1}]\!];\ \mathbf{C_1} = \mathbf{W_1}\mathbf{C} \tag{3.5}$$

$$\mathcal{Y}^2 = \mathcal{X} \times_3 \boldsymbol{W_2} = [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_2}]\!];\ \mathbf{C_2} = \mathbf{W_2}\mathbf{C} \tag{3.6}$$

$$\vdots$$

$$\mathcal{Y}^m = \mathcal{X} \times_3 \boldsymbol{W_m} = [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_m}]\!];\ \mathbf{C_m} = \mathbf{W_m}\mathbf{C} \tag{3.7}$$

Before we formalize the problem, we need to address some issues regarding the problem

1. Since we don't have access to the oracle tensor $\mathcal{Y}$ we cannot directly estimate $\mathbf{C_{opt}}$.
2. Most importantly, unlike in the case of [2, 3], where the "right" $K_{\text{opt}}$ is known beforehand, here we don't know the size $K_{\text{opt}}$ for the matrix $\mathbf{C_{opt}}$.

Thus, we seek to estimate $\widetilde{\mathbf{C}}$ as shown in equation 3.9, which is supposed to be good approximation of $\mathbf{C_{opt}}$. Using equation 3.4 and the fact that $\mathbf{C}_i = \mathbf{W}_i\mathbf{C}$ per equations 3.5-3.7, we have,

$$\mathbf{C}_i = \mathbf{W}_i\mathbf{M}\mathbf{C}_{opt} \tag{3.8}$$

In the above equation, we know $\mathbf{C}_i$ and $\mathbf{W_i}$, but we do not know $\mathbf{M}$ and $\mathbf{C}_{opt}$ – not even $K_{\text{opt}}$. As a result, we do not know the product $\mathbf{W}_i\mathbf{M}$. How can we bypass this seemingly insurmountable challenge? We propose to bring in ideas from sparse regression. The idea is to over-parameterize the product $\mathbf{W}_i\mathbf{M}$ as a new matrix variable $\mathbf{P_i}$, and use a sparse diagonal row-selection matrix to pick up the essential (reduced-dimension) row span of $\mathbf{C}_{opt}$. In more detail, we approximate

$$\mathbf{C}_i \approx \mathbf{P}_i\boldsymbol{\Lambda}\widetilde{\mathbf{C}} \tag{3.9}$$

where $\mathbf{P}_i$ is of size $K_i \times K$, $\widetilde{\mathbf{C}}$ is of size $K \times F$, and $\boldsymbol{\Lambda}$ is a sparse diagonal matrix of size $K \times K$, whose zero entries are meant to strike out columns of $\mathbf{P}_i$ and corresponding rows of $\widetilde{\mathbf{C}}$. The product $\mathbf{P}_i\boldsymbol{\Lambda}$ is a proxy for $\mathbf{W}_i\mathbf{M}$, with the diagonal $\boldsymbol{\Lambda}$ effectively providing us with a tall matrix of *a priori* unknown number of columns $K_{opt}$.

Using equation 3.9, our optimization problem is:

$$\mathcal{L} = \min_{\widetilde{\mathbf{C}},\mathbf{P}_i,\boldsymbol{\Lambda}} \sum_{i=1}^{m} ||\boldsymbol{C}_i - \boldsymbol{P}_i\boldsymbol{\Lambda}\widetilde{\boldsymbol{C}}||_F^2 + \alpha||\widetilde{\boldsymbol{C}}^{\boldsymbol{T}}||_F^2 + \beta||\boldsymbol{\Lambda}||_1 + \gamma||\boldsymbol{P_i}||_F^2 \tag{3.10}$$

We optimize equation 3.10 for matrices $\widetilde{\mathbf{C}}$, $\boldsymbol{\Lambda}$ and $\mathbf{P}_i$. the sparsity or L-1 regularization constraint on $\boldsymbol{\Lambda}$ is there to help reduce the granularity of the the $\widetilde{\mathbf{C}}$. The non-zero entries in the diagonal of $\boldsymbol{\Lambda}$ control the final granularity of the resulting tensor, thus we impose a sparsity penalty on the diagonal.

**Discussion**: Since we are trying to find the best aggregation in the third mode and we assume that there exists a low dimensional structure in that mode, one may ask why not just perform singular value decomposition (SVD) on the third mode matricization of the tensor and use the top-k right singular vectors as the basis for the aggregation, which would yield the best compression in the least squares sense.

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{svd}(\mathbf{X_3})$$
$$\mathcal{Y} = \mathcal{X} \times_3 \mathbf{V}_k$$

In our case, this does not solve the problem we set out to tackle. Even though there may be low-rank structure in the temporal mode, captured by the aggregation matrix $\mathbf{V}_k$, this aggregation does not respect the temporal sequence of slices, therefore is not appropriate for recovering the type of aggregation we are interested in.

