RecTen: A Recursive Hierarchical Low Rank Tensor Factorization Method to Discover Hierarchical Patterns from Multi-modal Data

Risul Islam [1], Md Omar Faruk Rokon [2], Evangelos E. Papalexakis [3], Michalis Faloutsos [4]
Dept of Computer Science, UC Riverside [1,2,3,4]
risla002@ucr.edu [1], mroko001@ucr.edu [2], epapalex@cs.ucr.edu [3], michalis@cs.ucr.edu [4]

Abstract

How can we expand the tensor decomposition to reveal a hierarchical structure of the multi-modal data in a self-adaptive way? Current tensor decomposition provides only a single layer of clusters. We argue that with the abundance of multi-modal data and time-evolving networks nowadays, the ability to identify emerging hierarchies is important. To this effect, we propose RecTen, a multi-modal hierarchical clustering approach based on tensor decomposition. Our approach enables us to: (a) recursively decompose clusters identified in the previous step, and (b) identify the right conditions for terminating this process. In the absence of a well-established benchmark, we evaluate our approach with synthetic and five real datasets. First, we test the sensitivity of the performance to different scenarios and parameters. Second, we apply RecTen on four online forums and a dataset that represents user interaction on GitHub. This analysis identifies meaningful and interesting behaviors, which further increases our confidence in the usefulness of our approach. For example, we identify some real events like ransomware outbreaks (55 users, 86 threads, December 2015, February 2016), the emergence of a black-market of decryption tools (34 users, 12 threads, February 2016), and romance scamming (82 users, 172 threads, March 2018). To maximize the impact of our work, we intend to: (a) develop a usable tool, (b) make the tool and our datasets publicly available. However, RecTen is a hierarchical clustering approach that can be used to take the pulse of large multi-modal data and let the data reveal its own hidden structures.

Introduction

Tensor decomposition has emerged as a powerful analytical tool with a rich variety of applications, but it focuses on identifying latent clusters without exploring any hierarchical structure that may exist. Tensors generalize the concept of a 2-dimensional matrix into multiple dimensions and we use the term *modes* to refer to these dimensions. On the one hand, current tensor-based approaches do an excellent job of identifying latent patterns in the form of soft clusters (an entity can belong to multiple clusters) and have been used successfully in a wide range of types of data across many disciplines. On the other hand, we argue that often behaviors and phenomena have an inherent hierarchical structure, which can provide interesting insights once it is revealed.

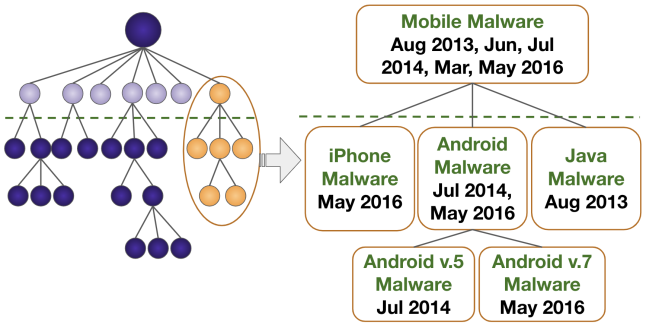


Figure 1: Output from RecTen after applying it on "Hack This Site" security forum data. The inset shows the structure of users and threads that focus on sub-topics of mobile malware, which is the focus of the parent cluster. Our decomposition also discovers the dates when the clusters are most active.

Problem: The main focus of this work is a relatively unexplored question: how can we extend tensor decomposition to identify hierarchies when such hierarchies exist in the data? We are given a multi-dimensional dataset, and we seek to identify clusters and their potentially hierarchical structure present in the data. The challenge here is twofold: (a) we do not know a priori anything about the data, such as the number of clusters or levels, (b) we want to adapt to different levels of "sensitivity" meaning that different parts of the data may hide more layers of hierarchy than others. We want to build on the power of tensor decomposition by expanding it to address the above challenges.

Formally, we state the problem as follows:

Given a Tensor T, how can we expand the decomposition of T in a recursive manner into a hierarchy of clusters, C_i? The input is a tensor and the output is a hierarchy of clusters.

Algorithmically, the problem poses two main challenges: (a) we want to recursively decompose clusters in each layer of the hierarchy, and (b) we want to identify the right conditions for terminating this process.

Motivating case-study: *What applications would benefit* *from such a capability?* Apart from being an interesting theoretical problem, we can think of several applications that could make use of an effective solution. Here we showcase a focused example. A security analyst wants to identify the origin, evolution, and developer interactions for a particular malware. The input data is a large number of online security forums where, surprisingly, emboldened hackers openly sell malicious tools and services, while they boast of and collaborate on malicious acts [Gharibshah, Papalexakis, and Faloutsos2020, Islam et al.2020b]. Each forum can be seen as a three-dimensional tensor with three modes: (a) users, (b) threads, and (c) time. The analyst initially wants a quick birds-eye view of the group activity of the users on each platform. Subsequently, she can focus on the threads that contain keywords/discussion of interest and conduct a more focused analysis, for example, evolution of these group activity in an iterative fashion. Fig. 1 provides a real example of the first step described above for the security forum "Hack This Site", which reveals the interesting and rich underlying structures with 27 clusters and 4 levels of hierarchy. A non-hierarchical approach would report only one level of detail e.g. only the first line of clusters above the green dashed line in the figure. In the *Discussion* section, we elaborate further on the motivation and real-world applications of multimodal hierarchical clustering.

Despite the vast literature on tensor decomposition, we are not aware of any work that fully explores the hierarchical tensor decomposition. We can group prior efforts into three main families: (a) hierarchical clustering in 2D matrices , (b) deep learning-based hierarchical clustering , and (c) non-hierarchical tensor decomposition clustering. We discuss the prior works in detail in the *Related Work* section, while we provide a qualitative comparison of these approaches in Table 2.

Contribution: As our key contribution, we propose RecTen, a hierarchical soft clustering approach based on tensor decomposition. Our approach provides the required mechanisms for recursively decomposing clusters, and for terminating this recursive process. We evaluate our proposed algorithm using both synthetic and real data. We use synthetic data to evaluate the performance given the absence of an established benchmark. We also use this synthetic data to evaluate the sensitivity of three internal parameters, which enables us to provide recommendations for hands-free operation.

a. RecTen compares favorably against the state-of-the-art algorithms. We find that RecTen performs favorably when compared to six other state-of-the-art methods, as shown in Table 3, using our synthetic hierarchical data. Interestingly, RecTen performs well even with flat (non-hierarchical) data as shown in Table 4.

b. RecTen extracts meaningful clusters with real-world data. We apply RecTen on four online forums and a dataset that represents user interactions on GitHub. This provides indirect evidence of the usefulness of RecTen in identifying meaningful clusters, such as tight-knit groups of users, and events. For example, the discovered clusters (27 clusters, across 4 levels of hierarchy) of "Hack This Site" forum in Fig. 1 are meaningful in the sense that they identify communities of special interest and point out the activity peaks in time.

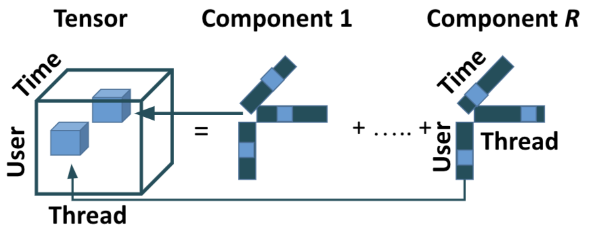


Figure 2: Visualization of tensor decomposition.

A usable open-source platform for maximal impact. As a tangible contribution, we implement RecTen as a powerful user-friendly platform that will be useful to researchers, and practitioners. The platform expects a Tensor as input and produces an output hierarchy of clusters which can be analyzed further to understand the hidden structures. The key advantages of RecTen platform is that it is user-friendly by being both automatic and customizable. RecTen can run with default parameter settings, but savvy end-users can optionally tune the parameters based on their needs and preferences.

