Structural Similarity-Based Matrix Factorization Using Type Extension Trees for Item Cold-Start

Casper Vorm
Department of Computer Science
cvorm16@student.aau.dk

Daniel Drejer Rasmussen
Department of Computer Science
ddra16@student.aau.dk

Thorbjørn Leonard Eilers
Department of Computer Science
teiler16@student.aau.dk

Abstract

Collaborative filtering(CF) is a family approaches that are frequently used in the recommendation domain. It works by recommending items based on similar behaviors among users. There are many CF approaches one of which is matrix factorization(MF) that lets the system discover the underlying characteristics of user-item interactions and is well known for its effectiveness. However in MF, as in most CF approaches the cold-start problem is present, which occurs because of the system's inability to draw any inferences for users or items due to lack of information. This paper proposes a combination of a well documented matrix factorization method: Learning Local Collective Embeddings (LCE)[19], together with a statistical relational learning approach: Type Extension Trees (TETs)[9]. This is done in the form of a pipeline, where we first use the TETs to learn the structural similarity between all items and using the metadata of the items, which can then be described with a item-item similarity matrix. Secondly, the resulting item-item similarity matrix is used to train the LCE model to provide better content based recommendations in item-cold start scenarios. Our approach is evaluated by testing in a cold-start scenario using three different publicly available datasets and comparing the results against other hybrid approaches. The experiments shows a slight increase in effectiveness compared to the original LCE in the item cold-start scenario.

Keywords

Type Extension Tree, Graph mining, Similarity search, Statistical relational learning, Recommender System, Item cold-start, Matrix factorization

1. INTRODUCTION

Recommendation systems are tools for suggesting items to specific users based on their perceived preferences. This is to help users more easily process large amounts of data when interacting with online platforms. There are many different approaches in the recommendation systems domain, but all of these approaches can be broadly classified into two groups: content-based (CB) systems and collaborative filtering (CF) systems. Both systems have their respective advantages and disadvantages. Content-based filtering utilizes the information of items to determine the users preferences by classifying the users likes and dislikes based on an item's features, then use that classification to recommend similar items. In contrast, collaborative filtering utilizes the community preferences as the base for recommendations and ignore the item features. Thus, the recommended items are those that similar users have liked in the past, where similarity between users is based on how many items they have in common.

The cold-start problem is one of the main problems concerning recommender systems, i.e. how to handle the situation when a new item or user is introduced into the system. In this paper, we specifically focus on the item cold-start problem and how to handle it. Collaborative filtering models are affected by this problem because of their dependency on users' item interaction history. On the other hand, Content-based systems do not have this problem as they can utilize items descriptions to provide recommendations. However, purely content-based approaches usually achieve lower accuracy, therefore they are often used in hybrid models.

The item cold-start problem is a considerable problem for recommendation systems, because of two reasons. First, online platforms, such as Amazon or YouTube, have hundreds or thousands of new items daily and effectively recommending these new items is important for keeping user engagement. Secondly, collaborative filtering methods tend to achieve better accuracy in contrast to content-based methods. As such, it is at the core of most state-of-the-art recommender systems [1]. However, collaborative filtering methods require that items are rated by a certain number of users in order for it to provide recommendations with good accuracy. Therefore, it is crucial for collaborative recommender systems to reach such a state quickly, in order for it to produce accurate recommendations for new items.

Ideally, we want a hybrid approach where we can utilize aspects of both collaborative filtering models and content-based models. Doing so, it is important to improve the accuracy of content-based models, or content-based components in hybrid models to optimize for cold-start scenarios. One approach is to utilize different data structures, such as heterogeneous networks e.g. knowledge graphs. Recently,

rich graph techniques have been applied more frequently in recommendation systems as the data representation has been shown to be quite meaningful when looking for similarities. However, many techniques are somewhat limited in their ability to deal with rich graph structures, which is something the field of statistical relational learning has a long history of dealing with. In this paper we propose a statistical relational learning model, inspired by counts-of-counts similarity[9], using Type Extension Trees(TETs) to handle the rich graph structure. In particular, this rich graph structure is used to encode items s.t. we improve the content-based signals, and the overall predictions made in combination with the collaborative signals.

Matrix factorization techniques are widely used in recommendation systems, these were greatly popularised by techniques such as Singular Value Decomposition (SVD)[10] used to approximate the collaborative matrices. These techniques are also being extended, such as Local Collective Embeddings (LCE)[18], a state-of-the-art collective matrix factorization technique. LCE collectively decomposes both the content and collaborative matrices into a common, low dimensional space that still preserves the geometrical structure of the data. Thus, given information about a new item, e.g. the genres of a movie, it can be projected into the common space and we can infer which users are most likely interested in this item. In this way, LCE mitigates the cold-start problem.

In this paper, we propose a combination of methods in the form of a pipeline, first learning the item-item similarities using TETs and feeding this into the LCE model to improve the content-based signals of the model. We call this combination of methods: Structural Similarity Factorization (LCE-SSF). By using the item-item similarities, together with the user-item interactions, we attempt to improve the recommendations in cold-start scenarios. We perform extensive experiments, where we evaluate our model against multiple baselines on the item cold-start scenario. Our contributions in this paper can be summarized as follows:

- We introduce a new method for recommendation, LCE-SSF, that combines content and collaborative information in a unified matrix factorization framework, while exploiting the structural similarities between items by utilizing TETs.
- We perform extensive experiments with LCE and LCE-SSF, and analyze the effects that different inputs, specifically the items, have on the models.
- We contribute the results from our experiments presented in this paper, which shows that our proposed method outperforms LCE on some datasets, and shows that when used as a graph regularization term it consistently shows an increase in performance.

2. RELATED WORK

Type Extension Trees for feature construction and learning in relational domains[8] In this paper, Jaeger et al. introduces the Type Extension Tree(TET) framework. The TET framework is a learning algorithm that through supervised leaning discover complex "counts-of-counts" features, which describe the combinatorial structure of a graph entity's neighborhood. They discovered through their experiments that TETs were able to discover *non-trivial* and *interpretable* count-of-count features. Their findings showed that models that exploit complex count-of-count information outperforms models using flat counts.

Counts-of-counts similarity for prediction and searching relational data[9] This paper is a continuation of the TET paper by Jaeger et al.[8]. They add a class of logistic evaluation functions that transforms a combinatorial counts-of-counts value into a hierarchical numerical structure, which they approximate using a collection of multi-dimensional histograms, and define an earth-mover's-distance based metric on these histograms. The distance is approximated using marginal distances, which in conjunction with a metric-tree supports efficient retrieval of nearest neighbors.

Item Cold-Start Recommendations: Learning Local Collective Embeddings[18] In this paper, Saveski et. al address a common problem in recommender systems, namely the item cold-start scenario, when new items are introduced, but has not been interacted with by any users, then no proper recommendations can be made. To handle this, they introduce *Local Collective Embeddings* or LCE. LCE a matrix factorization model that collectively learns both user-item and item-feature embeddings in a common, low dimensional space. Doing so, the content-based information of a new item can be used to project the item into the common space and infer which users are most likely interested in the new item.

Metadata Embeddings for User and Item Cold-start Recommendations [11] Maciej Kula proposes a hybrid MF model, namely LightFM. This model performs well in cold-start scenarios and when interaction data is sparse, because items and users are presented as linear combinations of their content features' latent factors. The model also performs at least on par with a pure collaborative MF model, where there is lots of interaction data. The features embeddings that the model produces, encode semantic information comparable with how it is done in word embedding approaches.

Learning Attribute-to-Feature Mappings for Cold-Start Recommendations [6] In this paper, Gantner et al. describe their attribute-aware matrix factorization model, that maps user or item attributes to the latent features of a standard MF model. Using these mappings, the factors of a trained, standard MF model can be applied to handle the cold-start problem (either a new-user or a new-item), while still retaining the advantages of standard MF, being speed and predictive accuracy. Furthermore, they introduce their framework MyMediaLite, as a collection of algorithms in the form of a algorithm library for recommendation systems.