### 3.1 Proposed method: Harvester

In this section we present our method HARVESTER to solve the problem in equation 3.10. We solve it using non negative multiplicative update (NMU) [14] in an alternating fashion. Here, we derive the update steps only using two views but this can easily be extended to more views.

Using tensor $\mathcal{X}$ we create two views for that tensor namely $\mathcal{Y}^1$ and $\mathcal{Y}^2$ which have lower resolution than the original tensor. To create these views we multiply the third mode of the tensor with a $\mathbf{W}_1$ and $\mathbf{W}_2$ matrix.

$$\mathcal{Y}^1 = \mathcal{X} \times_3 \boldsymbol{W}_1 \tag{3.11}$$

$$\mathcal{Y}^2 = \mathcal{X} \times_3 \boldsymbol{W}_2 \tag{3.12}$$

We create a tensor $\mathcal{Z}$ by stacking tensor $\mathcal{Y}^1$ and $\mathcal{Y}^2$ onto one another hence making them coupled in the first two modes. The dimension of tensor $\mathcal{Z}$ is $I \times J \times (K_1 + K_2)$. Then we perform CP decomposition on tensor $\mathcal{Z}$ which yields factor matrices needed for our optimization algorithm. We split factor matrix $\mathbf{D}$ into $\mathbf{C}_1$ and $\mathbf{C}_2$ based on their respective dimensions as shown below:

$$\begin{aligned} \mathcal{Z} &= [\mathcal{Y}^1; \mathcal{Y}^2] \\ \mathcal{Z} &\approx [\![\mathbf{A},\ \mathbf{B},\ \mathbf{D}]\!] \\ \mathbf{C}_1 &= \mathbf{D}(1 : K_1, :) \\ \mathbf{C}_2 &= \mathbf{D}(K_1 + 1 : K_2, :) \end{aligned} \tag{3.13}$$

We use equations 3.9 and 3.13 to create our optimization problem that follows from equation 3.10. The optimization steps to solve equation 3.14 are derived along the lines of [21].

$$\mathcal{L} = \min_{\widetilde{\mathbf{C}}, \mathbf{P}_1, \mathbf{P}_2 \boldsymbol{\Lambda}} \underbrace{||\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}}||_F^2}_{f} + \underbrace{||\boldsymbol{C}_2 - \boldsymbol{P}_2 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}}||_F^2}_{g} + \underbrace{\alpha||\widetilde{\boldsymbol{C}}^{\boldsymbol{T}}||_F^2 + \beta||\boldsymbol{\Lambda}||_1 + \gamma||\boldsymbol{P}_1||_F^2 + \gamma||\boldsymbol{P}_2||_F^2}_{h} \tag{3.14}$$

The gradient of $\mathcal{L}$ in eq. 3.14 w.r.t. $\widetilde{\mathbf{C}}$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \widetilde{\mathbf{C}}} &= \frac{\partial f}{\partial \widetilde{\mathbf{C}}} + \frac{\partial g}{\partial \widetilde{\mathbf{C}}} + \frac{\partial h}{\partial \widetilde{\mathbf{C}}} \\ \frac{\partial f}{\partial \widetilde{\mathbf{C}}} &= \frac{\partial}{\partial \widetilde{\mathbf{C}}}(||\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}}||_F^2) \\ \frac{\partial f}{\partial \widetilde{\mathbf{C}}} &= \frac{\partial}{\partial \widetilde{\mathbf{C}}} \operatorname{Tr}((\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})(\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})^T) \\ \frac{\partial f}{\partial \widetilde{\mathbf{C}}} &= -\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{C}_1 - \boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{C}_1 + \boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} + \boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} \\ \frac{\partial f}{\partial \widetilde{\mathbf{C}}} &= \mathbf{2}\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} - 2\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{C}_1 \end{aligned} \tag{3.15}$$

We can similarly derive $\frac{\partial g}{\partial \widetilde{\mathbf{C}}}$,

$$\frac{\partial g}{\partial \widetilde{\mathbf{C}}} = \mathbf{2}\boldsymbol{\Lambda}^T \mathbf{P}_2^T \mathbf{P}_2 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} - 2\boldsymbol{\Lambda}^T \mathbf{P}_2^T \tag{3.16}$$

Gradient of $h$ w.r.t $\widetilde{\mathbf{C}}$

$$\begin{aligned} \frac{\partial h}{\partial \widetilde{\mathbf{C}}} &= \frac{\partial}{\partial \widetilde{\mathbf{C}}} \operatorname{Tr}(\alpha \widetilde{\mathbf{C}}^{\mathbf{T}} \widetilde{\mathbf{C}}) \\ &= \mathbf{2}\alpha \widetilde{\mathbf{C}} \end{aligned} \tag{3.17}$$