We will make both our platform and datasets available that can help establish research benchmarks.

Background and Datasets

We provide some fundamental concepts, a qualitative evaluation, and a description of our datasets.

Tensors and decomposition. Tensor decomposition has been established as a powerful analytical tool. In recent years, it has been the basis for many algorithmic solutions and many practical applications [Kolda and Bader2009, Liu et al.2019, Papalexakis and Doğruöz2015]. The following paragraph discuss the basics of tensor decomposition. An expert reader may skip the paragraph below.

A d-mode tensor [Kolda and Bader2009] is a d-way array (here $d = 3$). So, we call $I \times J \times K$ tensor a "3-mode" tensor where "modes" are the number of dimensions to index the tensor; the "modes" can be $A = \{a_1, a_2, ..., a_I\}, B = \{b_1, b_2, ..., b_J\}$, and $C = \{c_1, c_2, ..., c_K\}$. Each 3D *element/entity* of the tensor, *X(i,j,k)*, captures the interaction of a_i, b_j, and c_k or zero in the absence of any interaction. In a decomposition, we decompose a tensor into *R* rank-one components, where *R* is the rank of the tensor, as shown in Fig. 2. That means, tensor is factorized into a sum of rank-one tensors, specifically, sum of outer products of three vectors (for three modes): $X \approx \sum_{r=1}^{r=R} A(:,r) \circ B(:,r) \circ C(:,r)$ where $A \in \mathbb{R}^{I\times R}$, $B \in \mathbb{R}^{J\times R}$, $C \in \mathbb{R}^{k\times R}$ and the outer product is derived by $(A(:,r) \circ B(:,r) \circ C(:,r))(i,j,k) = A(i,r)B(j,r)C(k,r)$ for all *i, j, k*. Each component represents a latent pattern in the data, and we refer to it as a **cluster**. For example, one such cluster in the "Offensive Community" security forum represents a group of 29 users that are active on the first weekends of July 2016 and discuss "multi-factor authentication failure" in a group of 72 threads. Each cluster is defined by three vectors, one for each dimension, which show the "*Participation Strength*" of each element for that cluster.

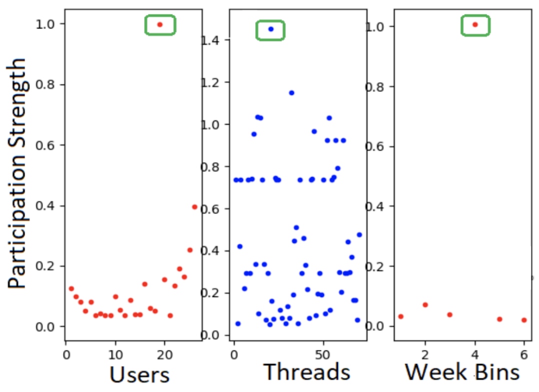


Figure 3: An example of a cluster (28 Users, 70 Threads, 6 Weeks) from the "Offensive Community" forum. The intensity in each vector helps us identify users, threads and time intervals that are "important" for the cluster.

Typically, one considers a threshold to filter out elements that do not exhibit significant *Participation Strength*, as we discuss later.

The algorithmic landscape. We provide a high-level overview of the algorithmic landscape with respect to multi-dimensional hierarchical clustering. We can consider the following families of approaches: (a) 2D matrix methods, (b) Deep Learning-based methods, and (c) tensor-based methods but without hierarchy support. A qualitative analysis is provided in Table 2. In a nutshell, 2D matrix approaches are limited in dimensions, and in our context often miss the extra temporal dimension [Bateni et al.2017, Gharibshah, Papalexakis, and Faloutsos2020]. Deep Learning approaches require large datasets to work well and often the results are less intuitive to interpret and explain [Karim et al.2020]. Finally, to the best of our knowledge, tensor-based approaches so far have not supported hierarchies. A more detailed discussion of previous works is provided in the *Related Work* section.

Datasets

In our evaluation, we consider the following datasets: (a) three security forums, (b) a gaming forum, and (c) a group of GitHub repositories of malware software and their authors. We provide a brief description of these datasets below.

a. Security forum datasets. We collect data from three security forums: Offensive Community (OC), Hack This Site (HTS), and Ethical Hackers (EH) [1] All the forums are in English language. In these forums, *users* initiate discussion *threads* in which other users can *post*. The discussions expand both "white-hat" and "black-hat" skills. The datasets of the security forums span 5 years from 2013 to 2017. Each tuple in each of our datasets contains the following information: user ID, thread ID, post ID, time, and post content.

b. Gaming forum dataset. We consider an online gaming forum, Multi-Player Game Hacking Cheats (MPGH) [2]

Dataset	Users	Threads/ Repositories	Posts	Active Days
Offensive Comm.	5412	3214	23918	1239
Ethical Hacker	5482	3290	22434	1175
Hack This Site	2970	2740	20116	982
MPGH	37001	49343	100001	289
GitHub	7389	8644	-	2225

Table 1: Basic statistics of our datasets.

This is one of the largest online gaming communities with millions of discussions regarding different insider tricks, cheats, strategy, and group formation for different online games. The dataset was collected in 2018 and contains 100K comments of 37K users [Pastrana et al.2018]. The format of each tuple is the same as security forum datasets.

c. GitHub dataset. GitHub platform enables the software developers to create software repositories in order to store, share, and collaborate on projects and provides many social-network-type functionalities.

We define some basic terminology here. We use the term *author* to describe a GitHub user who has created at least one repository. A *malware repository* contains malicious software and a *malware author* owns at least one such repository. Apart from creating a repository, users of GitHub can perform different *types of actions* including *forking, commenting*, and *contributing* to other repositories. Each tuple in the dataset is represented in the following format: malware author ID, malware repository ID, action type, time, and repository content.

We use a dataset of 7389 malware authors and their related 8644 malware repositories, which were identified by prior work [Rokon et al.2020]. This is arguably the largest malware archive of its kind with repositories spanning roughly 11 years.

The basic statistics of the datasets are shown in Table 1.

Our Approach

We present, RecTen, a novel tensor-based multi-step recursive approach that identifies patterns in an unsupervised way. Algorithm 1 provides the high-level pseudo-code of the basic workflow. Fig. 1 provides the sample output of RecTen. Conceptually, our approach works in three steps. First, we decompose a tensor into clusters at level 1. Second, we "perturb" each cluster at the current level which we consider as another tensor to be decomposed further. Third, we have two termination criteria that stop this recursive process. We explain each step below.

Step 1: Tensor-based Clustering

As a first step, we apply CP decomposition on the given input tensor. We provide a quick overview of the challenges and algorithmic choices in the decomposition algorithm below.

a. What is the ideal number of components to target in the decomposition? To answer this question, we use the AutoTen method [Papalexakis2016] and find the rank (R) of the tensor, which points to the ideal number of clusters to be decomposed into. AutoTen attempts to identify the solution that extracts a large-enough number of components while

[1]`Security forums: ethicalhacker.net, hackthissite.org, offensivecommunity.net.`

[2]`Gaming forum: mpgh.net.`

Family	Sample related works	Multi-modal ($\geq$ 3D)	Performance on small data	Interpretability	Detecting hierarchy
2D matrix	[Ward Jr1963]	✗	✓	✓	✓
Deep Learning	[Karim et al.2020]	✓	✗	✗	✓
Tensor Decomposition (Non-hierarchical)	[Islam et al.2020b]	✓	✓	✓	✗
RecTen	This work	✓	✓	✓	✓

Table 2: Overview of the related algorithmic landscape: a qualitative assessment.

maintaining a high core consistency, which is a metric for model appropriateness/goodness.

b. How can we decompose the tensor? We use the non-negative Canonical Polyadic, also known as CANDECOMP/ PARAFAC (CP), decomposition to find the clusters. RecTen achieves this non-negative factorization by adding the non-negative constraint in CP decomposition.