3. PROBLEM DEFINITION

We consider the item cold-start recommendation scenario, where we would like to suggest new items, with no interactions, to potentially relevant users. Given a new item q and its feature vector $\boldsymbol{q} \in \mathcal{R}^{1\times f}$, where f is the number of features, we want to calculate its structural similarity with all other items by utilizing TETs, giving us an item-item matrix $\mathbf{X_i} \in \mathcal{R}^{n\times n}$, where n is the number of items. In other

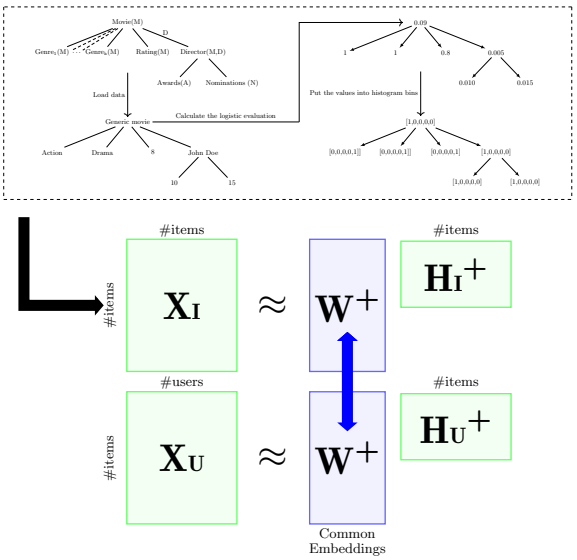


Figure 1: This figure shows a full overview of the model, which can be seen as a pipeline. In the top half, inside the dashed rectangle, we compute the similarities between items based on their TETs. The item-item similarities are fed into the LCE model in form of the item-item similarity matrix $\mathbf{X_I}$ together with the user-item interaction matrix $\mathbf{X_U}$. These matrices are then decomposed into the common, low dimensional space $\mathbf{W}^+$ and the latent spaces $\mathbf{H_I}^+$, $\mathbf{H_U}^+$, for $\mathbf{X_I}$ and $\mathbf{X_U}$ respectively, where $+$ denotes non-negativity.

words, each item in $\mathbf{X_i}$ is described by its similarity with all other items. Then, given the structural similarity with other items and the past activities of users, we want to retrieve the most potentially interested users in this item. This problem can be defined formally as follows. When training the model, we are given two matrices: a collection of n items described by their structural similarity as a item-item similarity matrix $\mathbf{X_i} \in \mathcal{R}^{n\times n}$, and a set of u users stored in a matrix $\mathbf{X_u} \in \mathcal{R}^{n\times u}$, where each cell (i, j) indicates if the user j has shown interest in item i. When testing the model, we are given a new item $\boldsymbol{q}$ with description $\boldsymbol{q_i} \in \mathcal{R}^{1\times n}$, and our goal is to predict $\boldsymbol{q}_u \in \mathcal{R}^{1\times u}$, i.e., to score how likely a user is to show interest in the new item.

4. MODEL

In this section we start by introducing the TETs and how these can be used to represent structural similarity of items by evaluating the data within the TETs. Then we will go over how the items are compared by putting the item representations in histograms and calculating distances between the different histograms. Lastly, we show how we use these distances or similarities in combination with the LCE model.

4.1 TET

The underlying data model of this approach, is that of an attributed, multi-relational graph, which is given by a set of entities (nodes) E, a set of attributes $A = a_1, ..., a_l$ defined on E, and binary relations $R = r_1, ..., r_m$ between the entities. Given the entities $e, e' \in E$ in some data graph, then $a_i(e)$ and $r_j(e, e')$ are ground atomic statements that can be either *true*, *false* or a real number. The atomic statement

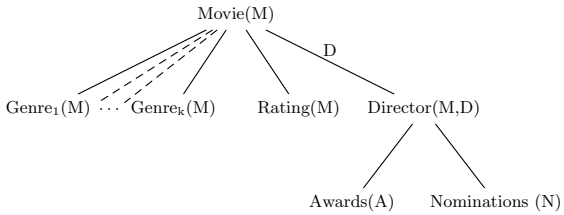


Figure 2: Describes the TET for a movie. The Genre$_k$ and Genre$_1$ describes the $1, 2, ..., k$ genres that a movie can have, which means there are k genre nodes.

are expressed on the form $a_i(X)$ or $r_j(X, Y)$, where X and Y are variable symbols. We use $\alpha(\boldsymbol{X})$ as a generic expression for an atomic statement of either of the two forms.

Example 1. Given a movie m, the movie has k different possible genres $g_1, g_2..., g_k$. Through the ground atomic statement $a_{g_i}(m)$, where g_i denotes one of the k genre attributes for movie m, we get a *true* or *false* value for the given genre g_i. Doing this for all k genre attributes $(a_{g_1}(m), ..., a_{g_k}(m))$ we get the *true* and *false* values related to the movie m. If movie m was simply a drama movie we would get one *true* value and $k - 1$ *false* values.

Definition 1. A type extension tree is a rooted tree whose nodes are labeled with atoms, whose edges can be labeled with variables, and as a special case the leaf nodes can be labeled with numerical atomic relations.

The TET specification, denoted as $T(\boldsymbol{X})$, is the TET notation used for a TET with free variables $\boldsymbol{X} = X_1, ..., X_k$, and corresponding tuples $\boldsymbol{e} = e_1, ..., e_k$ of graph entities. An example of a TET specification for a movie data base is shown in figure 2. The TET specification shows the features for a single entity of type Movie. The TET includes the attribute Movie with the underlying attributes Rating and the k Genre attributes. There is also the relation to the Director node with the underlying Awards and Nominations attributes. The Awards and Nominations nodes are numerical attributes that count the number of awards and nominations that the director has received. The numerical attributes of our TET specification will be feature scaled into the interval: $[0, 1]$, this is done to give a more even distribution when calculating the values of the tree.

4.1.1 *Logistic evaluation function*

To evaluate the TET $T(\boldsymbol{X})$, we need to define the value of the TET features and then use of these values for prediction tasks and similarity measures. This approach is inspired by the approach from *Jaeger et al.* [9].

Definition 2. Let $T(\boldsymbol{X})$ be a TET. A weight assignment $\boldsymbol{\beta}$ for T assigns a nonnegative real number to all non leaf nodes and edges of T. A weight assignment can be written as $(\beta_0^r, \beta_1^r, ..., \beta_m^r, \boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m)$, where β_0^r is the weight assigned to the root and β_i^r is the weight assigned to the edge from the root to the ith child, and $\boldsymbol{\beta}_i$ is the weight assignment to the ith subtree.

The parameters can be either set manually, learned, or be set to default, where the bias parameters β_0 are set to 0, and the weight parameters $\beta_1, ..., \beta_m$ are set to 1.
Given this weight assignment for the TET, we need to define TET values from the TET specification. It is useful to consider a subset of variables from the free variables $\boldsymbol{X}$ denoted with $\boldsymbol{Z} \subseteq \boldsymbol{X}$. The subset $\boldsymbol{Z}$ is used to denote a selection of entities from $\boldsymbol{e}$ such as $\boldsymbol{e}[\boldsymbol{Z}]$, which means we get a selection of the graph entities from the sub-TETs in the subset $\boldsymbol{Z}$, rooted at the atom $\alpha(\boldsymbol{e}[\boldsymbol{Z}])$. This is useful as the TETs are evaluated from the underlying nodes and sub-TETs, making it possible to express the elements of a given TET.