Combining equation 3.15, 3.16 and 3.17

$$\frac{\partial \mathcal{L}}{\partial \widetilde{\mathbf{C}}} = 2(\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} + \boldsymbol{\Lambda}^T \mathbf{P}_2^T \mathbf{P}_2 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} - \boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{C}_1 - \boldsymbol{\Lambda}^T \mathbf{P}_2^T \mathbf{C}_2 + \alpha \widetilde{\mathbf{C}}) \tag{3.18}$$

Setting $\frac{\partial \mathcal{L}}{\partial \widetilde{\mathbf{C}}}$ to zero and using non-negative multiplicative update method to compute update step. $\circledast$ here represents element wise product and fraction equation below represents element wise division.

$$\boxed{\widetilde{\mathbf{C}} \Leftarrow \widetilde{\mathbf{C}} \circledast \frac{\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{C}_1 + \boldsymbol{\Lambda}^T \mathbf{P}_2^T \mathbf{C}_2}{(\boldsymbol{\Lambda}^T \mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} + \boldsymbol{\Lambda}^T \mathbf{P}_2^T \mathbf{P}_2 \boldsymbol{\Lambda} + \alpha I)\widetilde{\mathbf{C}}}} \tag{3.19}$$

The gradient of $\mathcal{L}$ in eq. 3.14 w.r.t $\mathbf{P}_1$ (or $\mathbf{P}_2$ ):

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{P}_1} &= \frac{\partial f}{\partial \mathbf{P}_1} + \frac{\partial g}{\partial \mathbf{P}_1} + \frac{\partial h}{\partial \mathbf{P}_1} \\ \frac{\partial f}{\partial \mathbf{P}_1} &= \frac{\partial}{\partial \mathbf{P}_1}(||\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}}||_F^2) \\ \frac{\partial f}{\partial \mathbf{P}_1} &= \frac{\partial}{\partial \mathbf{P}_1} \operatorname{Tr}((\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})(\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})^T) \\ &= 2(\mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T - \mathbf{C}_1 \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T) \end{aligned} \tag{3.20}$$

Gradient of $g$ with respect to $\mathbf{P}_1$ is zero. Now computing gradient of $h$ with respect to $\mathbf{P}_1$

$$\begin{aligned} \frac{\partial h}{\partial \mathbf{P}_1} &= \frac{\partial}{\partial \mathbf{P}_1} \operatorname{Tr}(\gamma \mathbf{P}_1 \mathbf{P}_1^T) \\ &= 2\gamma \mathbf{P}_1 \end{aligned} \tag{3.21}$$

Combining equation 3.20 and 3.21

$$\frac{\partial \mathcal{L}}{\partial \mathbf{P}_1} = 2(\mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T - 2\mathbf{C}_1 \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T + 2\gamma \mathbf{P}_1) \tag{3.22}$$

Setting $\frac{\partial \mathcal{L}}{\partial \mathbf{P}_1}$ to zero and using non-negative multiplicative update method to compute update step. Again $\circledast$ here represents element wise product and fraction equation below represents element wise division.

$$\boxed{\mathbf{P}_1 \Leftarrow \mathbf{P}_1 \circledast \frac{\mathbf{C}_1 \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T}{\mathbf{P}_1(\boldsymbol{\Lambda} \widetilde{\mathbf{C}} \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T + \gamma \mathbf{I})}} \tag{3.23}$$

We can compute update step for $\mathbf{P}_2$ similarly,

$$\boxed{\mathbf{P}_2 \Leftarrow \mathbf{P}_2 \circledast \frac{\mathbf{C}_2 \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T}{\mathbf{P}_2(\boldsymbol{\Lambda} \widetilde{\mathbf{C}} \widetilde{\mathbf{C}}^T \boldsymbol{\Lambda}^T + \gamma \mathbf{I})}} \tag{3.24}$$

The gradient of $\mathcal{L}$ in eq. 3.14 with respect to $\boldsymbol{\Lambda}$:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Lambda}} &= \frac{\partial f}{\partial \boldsymbol{\Lambda}} + \frac{\partial g}{\partial \boldsymbol{\Lambda}} + \frac{\partial h}{\partial \boldsymbol{\Lambda}} \\ \frac{\partial f}{\partial \boldsymbol{\Lambda}} &= \frac{\partial}{\partial \boldsymbol{\Lambda}}(||\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}}||_F^2) \\ \frac{\partial f}{\partial \boldsymbol{\Lambda}} &= \frac{\partial}{\partial \boldsymbol{\Lambda}} \operatorname{Tr}((\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})(\boldsymbol{C}_1 - \boldsymbol{P}_1 \boldsymbol{\Lambda} \widetilde{\boldsymbol{C}})^T) \\ &= 2(\mathbf{P}_1^T \mathbf{P}_1 \boldsymbol{\Lambda} \widetilde{\mathbf{C}} \widetilde{\mathbf{C}}^T - \mathbf{P}_1^T \mathbf{C}_1 \widetilde{\mathbf{C}}^T) \end{aligned} \tag{3.25}$$