c. How can we strike a balance on cluster size? Each cluster, derived from a 3D tensor, is defined by three vectors whose lengths are equal to a dimension of the tensor as shown in Fig. 2. However, RecTen provides the functionality of having clusters with significant elements only. Therefore, we need a threshold to determine "significant participation in the cluster", which is a common practice for (a) avoiding unreasonably dense clusters [Sapienza, Bessi, and Ferrara2018], (b) enhancing interpretability, and (c) suppressing noise. So, the challenge is to impose this sparsity constraint and eliminate the need for ad-hoc thresholding to find the clusters with only significant entities. Our solution is to add L_1 norm regularization with non-negative CP decomposition. L_1 regularization pushes the small non-zero values towards zero. Therefore, for each vector, we filter out the zero-valued elements and produce clusters with significant elements only. In this way, we eliminate the noisy entities having the least significant contributions in the cluster. The final model that we use for finding the clusters looks like this:

$$\min_{A\geq 0, B\geq 0, C\geq 0} \|X - D\|_F^2 + \lambda(\textstyle\sum_{i,r} |A(i,r)| + \sum_{j,r} |B(j,r)| + \sum_{k,r} |C(k,r)|)$$

where λ is the **Sparsity Regularizer Penalty** and $D = \sum_r A(:,r) \circ B(:,r) \circ C(:,r)$. To find the clusters, we solve the above equation. Since the equation is highly non-convex in nature, we use the well-established Alternating Least Squares (ALS) optimizer as the solver. An example of a cluster after filtering is shown in Fig. 3.

Step 2: Processing for Next Level Decomposition

Having obtained the clusters from the previous level, our goal is to further decompose each cluster, if termination conditions are not met. Intuitively, the idea is to introduce some perturbation in the cluster to help reveal a structure that is currently avoiding detection.

Rank modification. The main mathematical challenge in this phase is to answer the question: "*How can we get the clusters ready to be decomposed further for the next level?*" Recall that the rank of every cluster is 1, which is why these clusters have not already been decomposed. To decompose a cluster at level l, we need the rank of the cluster to be greater than one. We propose to achieve this by introducing a small perturbation in the cluster by zeroing-out some tensor elements, which changes the rank of the cluster to ≥ 1.

We can illustrate the intuition behind this process with the following simple example . The top-level decomposition provides a set of a rank-one clusters. Let us assume that one such cluster contains two smaller "sub-clusters" within it, but there are enough interactions between these sub-clusters so that the top-level decomposition assumes that this is best represented as a single cluster of rank one. By appropriately removing a few connections between the sub-clusters, we can reveal the underlying structure, which will "push" the rank of the cluster to 2. Namely, the two sub-clusters have become sufficiently distinct for the next-level decomposition. We discuss more about this in *Discussion* section along with experimental results.

Choosing the "target elements" to zero-out. The natural next question is: "*Which elements to choose for zeroing out?*". We propose to select the non-zero valued elements stochastically, but with a bias towards elements with low numerical value.

Let's assume C_l to be a rank-one cluster at level l, $l \geq 1$. We get the processed cluster $\hat{C_l}$ with rank, $r >= 1$, in the following manner. We choose a subset of non-zero elements, $C_l(i,j,k)$, stochastically favoring elements with low numerical value. Specifically, the probability of selecting an element is inversely proportional to its numerical value $C_l(i,j,k)$. That means, the higher the element value, $C_l(i,j,k)$, the lower the probability of getting replaced with 0. Formally, for a 3-mode tensor, the probability of selecting an element, e, having value w among the non-zero elements of cluster C_l is given by the formula:

$$P(\text{e is chosen}) = \frac{\frac{1}{w}}{\sum_{\forall z \in C_l} \frac{1}{z}}$$

where z represents each non-zero value in C_l.

Determining the Deletion Percentage, ϵ : The next question that arises is the following: *"How many elements should we zero-out?"*. We introduce the **Deletion Percentage** parameter, ϵ, which determines the percentage of the total non-zero valued elements in the cluster which we zero-out for a cluster. More precisely, we get the ceiling of that number to ensure that it is an integer, but we also never zero-out all non-zero elements. In our *Evaluation* section, we study the sensitivity of our approach to the Deletion percentage parameter. In addition, we discuss whether these perturbations introduce artifacts in our *Discussion* section.

After assigning zero values to the selected elements, we obtain a perturbed cluster, which we will consider for decomposition in the next level as long as it does not meet the termination criteria, which we discuss below.

Step 3: Termination Condition

We stop the recursive procedure of clustering, when one of the two termination conditions is met:

a. Termination condition 1: cluster size. This termination condition suggests that when the cluster/tensor size is relatively small, we do not attempt to further decompose it. Now the obvious question is: "*When do we call a cluster relatively small?*" Note that we use the concept of cluster size to refer to the number of non-zero elements of the cluster. We introduce the **Minimum Cluster Size** parameter, **k**, which determines that: we do not decompose a cluster further if its size is less than k percent of the average size of its sibling clusters at the same level.

Intuitively, this criterion gives us the ability to provide a flexible mechanism to contain the depth of the recursive decomposition. Naturally, there are many different ways to specify such a condition, including a hard size limit. Here, we opted to specify it as a percentage of the sizes of the clusters of the level to allow for some "self-adaptation". We study the effect of this parameter in our *Evaluation* .

b. Termination condition 2: rank=1. Naturally, the recursive factorization cannot continue if the rank of a cluster is one, even after the perturbation. Therefore, the second terminating condition is: after perturbation, if AutoTen returns rank=1, we stop.

Claim 1. In RecTen, zeroing-out a strict subset of non-zero elements from the rank-1 tensor changes the rank to ≥ 1.

We present the intuition behind this claim here. Assume that zeroing-out some non-zero elements of tensor T leads to a new tensor $\dot{T}$ of rank zero. We want to prove $rank(\dot{T}) - rank(T) \geq 0$. By definition, a rank-zero tensor has only zero elements. Thus, $\dot{T}$ should have only zero elements. However, this introduces contradiction. It is not possible to have a zero-tensor as we only zero-out a strict subset of the non-zero elements. Thus, zeroing out can never lead to rank-0 tensor, i.e. $rank(\dot{T}) - rank(T) \nless 0$). Note that formally, we can represent the zeroing-out process using element-wise Hadamard Product between the rank-one tensor, T, and a basis tensor B i.e. $\dot{T} = T. * B$ where B is the basis tensor with elements either set (1) or reset (0).

The description of our approach is likely to generate the following questions to an astute reader:

a. Is the perturbation introducing an artificial hierarchical structure? We answer this question in the *Discussion* section.

b. How sensitive is the performance of the approach to its three main parameters? We answer this question in the *Evaluation* section, where we study the effect of the three parameters: (i) Deletion Percentage, ϵ, (ii) Minimum Cluster Size, k, and (iii) Sparsity Regularizer Penalty, λ.

Algorithm 1:

RecTen (*T*) Algorithm to factorize a Tensor recursively to have hierarchical clusters.

```
Input: Root Tensor, T
Output: Clusters arranged in hierarchical tree format
1  Clusters = [clusters from first level Decomposition of
     given root tensor, T]
2  Final_clusters_tree.insert(T,null)
3  Final_clusters_tree.insert(Clusters,T)
4  while Clusters != empty do
5      Temp_clusters=[]
6      for each C in Clusters do
7          if Termination Condition 1 then
8              continue
9          end
10         Construct $\hat{C}$ by removing some elements
             from $C$
11         if Termination Condition 2 then
12             continue
13         end
14         Decompose $\hat{C}$ into new clusters $C_i$'s
15         Temp_clusters.insert($C_i$'s)
16         Final_clusters_tree.insert($C_i$'s, C)
17     end
18     Clusters=Temp_clusters
19 end
20 return Final_clusters_tree
```

Evaluation

As RecTen is a multi-modal hierarchical soft clustering method, a thorough evaluation is challenging due to: (a) there is a lack of established ground truth, (b) there is not a well-established methodology for generating realistic synthetic data, (c) finding suitable evaluation metrics is non-trivial, and (d) it is not obvious what are the most appropriate reference methods. We present our efforts to address these challenges below.