Example 2. Using the subset $\boldsymbol{Z}$, we could imagine that $\boldsymbol{e}$ is a list: $\boldsymbol{e}$ = (Toy Story, Animation, Children's, Comedy, John Lasseter, 8.3). $\boldsymbol{X}$ would then be the index of the list: $\boldsymbol{X} = (0, 1, 2, 3, 4, 5)$. We use $\boldsymbol{e}[\boldsymbol{Z}]$ to get a slice of the graph entities or in this case a slice of the list. If we were only interested in the movie and the genres we would have: $\boldsymbol{Z} = (0, 1, 2, 3)$, which would result in $\boldsymbol{e}[\boldsymbol{Z}]$ = (Toy Story, Animation, Children's, Comedy).

In the following definition, we show the evaluation of a given TET specification. The evaluated tree will be a tree of numbers denoted by $V(T(\boldsymbol{e}))$. The values at the nodes of the tree are given by the recursions of the logistic evaluation of the tree. This is calculated when we calculate the value at the root node denoted by $\gamma^{\boldsymbol{\beta}}$. The recursion works by following the base-cases given for the tree $V(T(\boldsymbol{e}))$.

Definition 3. Let $T(\boldsymbol{X})$ be a TET with free variables $\boldsymbol{X} = X_1, ..., X_k$. Let $\boldsymbol{e} = e_1, ..., e_k$ be a $k-$tuple of graph entities. The value of the TET feature $T(\boldsymbol{X})$ for $\boldsymbol{e}$, denoted $V(T(\boldsymbol{e}))$ is defined as:

(i) If $T(\boldsymbol{X})$ consists of the single node $\alpha(\boldsymbol{X})$, then $V(T(\boldsymbol{e})) = \alpha(\boldsymbol{e}) \in \{$*true, false,* $\mathbb{R}\}$, where *true* evaluates to 1, and *false* evaluates to 0 (leaves are evaluated by the Boolean or numerical value of their atom).

(ii) Let $\alpha(\boldsymbol{Z})(\boldsymbol{Z} \subseteq \boldsymbol{X})$ be the atom at the root of T.

- If $\alpha(e[\boldsymbol{Z}]) =$ *false*, then $V(T(\boldsymbol{e})) = 0$ and then there is no further recursive evaluation of the TET.
- If $\alpha(e[\boldsymbol{Z}]) =$ *true* that means that TET $T(\boldsymbol{X})$ has $m \geq 1$ children that are roots of sub-trees $T_h(\boldsymbol{Z}_h)$ with free variables $\boldsymbol{Z}_h (h = 1, ..., m)$. $V(T(\boldsymbol{e}))$ is then defined as a tuple of real numbers $(\mu_1, ..., \mu_m)$ where μ_h is the value given at sub-tree T_h. To define μ_h given at the root of sub-tree $T_h(\boldsymbol{Z}_h)$ we distinguish two cases:
  - The edge leading to T_h is not labeled by a variable; then $\boldsymbol{Z}_h \subseteq \boldsymbol{X}$, and we define:
$$\mu_h = V(T_h(\boldsymbol{e}[\boldsymbol{Z}_h])) \quad (1)$$
  - the edge leading to T_h is labeled by a variable Y_h; then $\boldsymbol{Z}_h = (\tilde{\boldsymbol{Z}}_h, Y_h)$ for some $\tilde{\boldsymbol{Z}}_h \subseteq \boldsymbol{X}$, and we define:
$$\mu_h = \{V(T_h(\boldsymbol{e}[\tilde{\boldsymbol{Z}}_h], e')) \,|\, e' \in E\} \quad (2)$$
This is to be understood as a multiset of real numbers, counting the multiplicities of identical values obtained for different $e' \in E$.

We simplify the notation of the value tree $V(T(\boldsymbol{e}))$ to γ. So consider $\gamma = (\mu_1, ..., \mu_m)$, with real numbers μ_h of sub-trees T_h, we calculate γ using the logistic evaluation function and weight assignment $\boldsymbol{\beta}$, defined:

$$\gamma^{\boldsymbol{\beta}} := \sigma\left(\beta_0^r + \sum_{i=1}^{m} \beta_h^r \sum_{\gamma' \in \mu_h} \gamma'^{\boldsymbol{\beta}_h}\right)$$

where σ is the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$.

Through the logistic evaluation function, we calculate the root node $\gamma^{\boldsymbol{\beta}}$ and from that we calculate the value tree $V(T(\boldsymbol{e}))$, which is a nested multiset structure of real numbers. For the function to calculate the value at the root node $\gamma^{\boldsymbol{\beta}}$, it first needs the values at $\gamma'^{\boldsymbol{\beta}_h}$, which are the values for the h different sub trees. The logistic evaluation function takes recursive steps for all h children, when leaf nodes are reached, then base case (i) applies and the value at the leaf nodes are used to calculate the values at the non-leaf nodes. An example of going from TET specification(2), with movie data applied, to the evaluated tree $V(T(\boldsymbol{e}))$ can be seen on figure 3.

Example 3. Looking at figure 3. We can see the calculation for the TET value at the root node $\gamma^{\boldsymbol{\beta}}$. The values are calculated as follows: firstly there is only a single child node that isn't a leaf node, as leaf nodes are directly evaluated by their atomic value, as shown by base-case (i) in definition 2, no calculation is necessary for these nodes. The director node however needs to be calculated. In the TET specification there is a variable on the edge going to the node, therefore we consider equation 2. In def 2. This gives us $e' \in$ *Lana, Lilly*, which means there are two director nodes in the tree denoted by the D variable on the edge. We get *(The Matrix, Lana)*= $\sigma(-3+1\cdot(0.064+0.09)) = 0.055$ and *(The Matrix, Lilly)*= $\sigma(-3+1\cdot(0.059+0.09)) = 0.054$. The rest of the child nodes to the root node *(The Matrix)* are all leaf nodes. From this the root node can be calculated *(The Matrix)*= $\sigma(-3+1\cdot(1+1+0.87+0.055+0.054)) = 0.49$, which is equivalent to the full calculation of $\gamma^{\boldsymbol{\beta}}$ if we fold-out all the calculations: $\gamma^{\boldsymbol{\beta}} = \sigma(-3+1\cdot(1+1+0.87+(\sigma(-3+1\cdot(0.064+0.09)))+(\sigma(-3+1\cdot(0.059+0.09))))))$.

4.1.2 Histograms

The logistic evaluation function produces the tree $V(T(\boldsymbol{e}))$, which is a nested multiset structure. The tree contains a detailed description of the graph entities $\boldsymbol{e}$. The goal is to use these descriptions to define distances between entities $\boldsymbol{e}, \boldsymbol{e}'$, in other words finding the similarity between entities. However, in large and dense graphs the full structures of $V(T(\boldsymbol{e}))$ will become very large and therefore not suitable for efficient distance computation, or for storing the pre-computed $V(T(\boldsymbol{e}))$ for all $\boldsymbol{e}$. In order to alleviate this problem, we introduce a collection of 1-dimensional histograms as an approximate representation of the distribution of $V(T(\boldsymbol{e}))$. The histograms allow for fast distance computations and the approximation will be constant in size for all $\boldsymbol{e}$, which is independent from the size of the graph.

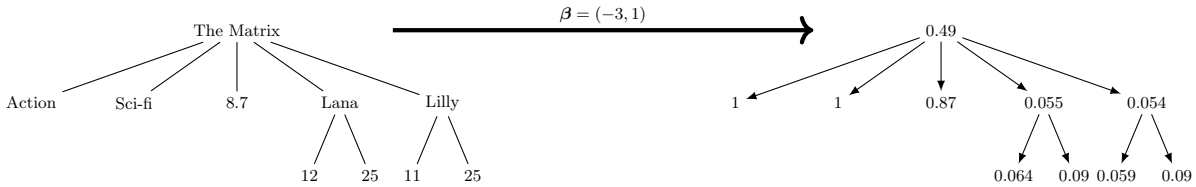


Figure 3: The leftmost figure shows a movie with its features out into the TET specification seen on 2 and the big bold arrow denotes the weight $\boldsymbol{\beta}$ being assigned to T used to calculate the TET value $T(V(\boldsymbol{e}))$.