We can similarly derive the derivative of $g$ w.r.t. $\mathbf{\Lambda}$,

$$\frac{\partial g}{\partial \mathbf{\Lambda}} = 2(\mathbf{P}_2^T\mathbf{P}_2\mathbf{\Lambda}\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^T - \mathbf{P}_2^T\mathbf{C}_2\widetilde{\mathbf{C}}^T) \tag{3.26}$$

Next, based on the update steps in Section 2 of [14], we compute the Gradient of $h$ w.r.t. $\mathbf{\Lambda}$

$$\frac{\partial h}{\partial \mathbf{\Lambda}} = \frac{\partial}{\partial \mathbf{\Lambda}}||\beta\mathbf{\Lambda}||_1 = \beta\mathbf{E} \tag{3.27}$$

where $\mathbf{E}$ is a matrix of all ones. We would like to reiterate that, although L-1 Norm is not a differentiable function in the entirety of its domain and thus we must instead compute its sub-gradient, the constraint of non-negativity on $\mathbf{\Lambda}$ allows us to circumvent that issue and compute the gradient as long as values of $\mathbf{\Lambda}$ are not negative (which we guarantee by non-negative initialization and subsequent iterations).

Combining equations 3.25, 3.26 and 3.27

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Lambda}} = 2\mathbf{P}_1^T\mathbf{P}_1\mathbf{\Lambda}\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^T - 2\mathbf{P}_1^T\mathbf{C}_1\widetilde{\mathbf{C}}^T + 2\mathbf{P}_2^T\mathbf{P}_2\mathbf{\Lambda}\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^T - 2\mathbf{P}_2^T\mathbf{C}_2\widetilde{\mathbf{C}}^T + \beta\mathbf{E} \tag{3.28}$$

Setting $\frac{\partial \mathcal{L}}{\partial \mathbf{\Lambda}}$ to zero, we use the non-negative multiplicative update method to compute the update step. $\circledast$ represents element wise product and fraction equation below represents element wise division.

$$\boxed{\mathbf{\Lambda} \Leftarrow \mathbf{\Lambda} \circledast \frac{2\mathbf{P}_1^T\mathbf{C}_1\widetilde{\mathbf{C}}^T + 2\mathbf{P}_2^T\mathbf{C}_2\widetilde{\mathbf{C}}^T}{2\mathbf{P}_1^T\mathbf{P}_1\mathbf{\Lambda}\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^T + 2\mathbf{P}_2^T\mathbf{P}_2\mathbf{\Lambda}\widetilde{\mathbf{C}}\widetilde{\mathbf{C}}^T + \beta\mathbf{E}}} \tag{3.29}$$

**Aggregation strategies** So far, we have computed $\mathbf{\Lambda}$, $\widetilde{\mathbf{C}}$, $\mathbf{P_1}$ and $\mathbf{P_2}$ which are needed to construct the aggregated tensor. We can construct the aggregated tensor in the following ways, given $\mathbf{C_o} = \mathbf{\Lambda}\widetilde{\mathbf{C}}$

*-aggregateOnNorm*: We use one of the techniques from [20], namely norm aggregation and apply it to matrix instead of tensor. Essentially we iterate over the rows of $\mathbf{C_o}$, add a candidate row to previous row, if the rate of change of norm between the previous row and sum of previous and candidate row is more than a certain threshold. If not, the candidate row becomes the previous row and process continues until we reach the end. We use that norm aggregated matrix $\mathbf{C_{norm}}$ to construct a tensor.

$$\begin{aligned}\mathbf{W_{norm}} &= aggregateOnNorm(\mathbf{C_o}, normThreshold)\\ \mathbf{C}_{norm} &= \mathbf{W}_{norm}\mathbf{C}_o\\ \mathcal{Y}^{norm} &\approx [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_{norm}}]\!]\end{aligned}$$

*-aggregateZeroRows*: Since $\Lambda$ is not only a diagonal matrix but it's also a diagonally sparse matrix, which is used in update steps to turn off (zero out) the rows of $\widetilde{C}$. We can just use those non zero rows as anchor points to create a matrix $\mathbf{C_{spar}}$ which contains only the non-zero rows of $\mathbf{C_o}$.

$$\begin{aligned}\mathbf{W}_{spar} &= aggregateZeroRows(\mathbf{C}_o)\\ \mathbf{C_{spar}} &= \mathbf{W_{spar}}\mathbf{C}_o\\ \mathcal{Y}^{spar} &\approx [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C_{spar}}]\!]\end{aligned}$$

We use tensors $\mathcal{Y}^{norm}$ and $\mathcal{Y}^{spar}$ for evaluation.