Synthetic Tensor Construction

To evaluate RecTen, we use flat and hierarchical synthetic 3-mode tensors, which we describe below.

A. D_Flat: a non-hierarchical synthetic tensor. We generate a flat (non-hierarchical) 3-mode tensor for evaluation purpose. The advantage of a synthetic tensor is that they have a well-established ground truth. To stress-test our algorithms, described later, we generate a 3-mode synthetic tensor, D_Flat. The dimension of D_Flat is $300 \times 300 \times 30$, which we find sufficient for our evaluation.

To elaborate, we start from a zero-tensor, Z. Let us consider that Z has three modes A, B, and C, with indices a_i, b_j, and c_k along the modes respectively. Then we insert some clusters in Z in such a way that these clusters are not decomposed into further clusters. We call these clusters flat because they span in level 1 only.

Fig. 4 shows that a total of 21 clusters (3 clusters from each of the 7 groups) have been introduced which forms the ground truth. Some of the clusters "overlap", if they get projected in only two dimensions.

The three-letter notation, e.g. SSD, indicates the mode along which the clusters have similar (S) or different (D)

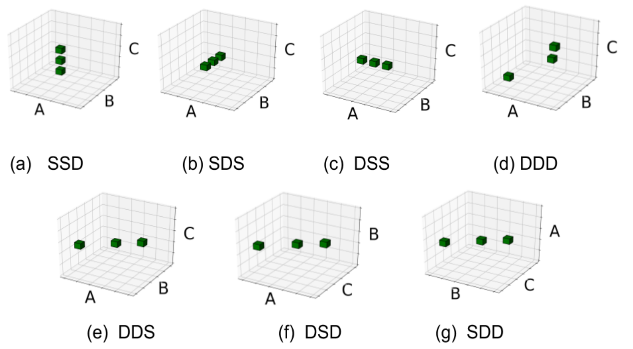


Figure 4: D_Flat: Creation of challenging (overlapping in 2-modes) clusters in our synthetic tensor by combining the depicted 21 clusters.

values in the corresponding dimension. For example, the three inserted SSD clusters in Fig. 4(a) have the same a_is and b_js, but different c_ks meaning that the three clusters contain same members (across A and B modes) but evolve in different times (along C mode).

How do we insert (i.e. add elements to) each cluster? We identify a center for each cluster and then arrange nodes (equivalently, non-zero elements) around that center by finding the position to insert stochastically. We introduce four parameters to control the size, and other properties of these clusters, which we refer to as **Synthetic Cluster Construction Parameters**. The number of nodes per cluster is controlled by the **concentration** parameter ρ while the **cluster radius**, d, determines the radius of the cluster. The value for each element is drawn from a Gaussian distribution, $G(\mu = 10, \sigma = 3)$.

B. D_Hi: a hierarchical synthetic tensor. The goal here is to generate a 3D hierarchical tensor. To do this, we use a two-dimensional Kronecker graphs [Leskovec et al.2010], which have well-defined and controlled hierarchical properties and introduce a third dimension that emulates a temporal evolution partially inspired by previous work [Guigoures, Boullé, and Rossi2012].

Specifically, we create a 2D hierarchical Kronecker matrix, and then we create a series of **slices**: each slice is a slightly modified version of the previous slice. The concentration (stacking) of these slices creates the third dimension that imitates an evolving network.

To elaborate, we first create a hierarchical graph utilizing Kronecker Multiplication of order three (K_3 adjacency matrix) in the base slice (dimension 125X125) demonstrated in Fig. 5. The values of non-zero elements (denoted as x) in this figure are drawn from a Gaussian distribution, $G(\mu = 3, \sigma = 1)$, while all the other elements have a zero value. Thus, the base slice has a two-level hierarchy with 5 clusters at level 1, and each cluster consists of 5 sub-clusters. Second, we create slices, which we limit to 10. For each new slice, we randomly choose $n\%$ of the total data points of the previous slice and assign them new values drawn from the aforementioned Gaussian distribution $G(\mu = 10, \sigma = 3)$. In this way, we create a 125x125x10 tensor with an underlying two-level hierarchical structure.

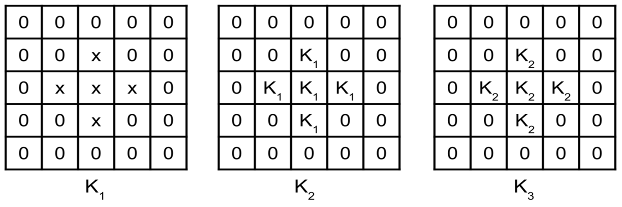


Figure 5: Example of Generation of Kronecker adjacency matrix K_3 for the base slice where $K_3 = K_2 \otimes K_1$ and $K_2 = K_1 \otimes K_1$. $\otimes$ is the Kronecker Multiplication operator.

We discuss the effect of the noise on the performance of RecTen below in this section.

Evaluation Metrics

Evaluating hierarchical multi-modal clustering is challenging as its quality can be analyzed from several different perspectives. For consistency, we adopt metrics from previous methods [Luu2011, Rand1971, Zhang and Shasha1989] which we present below.

A. Total Purity. Total Purity (*TP*) [Luu2011] captures the quality of the clustering and it is measured on a scale of 0 to 1 where *TP=1* indicates perfect clustering. Intuitively, TP represents the percentage of nodes that are associated with the correct cluster and assumes the existence of ground-truth.

B. Rand Index. The Rand Index (RI) [Rand1971] is a measure of similarity between two clustering algorithms on the same data. The metric considers all pairs of elements and counts pairs that are assigned in the same or different clusters by each algorithm. RI has a value within [0,1]. A value of 1 represents identical clustering solutions.

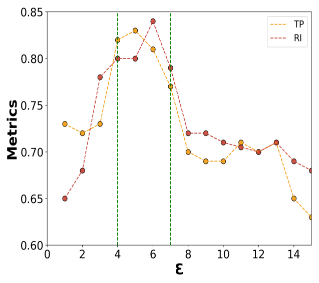
Given a set, S, of n elements and two clustering algorithms, X and Y, to compare, the formula to calculate RI is:

$$RI = \frac{a + b}{\binom{n}{2}}$$

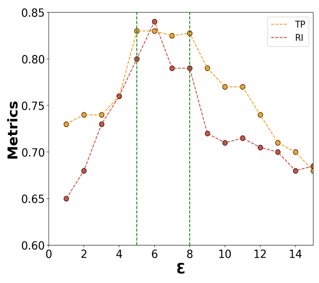
where a is the number of pairs of elements in S that are in the same cluster for X and in the same cluster for Y. b is the number of pairs of elements in S that are in the different clusters for X and in the different clusters for Y. In our case, n denotes the total number of non-zero elements in the synthetic tensor.

C. Tree Edit Distance. The Tree Edit Distance (TED) [Zhang and Shasha1989] measures the similarity of two trees. Here, we can represent a hierarchical clustering by a tree and use this metric. Given that member elements have identities of which cluster it belongs to in the ground truth, we label each node of the tree with the majority members' identity. We use the concept of labeled trees to distinguish between tree nodes.

The TED metric of two labeled trees, T_1 and T_2, is the number of insertion, renaming and deletion operations needed to transform one tree into an exact copy of the other tree. The lower value of the TED, the more similar are the trees. Thus, when compared with ground truth, low values of TED are preferable.

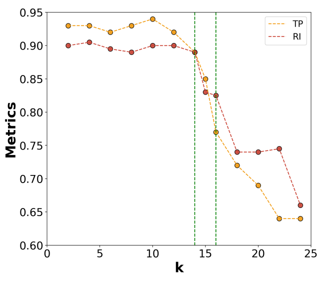


(a) Non-hierarchical data.