The idea is to create Node Histogram Trees (NHTs) for each $T(\boldsymbol{X})$ as an approximation for the logistic value tree $V(T(\boldsymbol{e}))$. This is achieved by arranging the node histograms for all nodes of T isomorphic to T, meaning that the structure of the trees are identical.

In the following definition, we define how the nodes in the TET value tree $V(T(\boldsymbol{e}))$ gets transformed into nodes of a node histogram tree. This is done by having a histogram for each node in the TET specification, where each histogram consists of a predefined number of equal size bins. The value or values of a given node is then put into the corresponding bin interval.

Definition 4. Let $\boldsymbol{\alpha} = \alpha_1 ... \alpha_k$ be the set of all atoms for a given TET T i.e. every node of T. Let $V_i = V(T(\alpha_i))$, where $\alpha_i \in \boldsymbol{\alpha}$, be the value of the TET feature $T(\boldsymbol{X})$ for α_i. Let B be the number of bins, obtained by partitioning the interval $[0, 1]$ into B equal width bins. A histogram is then a function, m_i, that counts the number of values in V_i for atoms $\alpha_1 ... \alpha_k$ that fall into each bin s.t. $n_{\alpha_i} = \sum_{i=1}^{B} m_i$ is a histogram of values V_i for a single atom α_i.

Example 4. In figure 4 on the top we see three TET value trees ($V(T(\boldsymbol{e}))$) for three different movies (M1, M2, M3). Each movie has the same TET structure as the one depicted in figure 2. The values are put into histograms and there is one histogram for each node, as denoted by $\boldsymbol{\alpha}$ in definition 4. Given the TET specification in Fig 2 there would be a histogram for each node (histograms for false nodes e.g. $g_1 = \textit{false}$ is not shown, but it is implied. This means that M2 is still isomorphic with M1 and M3, despite the additional node). The B bins are intervals of equal width, $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$. Creating the NHTs is straightforward, on the figure we can see the director node has a free variable and M1 has two directors, while M2 and M3 have only a single director. So using function m_i for the director node we count the entries in each bin and get the histograms: $m_i(0.055, 0.054) \rightarrow [2, 0, 0, 0, 0]$ for M1, $m_i(0.22) \rightarrow [0, 1, 0, 0, 0]$ for M2 and $m_i(0.09) \rightarrow [1, 0, 0, 0, 0]$ for M3.

Now we need to define a metric on NHTs. There are many choices for such a metric, however it makes most sense to follow the definition from *Jaeger et al.* [9], where we simplify the histograms. It is based upon the Earth Mover's distance (EMD)([16],[14]), a well-established metric on histograms for measuring the distance between two probability distributions. It should also be scale invariant, e.g. two users who have rated 1 and 10 similar movies, should be less similar than two users who have rated 101 and 110 similar movies, but could be just as similar as two users who have rated 10 and 100 similar movies.

4.1.3 Node histogram metric

In order to define a metric between NHTs, we must first consider a metric for individual histograms. This is where we have to incorporate EMD. EMD is most naturally defined on histograms that have an equal amount of mass i.e. counts. This is usually normalized, such that we get a probability distribution over the histogram bins.

Let h_1, h_2 be two 1-dimensional histograms that have the same number of bins, then the *relative count distance* between h_1 and h_2 is:

$$d_{r-count}(h_1, h_2) := 1 - \frac{min(c(h_1), c(h_2))}{\sqrt{c(h_1) * c(h_2)}} \tag{3}$$

Where h denotes individual histograms, $c(h)$ is the sum of all cell counts in h. This requires than $c(h_1)$ and $c(h_2)$ are both non-zero. We define $d_{r-count}(h_1, h_2) = 1$ if either $c(h_1)$ or $c(h_2)$ is zero. Otherwise, we define $d_{r-count}(h_1, h_2) = 0$ if both $c(h_1)$ and $c(h_2)$ is zero. We omit the proof that $d_{r-count}$ is a proper, scale-invariant metric, as shown in *Jaeger et al.* [9].

Now, let $\bar{h} = h/c(h)$ be the probability distribution on histogram bins obtained by normalizing h. EMD between these normalized histograms is defined in terms of some underlying ground distance between histogram bins (i.e. how much work must be done to turn one histogram, or "pile of dirt", into the other). We choose Manhattan distance as our metric: it is a commonly used distance metric for histogram bins. We restrict ourselves to only using 1-dimensional histograms. It's a benefit in terms of computational complexity and in [9], 1-dimensional slices in the multidimensional histograms are encountered frequently, which may also indicate that having multidimensional histograms are unnecessary. EMD can be computed by solving an instance of the transportation problem, using any algorithm for the minimum flow cost problem. However, as a special case if we only consider 1D-histograms, the EMD can be efficiently computed by simply keeping track of how much "dirt" that needs to be moved between consecutive bins[21]:

$$\begin{aligned} EMD_0 &= 0 \\ EMD_{i+1} &= A_i + EMD_i - B_i \\ TotalDistance &= \sum_{i=0}^{n} |EMD_i| \end{aligned} \tag{4}$$

Where A_i and B_i are the indices in the respective arrays

representing the 1D-histograms. combining both 3 and 4 into a new distance, via a simple mixture, we define:

$$d_{c-emd}(h_1, h_2) := \frac{1}{2}(d_{r-count}(h_1, h_2) + d_{emd}(\bar{h_1}, \bar{h_2})) \quad (5)$$

This definition gives equal weight to both 3 and 4, and we shall use it to calculate the distance between histogram trees.

4.2 Local Collective Embeddings

Given the problem definition in section 3, items are associated with a set of features and a set of users who interacted with them. Furthermore, using the TETs for structural similarity, each item is described by its similarity to every other item, based on the metric described in section 4.1.3. User-item interactions and item-item similarity are represented by two matrices, the item matrix $\mathbf{X_i} \in \mathcal{R}^{n \times n}$ and an item-user matrix $\mathbf{X_u} \in \mathcal{R}^{n \times u}$, where n is the number of items and u is the number of users. Using the same approach presented in LCE[18], both items and users should be represented in a common latent space by factorizing $\mathbf{X_i}$ and $\mathbf{X_u}$ collectively while enforcing a low-dimensional representation in a common space. In other words, each factor can be described by a set of items but also by a set of users. Given the matrices $\mathbf{X_i}$ and $\mathbf{X_u}$ we can formally define this as the following optimization problem:

$$\begin{aligned} \min: J = \frac{1}{2}[\zeta||\mathbf{X_i} - \mathbf{W}\mathbf{H_i}||_F^2 + (1-\zeta)||\mathbf{X_u} - \mathbf{W}\mathbf{H_u}||_F^2 \\ +\lambda(||\mathbf{W}||_F^2 + ||\mathbf{H_i}||_F^2 + \mathbf{H_u}||_F^2] \\ s.t. \quad \mathbf{W} \geq 0, \mathbf{H_s} \geq 0, \mathbf{H_u} \geq 0 \end{aligned} \quad (6)$$

The first two terms show the factorization of item-item matrix $\mathbf{X_i}$, followed by user-item matrix $\mathbf{X_u}$. Finding the common latent space representation for matrices $\mathbf{X_i}$ and $\mathbf{X_u}$ is done with the same matrix $\mathbf{W}$ for both decompositions. This leads to the following factorizations: $\mathbf{X_i} \approx \mathbf{W}\mathbf{H_i}$ and $\mathbf{X_u} \approx \mathbf{W}\mathbf{H_u}$. $\mathbf{W}$ is a matrix with k latent factors st. $\mathbf{W} \in \mathcal{R}^{n \times k}$. Similarly, $\mathbf{H_i}$ and $\mathbf{H_u}$ are matrices with the n items' latent factors and u users' latent factors respectively, st. $\mathbf{H_i} \in \mathcal{R}^{k \times n}$ and $\mathbf{H_u} \in \mathcal{R}^{k \times u}$. These are the matrices depicted in figure 1. The hyper-parameter $\zeta \in [0, 1]$ controls the importance of the factorization of $\mathbf{X_i}$ and $\mathbf{X_u}$ respectively. Setting $\zeta = 0.5$ gives the same importance to the factorization of both $\mathbf{X_i}$ and $\mathbf{X_u}$. Values of $\zeta > 0.5$ gives greater importance to the factorization of $\mathbf{X_i}$, while $\zeta < 0.5$ gives more importance to $\mathbf{X_u}$. The last three terms are Tikhonov (Frobenius norm) regularization of the three matrices $\mathbf{W}$, $\mathbf{H_u}$, and $\mathbf{H_i}$. This is controlled by the hyper-parameter $\lambda \geq 0$ and is used to enforce smoothness of the solution while also avoiding overfitting.