## 4 Experimental Evaluation

We implemented our method[2] in Matlab using tensor toolbox [5] and for using CP-WOPT [1] algorithm in tensor toolbox for tensor completion we make use of L-BFGS-B Matlab wrapper [3].

**Hyperparameter Tuning:** Our problem formulation has three hyperparameters which need to be tuned namely $\alpha$, $\beta$ and $\gamma$ as in equation 3.14. We use different ranges for the hyperparameters search space used in equation 3.14 based on the dataset, we specify those ranges while discussing the results for those datasets in their respective sub sections. We perform grid search over these values and choose two points from this grid search results based on following conditions:

**Sparsity threshold:** Number of non zero entries in diagonal of lambda ($\mathbf{\Lambda}$) matrix is less then some threshold. For synthetic dataset we set that threshold to be 1/10 of the third dimension of $\mathcal{X}$, e.g., for $\mathcal{X}$ being $100 \times 100 \times 500$, the threshold is set at $500/10 = 50$.

**Error threshold:** From all the points that meet the above threshold condition, we find the median error and only consider points with error less than that threshold.

After applying the above two thresholds, we choose two points, one which has smallest error on the loss function (min error point) and other one which have the lowest number of non-zero entries on the diagonal of the lambda ($\mathbf{\Lambda}$) matrix (min sparsity point). We use these two points for our analysis. For each of these points we create two tensors namely $\mathcal{Y}_{norm}$ and $\mathcal{Y}_{spar}$ as mentioned in previous section. In total, HARVESTER produces 4 candidate tensors for evaluation namely min error $\mathcal{Y}_{norm}$, min error $\mathcal{Y}_{spar}$, min sparsity $\mathcal{Y}_{norm}$, and min sparsity $\mathcal{Y}_{spar}$ The result presented in this work for our method HARVESTER will be of one or more of those points unless specified otherwise.

### 4.1 Evaluation Metrics and Baselines

We evaluate our method HARVESTER on two model-based measures namely relative fit of the model and CORCONDIA [6, 17] and also on a task-based measure on tensor completion [1]. In order to choose the "best" rank for the

[2] https://github.com/ravdeep003/harvester-sdm

[3] https://github.com/stephenbeckr/L-BFGS-B-C

tensor at hand, we use AutoTen [17] (modified with non-negativity constraints). Given the rank identified by AutoTen, we compute the following quality measures:
**Relative Fit of the model**: We evaluate the fit of non-negative CP using the AutoTen rank:

$$Relative\ Fit = 1 - \Big(\frac{||\mathcal{X}_{Input} - \mathcal{X}_{computed}||_F}{||\mathcal{X}_{Input}||_F}\Big) \tag{4.30}$$

**CORCONDIA:** The Core Consistency Diagnostic (CORCONDIA) [6] returns a score for how well the given factors obey the CP model, given a tensor and a decomposition at a certain rank. AutoTen, uses CORCONDIA to arrive to the selected rank, so we are essentially using the value reported by AutoTen.
**Tensor Completion:** This can be viewed as downstream task oriented evaluation, in which we hide 20% of non-zero entries and use CP_WOPT [1] for predicting the missing values. We report RMSE between the predicted and actual hidden values. The lower the RMSE the better the performance on the tensor completion task. For consistency with the previous two measures, for which higher is better, we compute $1/RMSE$.

**Baselines**: We compare against the following:
**IceBreaker++**: We compare our method HARVESTER against the greedy based method ICEBREAKER++ [20], which introduced the problem of *Trapped Under Ice* and provided a greedy approach to tackle the problem.
**Fixed aggregations**: We also compare our method against fixed aggregations views generated from the input tensor, which just aggregates the slices based on certain fixed window intervals.

**4.2 Synthetic data generation** We created the synthetic datasets in a similar fashion as mentioned in the [20]. We first create a sparse tensor of certain sparsity and then we distribute the non-zero entries in each frontal slice $\mathcal{X}(:,:,k)$ over certain fixed number of slices ($B$). We repeat this for every frontal slice in original tensor, every time distributing the non-zero entries to new fixed bucket of slices($B$) and concatenate all the buckets to create the dataset. Table 1 shows the different synthetic datasets used in the experiments. For example take SD1 (synthetic dataset 1), the original data is of size $100 \times 100 \times 10$, all the non-zero entries in the $k$-th slice $\mathcal{X}(i,j,k)$ are randomly distributed over 50 slices in the third mode with the same $i,j$ indices.

We create 4 types of dataset as shown in table 1, for each type of dataset we create 10 datasets and we report our findings for these 40 datasets. Hyper-parameter ranges used are : for $\alpha$ we use the values $[10^{-4}, 10^{-3}, 10^{-2}]$, for $\beta$ we use $[10^{-2}, 10^{-1}, 10^{0}, 10]$ and for $\gamma$ we use $[10^{-4}, 10^{-3}, 10^{-2}]$.