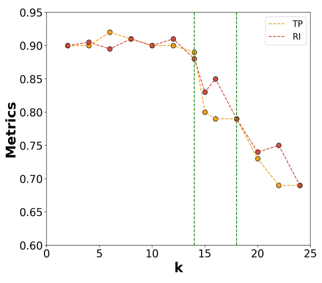


(b) Hierarchical date.

Figure 6: The effect of Deletion Percentage parameter ϵ on clustering quality metrics TP and RI ($k = 15$, $\lambda = 0, n = 10$).



(a) Non-hierarchical data.



(b) Hierarchical data.

Figure 7: The effect of Minimum Cluster Size k parameter on clustering quality metrics TP and RI ($\epsilon = 6$, $\lambda = 0, n = 10$).

The Sensitivity to Algorithmic Parameters

We assess the sensitivity of the performance of our approach to the three algorithmic parameters using Total Purity, and Rand Index metrics.

a. The sensitivity of RecTen to Deletion Percentage, ϵ. Choosing the right ϵ is crucial for RecTen. Very small ϵ may yield less clusters whereas very large ϵ may end up in extracting too many clusters. In both cases, the extracted clusters may not be meaningful. So, the goal is to find a sweet-spot, where we can unravel meaningful patterns with the least amount of deletion. Fig. 6 suggests that $\epsilon \in [4\%, 8\%]$ is our sweet-spot, where we achieve maximum performance based on both metrics, TP and RI, for both non-hierarchical and hierarchical cluster extraction from synthetic tensors. This implies that with reasonable amount of deletion, RecTen is able to extract reliable next level clusters.

b. The sensitivity of RecTen to Minimum Cluster Size, k. Another crucial parameter of RecTen is k which determines when to stop our recursive factorization. Very low k yields in small factorized clusters breaking down the pattern to even more parts (value of performance metrics close to 1) whereas very high value of k will preserve multiple convoluted patterns in a single cluster (value of performance metrics far away from 1). Fig. 7 exhibits that, for our synthetic tensors, both hierarchical and non-hierarchical, $k \in [14\%, 18\%]$ is our sweet region where we achieve reasonably high performance from RecTen based on both TP and RI performance metrics.

c. The sensitivity of RecTen to Sparsity Regularizer Penalty, λ.

The Sparsity Regularizer Penalty parameter, λ, is used to select cluster members during the decomposition as we explained earlier. Low values of λ create larger clusters, while high values create smaller clusters. Clearly, there is a need for a balanced solution that will provide maximal information and insights from the data. Varying the value of the parameter in our study, we find that a value of λ to 0.8 provides the best results w.r.t. the Total Purity metric. The full results are omitted due to space limitations.

Practical guidelines for using RecTen. There are only three algorithmic parameters in RecTen: (a) Deletion Percentage ϵ, (b) Minimum Cluster Size k, and (c) Sparsity Regularizer Penalty λ. Based on the experience from our study, we recommend the following default values for these parameters: $k = 15\%$, $\epsilon = 6\%$, $\lambda = 0.8$. Even a savvy end-user can tune these parameter knobs to tailor them to the needs of their study. To recap, setting ϵ to a high value can enforce extracting higher number of clusters by increasing the rank of a tensor. A much higher value of k will terminate the decomposition for larger sizes, providing a lower bound on the size of the clusters. Furthermore, a high value of λ will affect the size of the clusters during the tensor decomposition. Overall, our results suggest the following ranges for these parameters ϵ within 4%-8%, k within 14%-18%, and $\lambda = 0.8$.

The Sensitivity of RecTen to Dataset Properties

We wanted to evaluate the robustness of our approach to a wide range of data. An advantage of using the synthetic data is that we are able to create a wide range of datasets by varying the some parameters, e.g. Synthetic Cluster Construction Parameters. We briefly discuss their effects below. Unfortunately, the full set of results are omitted due to space limitations.

a. The effect of the amount of noise in the synthetic tensor construction: noise percentage, n. As mentioned earlier, we introduce some noise while creating each new slice while constructing our hierarchical dataset D_Hi. We vary the value of noise percentage, n, to stress-test the resilience of RecTen to this parameter. We find that RecTen can offer reasonable performance (TP=0.796, RI=0.8) for up to $n = 20\%$, at which point we observe a sharp drop in the performance.

b. The effect of the Synthetic Cluster Construction Parameters: d, ρ, μ, σ. We also analyze the performance of RecTen by varying radius d, concentration ρ, and data value distribution parameters μ and σ which affect the generation of our non-hierarchical synthetic tensor. We found that RecTen is relatively robust to different values of these parameters. For example, we found a case where doubling the parameter values did not change the performance significantly. Specifically, RecTen shows TP=0.85 for $d = 4$, $\rho = 8/\text{pattern}$, $\mu = 10$, and $\sigma = 3$, and drops to TP=0.835, when we double the parameter values to $d = 8$, $\rho = 16/\text{pattern}$, $\mu = 20$, and $\sigma = 6$.

We intend to evaluate our algorithm more extensively and with more families of synthetic tensors in future.

Comparison with State-of-the-art Methods

While there is not a widely-accepted set of baseline algorithms for multi-modal hierarchical clustering, we compare RecTen with a set of other widely-used and state-of-the-art which include both 2D hierarchical and 3D non-hierarchical methods. Specifically, we compare against the following approaches: (a) the widely-used basic Ward's method for Agglomerative Hierarchical Clustering Algorithm (AHC_ward) [Ward Jr1963], (b) the frequency-based Agglomerative Hierarchical Clustering Algorithm (AHC_freq) [Madheswaran and others2017], (c) the local cores-based Hierarchical Clustering Algorithm (DLORE-DP) [Cheng, Zhang, and Huang2020], (d) the minimum spanning tree-based Affinity [Bateni et al.2017] clustering (e) the non-hierarchical TenFor [Islam et al.2020b], and (f) the non-hierarchical DynamicT [Sun and Li2019].

Stress-testing: slices and noise. In the evaluation below, we will use the concept of the slice, which we can see as a snapshot in time of an evolving matrix. We introduced the idea of slice when we described the generation of our synthetic data earlier in this section. Recall that, we start with a fairly pristine clustering structure in the base slice and continue stacking slices on top of the base slice. Therefore, a 3D tensor can be viewed as a stack of slices. In each slice, we add some noise, which can be seen abstractly as modifying the cluster members (e.g. reducing the membership strength of an element). The goal is to stress-test the capabilities of RecTen to extract the underlying clusters despite the new modifications.

RecTen outperforms competitions especially in the face of higher noise. Our comparison results suggest that RecTen outperforms other baseline algorithms. Interestingly, the difference becomes more pronounced as we increase the third-dimension, as we explain below. To achieve this, we leverage the concept of slices, as discsussed above. We apply the 2D algorithms on 2D slice i of the synthetic tensors, while the 3D algorithms are applied on the 3D tensor constructed from slices 1 to i. For instance, we produce a hierarchy from slice 5 (2D) using 2D AHC_ward, and produce another hierarchy using RecTen from the 3D sub-tensor constructed by stacking slices 1 to 5. For completeness, we compare with both 2D and 3D algorithms as we explain below. The results from the 2D algorithms are offered *only as a reference point* and not as a head to head comparison. The results are reported in Table 3 for hierarchical data D_Hi and in Table 4 for the non-hierarchical data D_Flat.

The performance of RecTen improves as we consider more slices i in Table 3. For example, we see that the Total Purity increases from 0.82 to 0.86, and the Tree Edit Distance decreases from 5 to 2. This is not surprising: as the third dimension becomes longer, it increases the amount of information that the algorithm can use. We also see that RecTen outperforms the other 3D algorithms, TenFor and DynamicT, with respect to both metrics. For example, for slice 10 in Table 3, RecTen achieves a TP of 0.86, compared to 0.8 and 0.73 for the other algorithms. Similarly, RecTen has TED of 2 compared to 17 and 21. The performance difference of RecTen compared to other 3D methods is statistically significantly ($p > 0.05$).