4.3 Graph Regularization

We attempt to find a common low-dimensional space, using collective factorization as in Eq 6, that is optimized for the linear approximation of the data. We refer to the original paper [18] for a detailed description of the assumptions made regarding data distribution and how it can be used to discover a better low-dimensional space. The main assumption is the manifold assumption, which is that if two data points x_i and x_j are close in the original space, then their low-dimensional representations should also be close together in the latent space. As such, the idea is to model the local geometric structure by using k nearest neighbours on the data points, s.t. we connect the nearest neighbors to the corresponding data points. These connections between data points may be binary, or a scalar value, and results in a matrix $\mathbf{A}$ which can be used to measure the local closeness between two points x_i, x_j. The collective factorization, $\mathbf{W}$, maps data points x_i into low-dimensional representations w_i, where w_i refers to a row of $\mathbf{W}$. We measure the distance between points in the latent space using the Euclidean distance: $||w_i - w_j||^2$. In conjunction with matrix $\mathbf{A}$, which is built using $\mathbf{X_i}$ (where $\mathbf{X_i}$ can be constructed using TETs or some other similarity measure, e.g. cosine similarity), we can measure the local smoothness of the low level representations as follows:

$$S = \frac{1}{2}\sum_{i,j=1}^{n} ||w_i - w_j||^2 \mathbf{A}_{ij}$$

$$= \sum_{i=1}^{n}(w_i^T w_i)\mathbf{D}_{ii} - \sum_{i,j=1}^{n}(w_i^T w_j)\mathbf{A}_{ij}$$

$$= \mathrm{Tr}(\mathbf{W}^T\mathbf{D}\mathbf{W}) - \mathrm{Tr}(\mathbf{W}^T\mathbf{A}\mathbf{W}) = \mathrm{Tr}(\mathbf{W}^T\mathbf{L}\mathbf{W})$$

where $\mathbf{D}$ is a diagonal matrix which entries consist of the sums of the rows or columns of $\mathbf{A}$ (as $\mathbf{A}$ is symmetric), $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the Laplacian matrix of the graph and Tr() is the trace operator.

4.4 Optimization problem

We modify equation 6, with the term given above: $\mathrm{Tr}(\mathbf{W}^T\mathbf{L}\mathbf{W})$, s.t. it enforces locality, giving the following optimization problem:

$$\begin{aligned} \min: J = \frac{1}{2}[\zeta||\mathbf{X_i} - \mathbf{W}\mathbf{H_i}||_F^2 + (1-\zeta)||\mathbf{X_u} - \mathbf{W}\mathbf{H_u}||_F^2 \\ \varphi\mathrm{Tr}(\mathbf{W}^T\mathbf{L}\mathbf{W}) + \lambda(||\mathbf{W}||_F^2 + ||\mathbf{H_i}||_F^2 + \mathbf{H_u}||_F^2] \\ s.t. \quad \mathbf{W} \geq 0, \mathbf{H_s} \geq 0, \mathbf{H_u} \geq 0 \end{aligned} \quad (7)$$

where we introduce φ as a hyper-parameter that controls the degree of locality enforcement. In other words graph regularization, meaning $\varphi = 0$ would imply that there is no graph regularization.

We derive an iterative algorithm based on multiplicative update rules, as it is unrealistic to find the global minimum, because the optimization problem in Eq. 7 is non-convex in terms of the parameters $\mathbf{W}$, $\mathbf{H_i}$, $\mathbf{H_u}$. The partial derivatives of J with regards to $\mathbf{W}$, $\mathbf{H_i}$, $\mathbf{H_u}$ are computed as:

$$\begin{aligned} \nabla_{\mathbf{W}} J = \zeta\mathbf{W}\mathbf{H_i}\mathbf{H_i}^T - \zeta\mathbf{X_i}\mathbf{H_i}^T + (1-\zeta)\mathbf{W}\mathbf{H_u}\mathbf{H_u}^T - \\ (1-\zeta)\mathbf{X_u}\mathbf{H_u}^T + \varphi\mathbf{L}\mathbf{W} + \lambda\mathbf{W} \end{aligned} \quad (8)$$

$$\nabla_{\mathbf{H_i}} J = \zeta\mathbf{W}^T\mathbf{W}\mathbf{H_i} - \zeta\mathbf{W}^T\mathbf{X_i} + \lambda\mathbf{H_i} \quad (9)$$

$$\nabla_{\mathbf{H_u}} J = (1-\zeta)\mathbf{W}^T\mathbf{W}\mathbf{H_u} - (1-\zeta)\mathbf{W}^T\mathbf{X_u} + \lambda\mathbf{H_u} \quad (10)$$

We derive the following first-order conditions by applying the Karush-Kuhn-Tucker (KKT)[5] first-order optimally conditions to J. The KKT conditions is a way of specifying the first-order conditions for our constrained optimization problem with non-negativity constraints i.e. we minimize J with respect to these constraints. This results in a solution space

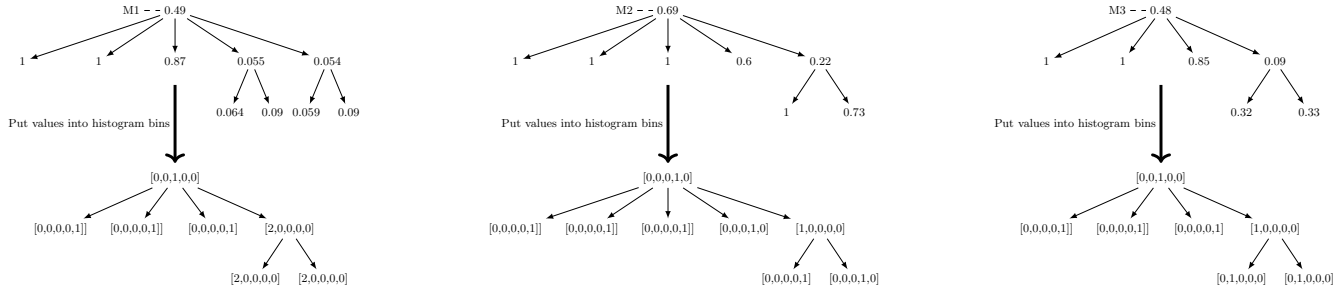


Figure 4: This figure shows the TET value trees for three different movies (M1, M2, M3) and how these values are put into histogram bins of the corresponding node histogram trees.

of J s.t. the following equality and inequality constraints are satisfied. The first condition we derive is non-negativity for the three matrices $\mathbf{W}, \mathbf{H_i}, \mathbf{H_u}$.