**4.3 Pareto-based evaluation for synthetic data** We evaluate the quality of our method HARVESTER and baseline methods on the three evaluation criteria as mentioned in section 4.1. We are seeking a tensor which maximizes CP_FIT, CORCONDIA and minimizes the RMSE for tensor completion (or maximize 1/RMSE). This is essentially a multi-objective optimization problem, and a widely accepted means of measuring optimality in this case is to identify points in that 3D space which lie in the so-called *Pareto frontier* [16], i.e., points which dominate the rest of the points with respect to the three dimensions/objectives. To that effect, we find the tensors which lie on the Pareto frontier of these evaluation metrics, and we count how frequently a given method produces such tensors. The more frequently a method yields tensors lying on the Pareto frontier, the better the quality of those tensors, thus, the higher performing the method. We perform Pareto finding operation on all the data points obtained using method HARVESTER, ICEBREAKER++, fixed aggregation views of the tensor which we used with our method. In finding Pareto-optimal points we excluded the original tensor.

To measure the effectiveness of our method HARVESTER against baselines we count the number of times each method is on the Pareto frontier for all the synthetic datasets we evaluated on and we present that information in table 2. We observe that points from our method HARVESTER are always on the Pareto frontier with variety of the synthetic datasets. As shown in table 2, HARVESTER performs far better than the baseline methods. In all of the synthetic datasets, we see that tensor generated by HARVESTER was found on the Pareto frontier. In SD4 ICEBREAKER++ did reasonably well, as it was on the Pareto frontier six times as opposed to HARVESTER which was on the Pareto frontier for all ten datasets.

**4.3.1 Rank Sensitivity** In this section, we explore the sensitivity of our algorithm HARVESTER with respect to different ranks. We run this experiment using a dataset of size $100 \times 100 \times 10$ generated using rank 10, which was then used to generate a fine grained tensor of size $100 \times 100 \times 500$, using the same method as described in section 4.3. In Figure 1(a), 1(b) and 1(c) we report mean and standard deviation of Corcondia, Relative fit and RMSE for the dataset over 5 runs. We observe that HARVESTER has a steady performance for the initial rank increase but further increase in the rank deteriorate some of the evaluation metrics.

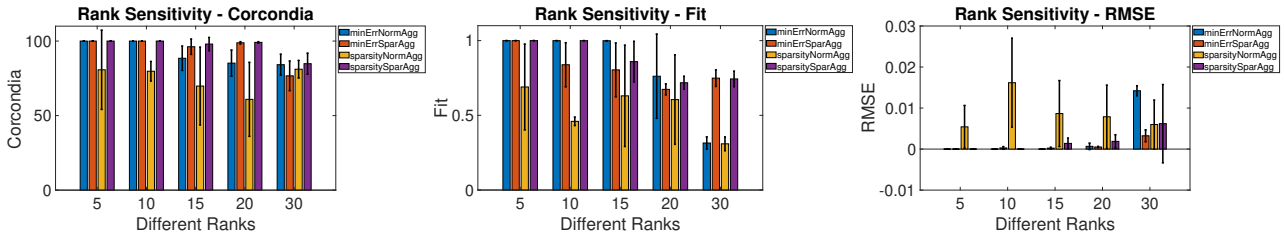
**4.4 Scalability Analysis** In this section we show the scalability analysis for HARVESTER as the third mode grows, since HARVESTER depends on the views

| Dataset | Original Dimension ($\mathcal{X}_{\text{og}}$) | Rank (R) | Bucket size ($B$) | Approximate Final Dimension( $\mathcal{X}$) | View 1 ($\mathcal{Y}^1$) | View 2 ($\mathcal{Y}^2$) | Number of datasets |
|---|---|---|---|---|---|---|---|
| SD1 | $100 \times 100 \times 10$ | 5 | 50 | $100 \times 100 \times 500$ | $100 \times 100 \times 100$ | $100 \times 100 \times 50$ | 10 |
| SD2 | $100 \times 100 \times 10$ | 10 | 50 | $100 \times 100 \times 500$ | $100 \times 100 \times 100$ | $100 \times 100 \times 50$ | 10 |
| SD3 | $100 \times 100 \times 10$ | 30 | 50 | $100 \times 100 \times 500$ | $100 \times 100 \times 100$ | $100 \times 100 \times 50$ | 10 |
| SD4 | $100 \times 100 \times 100$ | 20 | 20 | $100 \times 100 \times 2000$ | $100 \times 100 \times 400$ | $100 \times 100 \times 200$ | 10 |