RecTen performs well even with non-hierarchical data, which suggests that it does not "force" a hierarchy if such hierarchies do not exist in the data. The results are shown in Table 4. RecTen outperforms other methods in terms of both TP and RI (statistical significance $p > 0.05$). Interestingly here the difference between RecTen and TenFor and DynamicT is relatively smaller compared to the difference with hierarchical data.

Application Results and Observations

We provide proof of the effectiveness of RecTen by finding interesting hierarchical clusters from five different real datasets, discussed in *Background and Datasets* section. We utilized the forum analysis tools and NLP techniques [Islam et al.2020b] to profile the clusters and verified that they are meaningful: cohesive and focused on a topic or event.

a. Results from Security forum datasets. Applying RecTen on three security forum datasets reveals interesting clusters.

First, we describe how we constructed the input tensor. We construct a 3D tensor, T, by capturing the interaction of users with different threads at different weekly discretized times. Each element, $T(i, i, k)$, of the input tensor captures the interaction (in terms of the number of posts) of user i with thread j at discretized week *k* or zero in the absence of such interaction. We then fed the input tensor, T, to RecTen. We find a total of 101 clusters from three security forums (41 from OC, 27 from HTS, and 33 from EH) arranged in hierarchical format. Second, we dig deeper into the identified cluster to find out the discussion topics of the clusters. Some of the key results are described below.

We find that RecTen indeed captures meaningful hierarchical clusters from OC, which we validate with prior profiling NLP tools and manual investigation. For instance, a cluster at level 1 revolving around Ransomware related discussion. The discussion started in Dec 2015 and was further instigated in Feb 2016 (observed from the time dimension of the identified cluster) which mirrors the outbreak of SimpleLocker ransomware at that time. The detected cluster actually indicated the early signs of the coming world-wide ransomware disaster. Upon further investigation of that cluster by going one level down, we identified a smaller cluster with 12 threads and 34 sellers of their decryption tools (to recover from the malware) in February 2016. By going into the next level in the hierarchy, we identify that a well-known company, MDS, was also selling the decryption tools in this underground market.

Investigating one of the identified clusters at level 1 from the HTS forum, we identify a group of 32 threads by user *DoSman* offering a free trial of his attack tools. Diving deep into the next level, we find posts related to selling DOS attack tools and phishing tools in April-May 2016.

The EH forum also revealed interesting clusters. For example, we identify *VandaDGod*, an expert Linux hacker (or possibly a group of hackers), who shared a popular tutorial series of hacking in Kali Linux in November 2017. Going one step lower, we find clusters related to 'Hacking into Banks' and 'Hacking Routers'. We find a series of posts to

Baselines	**Slice 4**		**Slice 6**		**Slice 8**		**Slice 10**	
	TP	TED	TP	TED	TP	TED	TP	TED
AHC_ward (2D)	0.79	6	0.75	8	0.7	8	0.65	10
AHC_freq (2D)	0.79	8	0.75	10	0.69	13	0.71	15
DLORE-DP (2D)	0.8	6	0.77	6	0.74	5	0.73	4
Affinity (2D)	0.77	6	0.73	7	0.72	3	0.7	3
TenFor (3D)	0.78	16	0.79	18	0.8	20	0.8	21
DynamicT (3D)	0.71	6	0.72	16	0.73	15	0.73	17
RecTen (3D)	**0.82**	**5**	**0.84**	**4**	**0.84**	**2**	**0.86**	**2**

Table 3: Performance evaluation of RecTen compared to baseline algorithms in terms of Total Purity (TP) and Tree Edit Distance (TED) metrics for hierarchical synthetic data D_Hi. We use bold for the best performance per column.

Baselines	**Slice 4**		**Slice 6**		**Slice 8**		**Slice 10**	
	TP	RI	TP	RI	TP	RI	TP	RI
AHC_ward (2D)	0.79	0.81	0.75	0.77	0.74	0.76	0.7	0.71
AHC_freq (2D)	0.77	0.78	0.75	0.74	0.69	0.73	0.72	0.7
DLORE-DP (2D)	0.79	0.76	0.77	0.76	0.73	0.75	0.73	0.74
Affinity (2D)	0.77	0.76	0.72	0.7	0.71	0.73	0.71	0.73
TenFor (3D)	0.79	0.8	0.81	0.81	0.82	0.82	0.82	0.83
DynamicT (3D)	0.82	0.81	**0.82**	0.82	0.82	0.83	0.81	0.82
RecTen (3D)	**0.83**	**0.82**	**0.82**	**0.84**	**0.83**	**0.84**	**0.85**	**0.87**

Table 4: Performance evaluation of RecTen compared to reference algorithms. We have presented the results of Total Purity and Rand Index metrics for non-hierarchical synthetic data D_Flat.

recruit new members for hacking into banks by *VandaDGod* in the next level. A simple Internet search reveals that extensive and notorious reputation: *VandaTheGod* is accused of hacking several government sites.

We argue that the above results provide a strong indication that RecTen captures meaningful clusters and reveals interesting activities.

b. Results from Gaming forum dataset. We analyze the gaming forum MPGH, and we find a total of 233 clusters from MPGH organized in 6 levels. As expected, we find clusters related to gaming strategies for specific games, but we also found some unexpected clusters. For example, we identify a big cluster at level one revolving around different scamming and hacking-related objections. In the next level, this cluster consists of clusters revolving around online gaming account scamming and 'Romance Scamming'. It seems that in some online games where users can chat among themselves, scammers connect with other users to win their trust and extract money. Also, we identify clusters related to searching for experienced hackers after a painful defeat in a game. These surprising findings indicate that online gaming forums are being a new potential source of security threats.

c. Results from GitHub dataset. Similar to security forums, we construct a 3D tensor for GitHub dataset. Each element, $T(i, j, k)$, of the input tensor captures the interaction (in terms of the total number of create, fork, comment and contribution performed) between: (a) author i, (b) repository j, (c) per week k. Applying RecTen on this tensor, we extract a total of 79 clusters in 4 levels. An interesting cluster contains Windows related malware in the first level and the next level contains Windows related ransomware. These clusters formed mainly in Jan 2016 when ransomware was spreading worldwide and malicious authors started developing more ransomware in GitHub inspired by the attack success. RecTen can help the security enforcement authorities to keep track of which and how malware are being developed and getting popularity over time.

Empirical results comparison with TimeCrunch. TimeCrunch [Shah et al.2015] is also a tensor-based tool to discover patterns, but it extracts only six fixed types of temporal structures, such as near cliques, bipartite cores, and spikes. We apply TimeCrunch on three security forum datasets and find a total of only 17 structures. All these 17 structures are captured in RecTen as well. Moreover, RecTen captures a total of 101 clusters from these security forums. We think that this suggests RecTen strikes a good balance between finding too few and too many clusters of interest. We omit the detailed finding of TimeCrunch due to space limitations.

Computational effort. The computation required by RecTen is not excessive. The average runtime for preparing the final hierarchical Tree view of the biggest forum with 100K posts, MPGH, takes only 2.39 minutes on average whereas TimeCrunch takes 1.98 minutes on average. We reason that TimeCrunch is faster because it operates in single-level decomposition, whereas RecTen performs a recursive multi-level decomposition. In that sense, the computational effort of RecTen seems reasonable. However, we intend to study the scaling properties of our approach with larger datasets. Our experiments were conducted on a laptop with 2.3GHz Intel Core i5 processor and 16GB RAM and the implementation used Python v3.6.3 packages.