$$\mathbf{W} \geq 0, \mathbf{H_i} \geq 0, \mathbf{H_u} \geq 0, \tag{11}$$

The second condition specifies that the gradient for $\mathbf{W}, \mathbf{H_i}$, $\mathbf{H_u}$ must be non-negative. Ideally, we want to find a saddle point that is the global minimum s.t. $\nabla_{\mathbf{W}} J, \nabla_{\mathbf{H_i}} J, \nabla_{\mathbf{H_u}} J = 0$, but this is not realistic in practice.

$$\nabla_{\mathbf{W}} J \geq 0, \nabla_{\mathbf{H_i}} J \geq 0, \nabla_{\mathbf{H_u}} J \geq 0, \tag{12}$$

$$\mathbf{W} \odot \nabla_{\mathbf{W}} J = 0, \mathbf{H_i} \odot \nabla_{\mathbf{H_i}} J = 0, \mathbf{H_u} \odot \nabla_{\mathbf{H_u}} J = 0, \tag{13}$$

where $\odot$ is the element-wise matrix multiplication operator. Substituting the derivatives of J from Equations 8, 9, and 10 in Equation 13, leads to the following update rules:

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{[\zeta \mathbf{X_i} \mathbf{H_i}^T + (1-\zeta)\mathbf{X_u}\mathbf{H_u}^T + \varphi \mathbf{AW}]}{\zeta \mathbf{W}\mathbf{H_i}\mathbf{H_i}^T + (1-\zeta)\mathbf{W}\mathbf{H_u}\mathbf{H_u}^T + \varphi \mathbf{DW} + \lambda \mathbf{W}} \tag{14}$$

$$\mathbf{H_i} \leftarrow \mathbf{H_i} \odot \frac{[\zeta \mathbf{W}^T \mathbf{X_i}]}{[\zeta \mathbf{W}^T \mathbf{W} \mathbf{H_i} + \lambda \mathbf{H_i}]} \tag{15}$$

$$\mathbf{H_u} \leftarrow \mathbf{H_u} \odot \frac{[\zeta \mathbf{W}^T \mathbf{X_u}]}{[\zeta \mathbf{W}^T \mathbf{W} \mathbf{H_u} + \lambda \mathbf{H_u}]} \tag{16}$$

where $\div$ denotes the element-wise matrix division operator. The update rules for $\mathbf{H_i}$ and $\mathbf{H_u}$ are identical to the ones in the original Non-negative matrix factorization (NMF) formulation [12]. The update rule for $\mathbf{W}$ is derived in [17].

4.5 Inference

After the model has been trained, and we have obtained $\mathbf{W}$, $\mathbf{H_i}$, $\mathbf{H_u}$, we can utilize these factors for prediction. For example, given the similarity vector of a new item $\mathbf{q}_i$, we can predict the users that are most likely to interact with it, i.e. $\mathbf{q}_u$. To utilize the factors for prediction, we project the item vector $\mathbf{q}_i$ into the common latent space $\mathbf{W}$ by solving the overdetermined system $\mathbf{q}_i = \mathbf{w}\mathbf{H}_i$ using the least squares method (i.e. the vector $\mathbf{q}_i^{1\times k}$ has smaller dimension than $\mathbf{H_u} \in \mathcal{R}^{k \times n}$). Then vector $\mathbf{w}$ is used to capture the factors from the common latent space that explain the observed item $\mathbf{q}_i$, and can be used to infer the missing part of the query: $\mathbf{q}_u \leftarrow \mathbf{w}\mathbf{H_u}$. We can then rank users, as each element of $\mathbf{q}_u$ represents a score of how likely it is that the user will interact with the new item.

5. EXPERIMENTS

In this section we present our extensive experiments to evaluate the performance of our model on the item cold-start scenario to answer the following research questions: (**RQ1**) How does LCE-SSF compare against state-of-the-art approaches and how does LCE benefit from TETs? (**RQ2**) How does LCE benefit from TET induced similarity matrices as a graph regularization term. (**RQ3**) How does the structural similarity of TETs compare to other known similarities in terms of being utilized as a graph regularization term and as an input matrix?

5.1 Experimental Setup

5.1.1 Dataset

We conduct experiments on three datasets, MovieLens^{1M}, MovieLens^{100K}, and BX-Books from publicly available repositories. We choose these datasets as they all vary in sparsity and that additional features can be added to the items in the datasets using publicly available APIs. Their statistics are summarized in Table 1.

MovieLens[7] The MovieLens item dataset, contains the features (*movieId*, *title*, *genres*), where there are 20 genres in total. We expand the MovieLens dataset by adding additional features from IMDb[2], such as IMDb score (an average rating of a movie in the range between 0-10) and director features e.g. awards and nominations. The added features and their coverage are displayed on Table 2. We chose to use IMDb's average rating as a feature, as opposed to calculating on the average rating based on the interaction dataset, as it would not be possible to calculate the features for items in our test set, which is critical when dealing with item cold start. It is however arguable that such a feature would not necessarily be present in a real world scenario, as it is based on collaborative data. The MovieLens interaction dataset, meaning the rating data, consists of (*userId*, *movieId*, *rating*, *timestamp*), where the ratings are between 0 and 5 and each movie has at least 20 ratings.

Dataset	Sparsity	Users	Items	Interactions
MovieLens^{1M}	95.53%	6.040	3.706	1.000.209
MovieLens^{100K}	93.69%	943	1.682	100.000
BX-Books	99.99%	278.858	271.379	1.149.780

Table 1: Statistics of the datasets

BX-Books[24] The BX-Books item dataset consists of the

features, (*ISBN*, *Book-Title*, *Book-Author*, *Publisher*, *Year-Of-Publication*, *Image-URL-S*, *Image-URL-M*, *Image-URL-L*). We do not however use any of these features except for the ISBN and the Book-Title, as they do not hold any statistical relevance as features for calculating similarity. We therefore expand the BX-Books item dataset by adding additional features from Google Books[1], such as the page count, score (the average rating), votes (number of users who rated the item), and categories. The added features and their coverage for the BX-Books dataset can be seen on Table 3. The BX-Books interaction dataset consists of (*User-ID*, *ISBN*, *Book-Rating*), where the rating is between 0-10.

Dataset	Awards	Nominations	Votes	Director	Score
MovieLens^{1M}	75.92%	75.92%	75.92%	75.92%	75.92%
MovieLens^{100K}	85.13%	85.13%	85.25%	85.13%	85.25%

Table 2: Coverage of IMDb features for MovieLens dataset

Additionally, we observe that there exists 9.082 categories in total where some categories are very niche oriented/specific, and only few items are associated with said category, such as "Jason Bourne (Fictitious Character)", or "C++ (Computer Programming)". As such, we chose to merge such niche categories together by matching strings between parentheses e.g. "Fictitious Character", to create more meaningful/connected features and limit the complexity of the TETs for run-time purposes. Afterwards, we map the remaining categories into 48 unique categories, based on the BISAC Subject Headings List[2]. Furthermore, we filter our interaction dataset such that users who have rated less than 20 books are removed from the dataset, including their interactions, giving us a new interaction dataset consisting of 294.315 interactions, 7124 users and 10632 items with a sparsity of 99.6%. There are two reasons for doing this: (1) the dataset becomes less sparse and more in line with MovieLens, as they also have a minimum of 20 ratings per user, and (2) the dimensions of both the user-item and item-item similarity matrix becomes much smaller, making it easier to compute the matrices. The TET for the MovieLens dataset, and the features

Dataset	Page Count	Categories	Score	Votes
BX-Books	99.8%	100%	84.63%	84.63%

Table 3: Coverage of Google Books features for the filtered BX-Books dataset.

used in our experiments can be found on figure 2, while the TET for BX-Books dataset can be found on figure 14 in the Appendix.