Table 1: Table of Synthetic Datasets analyzed

| Dataset | Original Dimension ($\mathcal{X}_{\text{og}}$) | Approximate Final Dimension( $\mathcal{X}$) | Harvester Count | IceBreaker Count | Views Count |
|---|---|---|---|---|---|
| SD1 | $100 \times 100 \times 10$ | $100 \times 100 \times 500$ | **10** | 0 | 0 |
| SD2 | $100 \times 100 \times 10$ | $100 \times 100 \times 500$ | **10** | 0 | 0 |
| SD3 | $100 \times 100 \times 10$ | $100 \times 100 \times 500$ | **10** | 1 | 0 |
| SD4 | $100 \times 100 \times 100$ | $100 \times 100 \times 2000$ | **10** | 6 | 0 |

Table 2: How many times each method was on the Pareto frontier




(a) CORCONDIA (b) Fit (c) RMSE

Figure 1: HARVESTER sensitivity as a function of the rank on a $100 \times 100 \times 10$ dataset with true rank 10

of the original tensor. We run HARVESTER on the different settings one with coarser views and other one with finer views. In Figure 2(a), we report time taken by HARVESTER for both of these inputs. When the third mode is smaller both aggregations roughly take the same time, but as the dimension grows we see that HARVESTER with coarser aggregation runs faster.

We also evaluated the scalability of HARVESTER against ICEBREAKER++. Figure 2(b) shows the time taken by each method. Since ICEBREAKER++ depends on the performance of the utility functions, we ran ICEBREAKER++ without the missing value prediction utility function (the most time consuming utility function used in that work). We also ran ICEBREAKER++ with one iteration of missing value prediction utility function as opposed to 10 times reported in the paper [20]. We compared it with HARVESTER with both coarser and fine grained views. As shown in Figure 2(b), HARVESTER is *significantly faster* than ICEBREAKER++.

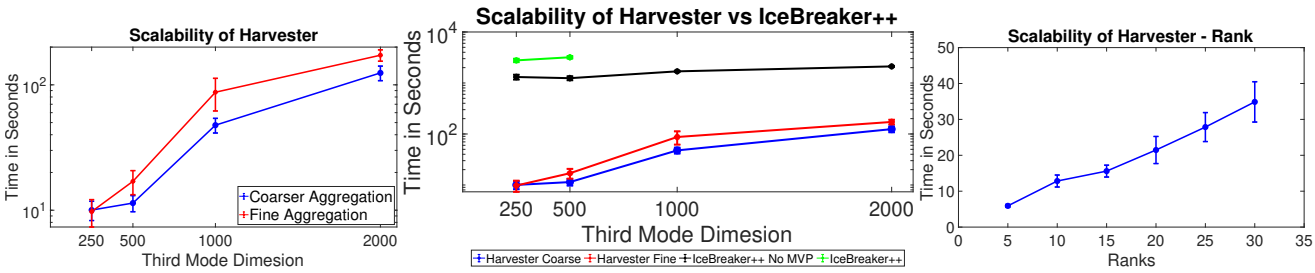
Finally, we evaluate the behavior of HARVESTER as the rank which is used to compute the coupled tensor factorization to set up the problem with $\mathbf{C}_i$'s increases. For an input dataset to HARVESTER of size $100 \times 100 \times 500$ we ran it 10 times for each rank and report runtimes in Figure 2(c). We observe that HARVESTER scales linearly with respect to rank.

### 4.5 Real-world case study: Foursquare Dataset

We use the foursquare dataset[4] [25] for check-ins in New York City collected over 10 months from 12 April, 2012 to 16 February, 2013. We constructed a 3-mode tensor with user, venue categories and time as the three modes where each entry in tensor is number of check-in for a user for a particular venue category for a given day. We aggregated temporal mode on the daily basis to create an input tensor and we used top 126 venue categories (venue categories which had more check-ins than the median check-in value of each categories). We, thus, end up with a tensor of size $1083 \times 126 \times 250$ (user, venue categories, days). We create two views from the input tensor of size $1083 \times 126 \times 125$ and $1083 \times 126 \times 50$ respectively. These two views were used by HARVESTER to find tensor of optimal granularity. By conducting the same Pareto-based evaluation, we observed that *only* HARVESTER *tensors are on the Pareto frontier.*

In the interest of space, we present a small indicative set of the latent factors extracted from one of those

[4] `https://www.kaggle.com/datasets/chetanism/foursquare-nyc-and-tokyo-checkin-dataset`




(a) Scalabiltiy wrt. third mode dim. (b) HARVESTER vs ICEBREAKER++ scalability (c) Scalability wrt. rank

Figure 2: Scalability Analysis

HARVESTER tensors. Figure 3 shows three indicative latent factors, where we highlight the venues mostly associated with that factor with high intensity in a word cloud, alongside the temporal profile of that latent factor. In Figure 3(a) we observe venues which are mostly associated with workplaces, and the corresponding temporal profile shows activity which dips during vacation periods such as the summer. Fig 3(b) appears to have clustered various parks and venues related to outdoor activity, with an overwhelming spike during one of the Springs in our dataset, which is a popular season for outdoor activities in NYC. Finally, the last component in Figure 3(c) has identified what appears to be areas where college students congregate and study, with activity that aligns with times during the academic year.