Discussion

a. What applications would benefit from multi-modal hierarchical clustering? Apart from being an interesting theoretical problem, we can think of several real-world applications that could benefit greatly from a hierarchical decomposition. Based on our expertise and interest, we consider the dynamics of an online community which includes many hot-button applications such as online fraud, fake news dissemination, online review tampering, and opinion manipulation, as we explain below. As online seems to dominate physical interactions, there is a plethora of online communities, from social networks to discussion forums, and collaboration platforms. Analyzing these communities can provide immensely useful information on who interacts with whom and for what purpose. Specifically, one may want to a) identify groups of misbehaving agents, including cyberbullying and harassment, b) detect online collaboration and collusion, c) detect information spread over time. We argue that any such analysis can greatly benefit by having a capability to automatically provide the interaction landscape through a hierarchy of clusters that represent important activities: events and groups.

More generally, there is a plethora of applications with multi-modal datasets in the financial, and medical sectors that could benefit from an effective multi-modal hierarchical analysis. For example, one could consider the study of drug efficacy over a long period of time, where grouping patients by medical, demographic, and behavioral features, as they evolve over time could illustrate the effects of the drug better. In the *Related Work* section, we provide more examples of such applications.

b. Does RecTen introduce artificial hierarchical structure in the absence of such? A valid concern is whether the perturbation generates hierarchical structure that is not there in the initial data. Although possible, but we argue that under the right conditions, it does not. First, the end-users can control the amount of perturbation, and keeping it reasonably low can minimize the danger. Second, our experimental results on real data suggest that even large clusters are fairly resilient to perturbation for a wide range of perturbations if there is no sub-cluster in those. Indicatively, we can refer to our analysis on the MPGH forum. We find that the biggest cluster (450 users, 530 threads, 23 weeks) discusses about general topics on gaming strategies. We vary the Deletion Percentage, ϵ, from 2% to 15%. For all values, the large cluster was nearly the same each time, and it was never pushed to further decomposition even for high values of the parameter. In fact, we have experimental indication that RecTen produces almost same hierarchy as ground truth for hierarchical data and respects the flatness of non-hierarchical data as well. For instance, experimenting with $k = 15, \epsilon = 6$, and $\lambda = 0.8$ in the non-hierarchical synthetic data, RecTen leads to an average weighted level of 1.14, which is very close to the ideal value of 1 for this case. Although these are already encouraging results, we will investigate this further with different types of real data in the future.

However, in a similar vein, one can question if the perturbation introduces information loss. The answer here depends on how one will use the hierarchical clustering. If it will be used to compress the information, then some information loss will take place. However, if the clustering will only be used to infer an underlying structure (e.g. assign nodes to clusters), the question of information loss becomes less relevant compared to the question as to whether the hierarchical clustering is meaningful. The two arguments we would like to make here that would hopefully close this case are: (i) the deletions are meant to disentangle the lower-level clusters which have enough cross-cutting connections that the top-level decomposition deemed to be a single cluster, (ii) the decomposition is known (as it did in the top level) to impute missing values that are needed in order to represent a cluster as a rank-one entity. Therefore, even if the deletions go a bit further than needed (i.e., start deleting more intra-cluster and less inter-cluster connections), the rank-one modeling is able to complete those deletions, provided that sub-clusters are present and deletion is not extensive.

c. Is our evaluation sufficient given the absence of extensive ground truth? We would have loved to have tested our algorithm against a well-established benchmark. Given its absence, we followed a two-prong approach. First, we evaluate RecTen with synthetic data, where we can know the ground truth, and create a wide range of datasets. In addition, we compare with six different state-of-the-art methods. Second, we resort to manual inspection of our analysis on real datasets. We argue that our evaluation provides sufficient evidence of the overall effectiveness and competitiveness of our method. In the future, we will evaluate our work on more synthetic and real datasets. In addition, we will provide our labeled datasets as a building block towards a community-wide benchmark.

Related Work

To the best of our knowledge, RecTen is the first tensor-based approach that extracts a multilevel hierarchical clustering from multi-modal data recursively. We highlight the most relevant and recent methods, which we group into the following categories.

a. Discovering hierarchical structures without using tensor decomposition. Hierarchical structure discovery is a very common task in data mining. The algorithms that are being used mostly include but not limited to bottom-up Agglomerative Hierarchical Clustering, top-down Hierarchical k-means Clustering, and variations of these algorithms.

The basic and widely used version of Agglomerative Hierarchical Clustering is Ward's method, AHC_ward [Ward Jr1963]. Different variations of bottom-up Hierarchical Clustering are being used recently as well [Teffer, Srinivasan, and Ghosh2019, Mahalakshmi, MuthuSelvi, and Sendhilkumar2020]. DLORE-DP [Cheng, Zhang, and Huang2020] focuses on developing a local cores-based hierarchical algorithm for dataset with complex structures. Another work, AHC_freq [Madheswaran and others2017], proposes an improved frequency-based agglomerative clustering algorithm for detecting distinct clusters on two-dimensional dataset. A recent minimum spanning tree-based bottom-up hierarchical clustering is Affinity [Bateni et al.2017] which works well for extracting structures from graphs. Note that the above-mentioned methods do not applicable for multi-modal data.

Variations of Top-down hierarchical algorithm like hierarchical k-means is also being used in different domains ranging from modeling blast-produced vibration [Nguyen et al.2019] to large graph embedding [Nie, Zhu, and Li2020]. DenPEHC [Xu, Wang, and Deng2016] presents a top-down density peak-based hierarchical clustering method which introduces a grid granulation framework to enable Den-PEHC extract clusters from large-scale and high-dimensional datasets. Recent methods [Roy and Pokutta2017, Charikar and Chatziafratis2017] utilizes the sparsest cut to extract hierarchical clusters for the data. [Kuang and Park2013] focuses on hierarchical document clustering leveraging non-negative matrix factorization. All the above-mentioned algorithms suffer from the same problem. These algorithms are applied on data represented in 2D matrix format. Therefore, finding clusters in multi-modal data requires new strategy. The work of [Guigoures, Boullé, and Rossi2012] focuses on tri-clustering in time-evolving graphs but did not utilize tensor factorization at all.

Different variations of deep learning-based clustering strategies are also prominent. But they basically use the deep neural networks to generate the features and then apply traditional machine learning clustering algorithms to finally compute the clusters [Karim et al.2020, Tian, Zhou, and Guan2017]. However, none of these strategies have ever been applied on multi-modal hierarchical clustering.

b. Advanced tensor decomposition: interactivity and hierarchical data. Although the recursive use of tensor is very rare, a very recent work [Abdali, Shah, and Papalexakis2020] uses a two-level tensor decomposition to detect fake news. Though the authors use the term "hierarchical" for their model, it is only two-levels and combines disparate datsets to achieve its goal. Another work [Wang et al.2015] proposes an interactive framework using tensor decomposition in order to detect topic hierarchy, which differs from the recursive tensor factorization that we do in this study.

c. Tensor decomposition approaches and applications. Tensor decomposition is a well-studied area of research. For our work, we have used CP decomposition but there are other bunch of tensor decomposition approaches. Tucker Decomposition [Kim and Choi2007] is the most well-known of them but it is not capable of generating unique decomposition. There are other tensor clustering approaches but they are applicable in focused domain, for example, tensor graph clustering to detect higher-order cycles [Benson, Gleich, and Leskovec2015], approximation algorithm for 1-d clustering [Jegelka, Sra, and Banerjee2009]. Another recent tensor-based clustering is Dynamic Tensor Clustering (DynamicT) [Sun and Li2019] which works better for dynamic tensors but struggles for general tensors. All of the above-mentioned algorithms suffer from the same general problem of not being hierarchical.

Tensor decomposition has a wide range of applications in diverse domains for categorical data [Islam et al.2020b, Kolda and Bader2009, Liu et al.2019, Papalexakis and Doğruöz2015].