5.1.2 Baselines

Bayesian Personalised Ranking (BPR)[15] is a pairwise personalized ranking optimization framework that deals with (*user, positive item, negative item*) triplets, which adopts stochastic gradient descent (SGD). It assumes that the users prefers positive items over all other non-observed items. The pairwise personalized ranking loss function is derived from the maximum posterior estimator, that aims to maximize the posterior probability (the revised or updated probability of an interaction occurring after taking into consideration new information), s.t. we maximise the prediction difference between a positive example and a randomly chosen negative example. This baseline was implemented using the library LightFM[11], which is a Python implementation of a number of popular recommendation algorithms. We use the default parameters, for setting the number of components (the dimensionality of the feature latent embeddings) and the number of epochs (an epoch refers to a full cycle through the training dataset).

Weighted Approximate-Rank Pairwise loss (WARP)[23], also deals with (*user, positive item, negative item*) triplets, like the BPR model, and also adopts SGD. However, unlike BPR, the negative items are not chosen at random, but chosen from among those negative items which would violate the desired item ranking given the state of the model. In other words, the WARP loss function will randomly sample output labels of a model, until it finds a pair which it knows are wrongly labelled, and will then only apply an update to these two incorrectly labelled examples. This is useful when only positive interactions are present and optimising the top of the recommendation list (precision@k) is desired. This baseline was implemented using LightFM[11], where we also employ the default parameters for setting the number of components and epochs.

ItemAttributeKNN utilizes content-based signals, meaning an items features/attributes, for constructing an item-item similarity matrix using cosine similarity. Then, using this item-item matrix, it finds the k nearest neighbors of each item and sets their respective entry values to 1, while all other items entry values are set to 0. The philosophy is, that by finding similar items in combination with a user's ratings on those similar items, we can infer a user's rating on a given item. This baseline was implemented using the recommender system library MyMediaLite[6], where we use the default parameters. Specifically, the hyper-parameter ζ that controls the importance of each factorization is set to 0.5, and k is set to 80.

Random, as the name suggest, recommends item to users randomly. We include it as an experimental baseline. This baseline was implemented using MyMediaLite[6].

LCE[18] We include LCE as a baseline as described in section 4.2-4.5, where we use the publicly available Matlab implementation[3]. We set the hyper parameters as follows: $k = 300$ (the dimensionality of the latent space), $\zeta = 0.5$, $\lambda = 0.5$, $\epsilon = 0.001$, $\varphi = 0.0$ or $\varphi = 0.05$ depending on whether graph regularization is included, and the maximum number of iterations is set to 600.

5.1.3 Evaluation metrics

We adopt two widely used evaluation protocols as done in [20, 4, 13, 22, 3], Average Precision@K, Average Recall@K.

Average Precision@K is the fraction of $K \subseteq Pred_{items}$ recommended items that is relevant to the user, across all k-folds:

$$Precision@K = \frac{\sum_{u \in U} \#tp}{\sum_{u \in U} \#tp + \#fp}$$

[1] https://developers.google.com/books

[2] https://bisg.org/page/bisacedition

[3] https://github.com/msaveski/LCE

Average Recall@K is the fraction of $K \subseteq Pred_{items}$ recommended items that is relevant to the user and correctly identified, across all k-folds:

$$Recall@K = \frac{\sum_{u \in U} \#tp}{\sum_{u \in U} \#tp + \#fn}$$

where U denotes the set of users, $\#tp$ denotes the number of true positives, the number of items that a user u has rated from the list of K recommended items and $\#fn$ denotes the false negatives, being the number of rated items for u that were not included in the K recommended items.

We set K = [3,5,10,20,50], as done in [20, 4, 13].

5.1.4 Data Processing

The baseline methods presented in section 5.1.2 have different requirements in terms of the input. This section will cover the data processing considerations for all of them, including the construction of TETs.

TETs[8] The TETs are built according to the TET specification seen on figure 2 for MovieLens and figure 14 for BX-Books. The values for the logistic evaluation function have been set manually $\boldsymbol{\beta} = (-3, 1)$ for MovieLens, and $\boldsymbol{\beta} = (0, 1)$ for BX-Books. These values have been set such that we get evenly distributed histograms for each of the datasets. In addition, we normalize some of the features to ensure that all the values are within the same numerical range, such that the logistic evaluation function can evaluate all values correctly. If this consideration was not made, then some values would be evaluated at the extreme, e.g. 1 or 0, due to the value disparity, e.g. an IMDb score value is between 0 and 10, while the number of users who scored a movie (votes) could range from 0 to 1.000.000. However, if the disparity of values is not of noticeable difference, then the appropriate solution would be to find the appropriate weight and bias values for the logistic function. We choose to use feature scaling for normalizing, as it is the most common method to bring values into the range $[0, 1]$. We do so with the set of real numbers for a feature $P = (p_1, ..., p_k)$, where if $P \in \mathbb{R}$, then $p_i (p_i \in P)$ is normalized $p_i' = \frac{p_i - \min(P)}{\max(P) - \min(P)}$. The max and min function simply finds the highest and lowest number in the set P, in case of a p_i so that $\frac{p_i - \min(P)}{\max(P) - \min(P)} > 1$ (e.g. if test set contains higher values of p compared to training set), then we simply say that $\frac{p_i - \min(P)}{\max(P) - \min(P)} \in [0, 1]$ and any values higher than 1 are then set to 1.

LCE[18] The input for this method is a user-item interaction matrix $\mathbf{X_u} \in \mathcal{R}^{n \times u}$ and an item-feature matrix $\mathbf{X_s} \in \mathcal{R}^{n \times f}$, where n is the number of items and f is the number of features. The features in $\mathbf{X_s}$, for the MovieLens datasets, have been constructed s.t. the genres, and scores have been onehot encoded, while awards, nominations, and votes, have been normalized and is used as a scalar value. It is important to note that we do not include director as a onehot encoded feature, as it has a severe negative impact on the performance. The same approach is used when constructing the features for $\mathbf{X_s}$ for the BX-Books dataset, where we onehot encode categories and use the normalized scalar value for the remaining features. Furthermore, if graph regularization is used, we compute either a cosine or Pearson similarity matrix, or use the existing structural similarity matrix from the TET's. The cosine similarity matrix is computed from the row vectors (or column vectors) of the $\mathbf{X_s}$ matrix:

$$\mathbf{X_i}_{cosine-sim} = \frac{\mathbf{X_s} \cdot \mathbf{X_s^T}}{\|\mathbf{X_s}\| \|\mathbf{X_s^T}\|} \tag{17}$$

For Pearson correlation, the correlation matrix is computed for the row vectors of $\mathbf{X_s}$ s.t. for every pair of vectors x and y (i.e. pairs of items), we obtain a similarity score at the i, j'th entry of the similarity matrix:

$$\mathbf{X_i}_{Pearson[i,j]} = r_{xy} = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sqrt{(\sum x_i^2 - n\bar{x}^2)}\sqrt{(\sum y_i^2 - n\bar{y}^2)}} \tag{18}$$

where r_{xy} is the calculated Pearson correlation in the entry $(i, j) \in \mathbf{X_i}$, and n is the sample size of items.

LCE-SSF The input for our method is an user-item interaction matrix $\mathbf{X_u}$ and an item-item matrix $\mathbf{X_i}$, where $\mathbf{X_i}$ have been created using the structural similarity between TETs as items. This item-item matrix is simply constructed by calculating the distance for each item to all other items, using the NHTs described in section 4.1.3.

BPR & WARP[11] The input for these methods is a user-item interaction matrix $\mathbf{X_u}$, and an item representation expressed in terms of representations of their features: an embedding is estimated for every feature (except for director), and these features are then summed together to arrive at representations for users and items. When splitting the interaction matrix, as these methods does not handle cold-start inherently, we include all items and users in our training set, and instead of removing the interactions associated with the items in the test set from the training set, we instead set the respective values to 0. It is important to note that this is a fix, as all the items and users have to be present at the time of training for the model to function properly.