It is important to note here without any external information about different events that coincide with the activity observed, we can only make speculations about the spikes and fluctuations we observe in the temporal activity of different components. The patterns we show in Figure 3 are intuitive and lend themselves to simple explanation by non experts, a fact which lends more credence to the quality of the components extracted. A platform like Foursquare, or different vendors, however, can use such extracted components to understand different spatio-temporal trends over the City, and potentially integrate those components with other information (such as deals, specials, etc) which is not available to us and would provide more insight for time points identified by the decomposition.

## 5 Related Work

In addition to existing tensor works which we reviewed in the Introduction [8, 15], a neighboring problem to ours is the aggregation of temporal edges into a mature graph [23, 24]. However, those works are specific to graphs and do not generalize to arbitrary tensor data. Closest to our work is [19, 20], which provides a greedy solution to our problem. However to the best of our knowledge we are the first work to introduce a principled way to finding an optimal aggregation in the temporal mode using aggregated views of the input data and formalizing it in terms of an optimization problem.


Figure 3: Indicative latent factors for Foursquare

## 6 Conclusions

We developed HARVESTER, a factorization-based tensor temporal granularity estimation, which uses multiple aggregated views of an input tensor and identifies the best aggregation. We extensively evaluate HARVESTER and demonstrate that it generates tensors of superior quality to the current state of the art.

## 7 Acknowledgements

Research was supported by the National Science Foundation under CAREER grant IIS 2046086 and grant IIS-1908070, by the Army Research Office W911NF19-1-0407, and by the U.S. Army Combat Capabilities Development Command Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Combat Capabilities Development Command Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## References


[1] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[2] Faisal M Almutairi, Charilaos I Kanatsoulis, and Nicholas D Sidiropoulos. Tendi: Tensor disaggregation from multiple coarse views. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 867–880. Springer, 2020.

[3] Faisal M Almutairi, Charilaos I Kanatsoulis, and Nicholas D Sidiropoulos. Prema: Principled tensor data recovery from multiple aggregated views. *IEEE Journal of Selected Topics in Signal Processing*, 15(3):535–549, 2021.

[4] Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E Papalexakis, and Danai Koutra. Com2: fast automatic discovery of temporal ('comet') communities. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.

[5] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 3.1. Available online, June 2019.

[6] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in parafac models. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(5):274–286, 2003.

[7] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[8] Alexander Gorovits, Ekta Gujral, Evangelos E Papalexakis, and Petko Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1465–1474, 2018.

[9] Alexander Gorovits, Lin Zhang, Ekta Gujral, Evangelos Papalexakis, and Petko Bogdanov. Mining bursty groups from interaction data. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 596–605, 2021.

[10] Richard A Harshman et al. Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

[11] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 115–124, 2014.

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

[13] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. Higher-order web link analysis using multilinear algebra. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[14] Denis Krompaß, Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Non-negative tensor factorization with rescal. In *Tensor Methods for Machine Learning, ECML workshop*, pages 1–10, 2013.

[15] Taehyung Kwon, Inkyu Park, Dongjin Lee, and Kijung Shin. Slicenstitch: Continuous cp decomposition of sparse tensor streams. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 816–827. IEEE, 2021.

[16] Alexander V Lotov and Kaisa Miettinen. Visualizing the pareto frontier. In *Multiobjective optimization*, pages 213–243. Springer, 2008.

[17] Evangelos E Papalexakis. Automatic unsupervised tensor mining with quality assessment. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 711–719. SIAM, 2016.

[18] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–44, 2016.

[19] Ravdeep Pasricha, Ekta Gujral, and Evangelos Papalexakis. Adaptive granularity in time evolving graphs as tensors. In *Proceedings of the 16th International Workshop on Mining and Learning with Graphs (MLG)*, 2020.

[20] Ravdeep Pasricha, Ekta Gujral, and Evangelos E Papalexakis. Adaptive granularity in tensors: A quest for interpretable structure. *arXiv preprint arXiv:1912.09009*, 2019.

[21] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[22] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[23] Sucheta Soundarajan, Acar Tamersoy, Elias B Khalil, Tina Eliassi-Rad, Duen Horng Chau, Brian Gallagher, and Kevin Roundy. Generating graph snapshots from streaming edge data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 109–110, 2016.

[24] Rajmonda Sulo, Tanya Berger-Wolf, and Robert Grossman. Meaningful selection of temporal resolution for dynamic networks. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs*, pages 127–136, 2010.

[25] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2014.