Relatively recently tensor-based techniques have been used in social media analysis. Very recent approaches, TenFor and HackerScope [Islam et al.2020b, Islam et al.2020a], use non-hierarchical tensor decomposition to find interesting events and hackers' dynamics in social media and forums. TimeCrunch [Shah et al.2015] focuses on mining some temporal patterns from time-evolving dynamic graphs. More recent studies [Papalexakis and Doğruöz2015] use tensor to model multilingual social networks in online immigrant communities. Other works [Liu et al.2019, Gujral and Papalexakis2018] use tensor decomposition to study the online communities and their evolution.

Conclusion

We propose, RecTen, an unsupervised tensor-based approach to systematically discover hierarchical clusters in a multi-dimensional space. It can operate parameter-free with default values, but optionally allow parameter tuning for an expert end-user. We show the effectiveness of our approach by an extensive evaluation using both synthetic data and five real datasets.

From an algorithmic point of view, the key advantages of our approach could be summarized in the following points: a) we harness the power of tensor decomposition, b) we provide hierarchies, and (c) we compare favorably to or outperform previous methods. From a practical point of view, RecTen has three attractive features: (a) it operates in an unsupervised way, (b) it generalizes well to both categorical and numerical multi-modal data, and (c) it can operate with default parameters, or customized by a savvy user.

Our work is a step towards a powerful capability, which can allow the data analysts and researchers to mine the wealth of information that exists in massive multi-modal data. Our commitment to providing our tools and data to researchers and practitioners will hopefully amplify the impact of this work.

Acknowledgements

This work was supported by the UC Multicampus-National Lab Collaborative Research and Training (UCNLCRT) award #LFR18548554.

References

Abdali, S.; Shah, N.; and Papalexakis, E. E. 2020. Hijod: Semi-supervised multi-aspect detection of misinformation using hierarchical joint decomposition. *ECML-PKDD*.

Bateni, M.; Behnezhad, S.; Derakhshan, M.; Hajiaghayi, M.; Kiveris, R.; Lattanzi, S.; and Mirrokni, V. 2017. Affinity clustering: Hierarchical clustering at scale. In *Advances in Neural Information Processing Systems*, 6864–6874.

Benson, A. R.; Gleich, D. F.; and Leskovec, J. 2015. Tensor spectral clustering for partitioning higher-order network structures. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 118–126. SIAM.

Charikar, M., and Chatziafratis, V. 2017. Approximate hierarchical clustering via sparsest cut and spreading metrics. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 841–854. SIAM.

Cheng, D.; Zhang, S.; and Huang, J. 2020. Dense members of local cores-based density peaks clustering algorithm. *Knowledge-Based Systems* 105454.

Gharibshah, J.; Papalexakis, E. E.; and Faloutsos, M. 2020. Rest: A thread embedding approach for identifying and classifying user-specified information in security forums. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, 217–228.

Guigoures, R.; Boullé, M.; and Rossi, F. 2012. A triclustering approach for time evolving graphs. In *2012 IEEE 12th International Conference on Data Mining Workshops*, 115–122. IEEE.

Gujral, E., and Papalexakis, E. E. 2018. Smacd: Semi-supervised multi-aspect community detection. In *ICDM*, 702–710. SIAM.

Islam, R.; Rokon, M. O. F.; Darki, A.; and Faloutsos, M. 2020a. Hackerscope: The dynamics of a massive hackeronline ecosystem. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining (ASONAM)*. IEEE/ACM.

Islam, R.; Rokon, M. O. F.; Papalexakis, E. E.; and Faloutsos, M. 2020b. Tenfor: A tensor-based tool to extract interestingevents from security forums. In *Proceedings of International Conference on Advances in Social Network Analysis and Mining (ASONAM)*. IEEE/ACM.

Jegelka, S.; Sra, S.; and Banerjee, A. 2009. Approximation algorithms for tensor clustering. In *International Conference on Algorithmic Learning Theory*, 368–383. Springer.

Karim, M. R.; Beyan, O.; Zappa, A.; Costa, I. G.; Rebholz-Schuhmann, D.; Cochez, M.; and Decker, S. 2020. Deep learning-based clustering approaches for bioinformatics. *Briefings in Bioinformatics*.

Kim, Y.-D., and Choi, S. 2007. Nonnegative tucker decomposition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. IEEE.

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

Kuang, D., and Park, H. 2013. Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 739–747.

Leskovec, J.; Chakrabarti, D.; Kleinberg, J.; Faloutsos, C.; and Ghahramani, Z. 2010. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research* 11(2).

Liu, Y.; Yan, G.; Ye, J.; and Li, Z. 2019. Community evolution based on tensor decomposition. In *ICPCSEE*, 62–75. Springer.

Luu, T. 2011. Approach to evaluating clustering using classification labelled data. Master's thesis, University of Waterloo.

Madheswaran, M., et al. 2017. An improved frequency based agglomerative clustering algorithm for detecting distinct clusters on two dimensional dataset. *Journal of Engineering and Technology Research* 9(4):30–41.

Mahalakshmi, G.; MuthuSelvi, G.; and Sendhilkumar, S. 2020. Gibbs sampled hierarchical dirichlet mixture model based approach for clustering scientific articles. In *Smart Computing Paradigms: New Progresses and Challenges*. Springer. 169–177.

Nguyen, H.; Bui, X.-N.; Tran, Q.-H.; and Mai, N.-L. 2019. A new soft computing model for estimating and controlling blast-produced ground vibration based on hierarchical k-means clustering and cubist algorithms. *Applied Soft Computing* 77:376–386.

Nie, F.; Zhu, W.; and Li, X. 2020. Unsupervised large graph embedding based on balanced and hierarchical k-means. *IEEE Transactions on Knowledge and Data Engineering*.

Papalexakis, E., and Doğruöz, A. S. 2015. Understanding multilingual social networks in online immigrant communities. In *WWW*, 865–870.

Papalexakis, E. E. 2016. Automatic unsupervised tensor mining with quality assessment. In *SDM16*, 711–719. SIAM.

Pastrana, S.; Thomas, D. R.; Hutchings, A.; and Clayton, R. 2018. Crimebb: Enabling cybercrime research on underground forums at scale. In *WWW*, 1845–1854.

Rand, W. M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66(336):846–850.

Rokon, M. O. F.; Islam, R.; Darki, A.; Papalexakis, E. E.; and Faloutsos, M. 2020. Sourcefinder: Finding malware source-code from publicly available repositories. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses*.

Roy, A., and Pokutta, S. 2017. Hierarchical clustering via spreading metrics. *The Journal of Machine Learning Research* 18(1):3077–3111.

Sapienza, A.; Bessi, A.; and Ferrara, E. 2018. Non-negative tensor factorization for human behavioral pattern mining in online games. *Information* 9(3):66. Multidisciplinary Digital Publishing Institute.

Shah, N.; Koutra, D.; Zou, T.; Gallagher, B.; and Faloutsos, C. 2015. Timecrunch: Interpretable dynamic graph summarization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1055–1064.

Sun, W. W., and Li, L. 2019. Dynamic tensor clustering. *Journal of the American Statistical Association* 114(528):1894–1907.

Teffer, D.; Srinivasan, R.; and Ghosh, J. 2019. Adahash: hashing-based scalable, adaptive hierarchical clustering of streaming data on mapreduce frameworks. *International Journal of Data Science and Analytics* 8(3):257–267.

Tian, K.; Zhou, S.; and Guan, J. 2017. Deepcluster: A general clustering framework based on deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 809–825. Springer.

Wang, C.; Liu, X.; Song, Y.; and Han, J. 2015. Towards interactive construction of topical hierarchy: A recursive tensor decomposition approach. In *ACM SIGKDD*, 1225–1234.

Ward Jr, J. H. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58(301):236–244.

Xu, J.; Wang, G.; and Deng, W. 2016. Denpehc: Density peak based efficient hierarchical clustering. *Information Sciences* 373:200–218.

Zhang, K., and Shasha, D. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing* 18(6):1245–1262.