Random & ItemAttributeKNN[6] The input for these methods is a rating/interaction dataset and an item-feature mapping, where each feature has been one-hot encoded to unique ids, such that the item feature mapping consists of tuples of the form (*item_id*, *feature_id*). As with LCE, we exclude the director feature as it adds too much noise.

5.1.5 Evaluation Protocol

We employ k-folds cross-validation, such that for each of the datasets (MovieLens^{1M}, MovieLens^{100K}, BX-Books), we count the number of ratings for each item in the dataset and sort them such that the items with the most interactions are located at the beginning of the list. Then, sequentially, the items are divided into k folds, such that the first item in the list is placed into the first fold, the second item in the second fold, and so forth, until all items have been divided among the folds. This is done to ensure an even data distribution across folds, s.t. we do not remove or include all popular items in one fold. However, there is no guarantee that we do not remove all interactions for some users in some folds. When training and evaluating the models, we iterate over every fold s.t. the current fold is used as a test set, and the remaining folds as a training set. The training set consists of all interactions except the ones associated with the items in the current fold and the test set consist of all the events associated with the items in the current fold. We use 5 folds

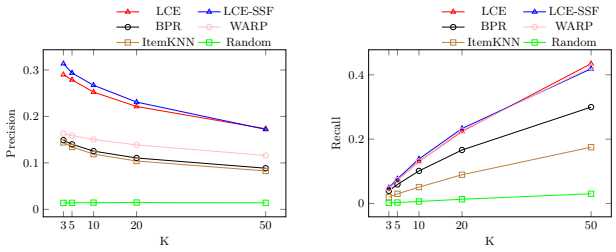
($k = 5$), selecting 20% of items as cold items for each fold.

5.2 Results

This section contains a series of experiments in the item cold-start scenario, that attempts to answer the three questions posed in beginning of section 5. The results for all of the experiments conducted in this paper can be seen on table 4.

5.2.1 RQ1: Baseline Comparison

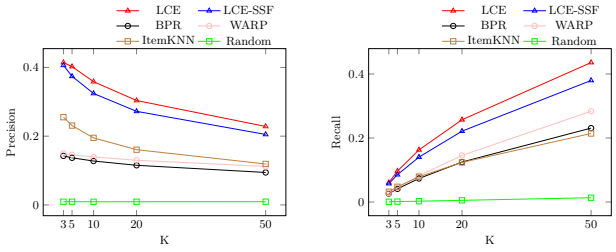
The results on figure 5, 6, 7 shows the average precision and recall for the methods listed in section 5.1.2 on the datasets, MovieLens^{100K}, MovieLens^{1M} and BX-Books.



(a) Precision@K **(b)** Recall@K

Figure 5: Comparison of methods on MovieLens^{100k}.

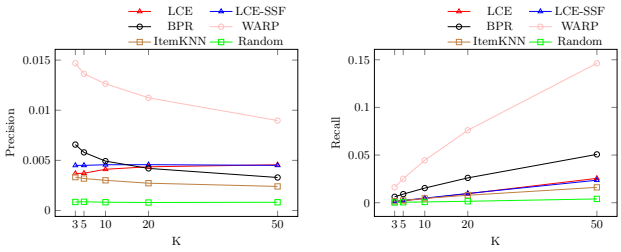
We can observe on figure 5a, that LCE-SSF performs the best in terms of precision when K = $[3, 5, 10, 20]$, significantly outperforming LCE by 8.2% when K=3, 5.29% when K=5. This trend of decrease in difference of performance between LCE-SSF and LCE continues as K increases, ending with LCE outperforming LCE-SSF with 0.83%, achieving the best precision when K=50. This indicates that the structural similarity helps facilitate pushing relevant items towards the top of the list. This is significant, as users are usually presented with a short list of recommendations, making accurate recommendations on the top of the list an important part of user experience. Similarly, we can observe the same pattern on figure 5b, where LCE-SSF performs the best in terms of recall at all K, except when K=50, where LCE outperforms our model with 3.7%.



(a) Precision@K **(b)** Recall@K

Figure 6: Comparison of different methods on MovieLens^{1M}.

However, if we observe the performance when running on MovieLens^{1M} on figure 6, then we note that the LCE out performs all baselines, both in terms of recall and precision at all K. Specifically, the difference in precision increases as we increase K, s.t. LCE outperforms LCE-SSF with 1.95% when K=3, 7,05% when K=5, and with 10.2% when K=50. The same trend holds for recall as well. One explanation for the difference in results between MovieLens^{100K} and MovieLens^{1M} is that LCE-SSF simply performs better in denser datasets compared to LCE, or that the increase in items has a negative effect on LCE-SSF, adding too many features.



(a) Precision@K **(b)** Recall@K

Figure 7: Comparison of different methods on BX-Books.

Lastly, we observe how our baselines perform on the much sparser dataset, BX-Books, as seen on figure 7. Here we note that, in general, the precision and recall is drastically lower than compared to the previous datasets. One explanation for these performances is the nature of the dataset being very sparse, consisting of many users and items, making it hard to accurately recommend items. We would also argue that the features available for these items are sparse, making it hard to distinguish between items and perform well when dealing with item-cold start. However, we still observe on figure 7a that LCE-SSF slightly outperforms LCE in terms of precision for all K, initially with 22.08% when K=3, 11.15% when K=10, and 3.26% when K=50. Most noticeable, is that the baselines BPR and WARP outperforms LCE and LCE-SSF for most K. Specifically, WARP, performs really well compared to the other baselines, because it is optimized for very sparse datasets, as mentioned in [11] and WARP generally outperforms BPR, which is explained in the documentation[4].

5.2.2 RQ2: Graph Regularization

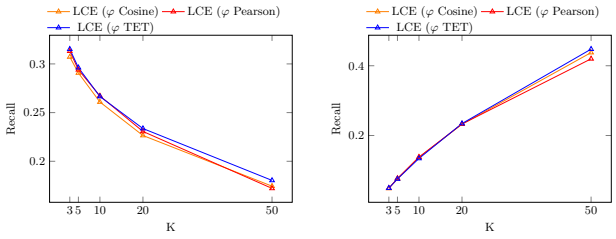
This section of our experiments is devoted to investigating the performance of graph regularization using different item-item similarity matrices, cosine and Pearson as described in section 5.1.4, in comparison with TET as a graph regularization term. The different models shown in these experiments are labeled i.e. LCE (φ TET), where φ denotes that it is using graph regularization and that the matrix $\mathbf{A}$ used for graph regularization is based on the item-item similarity matrix constructed using either Pearson, cosine, or TETs. The graph regularization parameter φ, that control the influence of the graph regularization term, is set to 0.05.

We can see on figure 8 the performances of the different graph regularization terms on the MovieLens^{100K} dataset. We notice that TETs as a graph regularization term, slightly outperforms cosine by 2,6% and Pearson by 0,65% when K=3, and 3,38% and 4,76% when K=50. We also see on figure 8a that Pearson performs the best in terms of precision when K = 10, outperforming TETs with 0,23%, and in terms of recall on figure 8b when K=$[5, 10]$, outperforming TETs

[4] https://making.lyst.com/lightfm/docs/examples/warp_loss.html

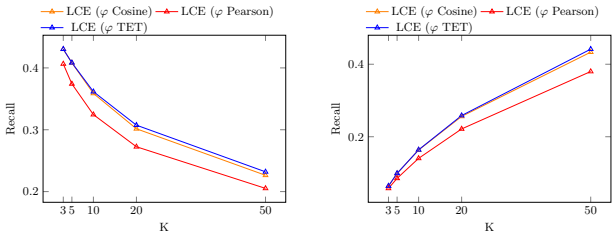
MovieLens^{100K}	Precision@3	Precision@5	Precision@10	Precision@20	Precision@50	Recall@3	Recall@5	Recall@10	Recall@20	Recall@50
LCE	0,2897	0,2787	0,2523	0,2213	0,1734	0,0457	0,0728	0,1307	0,2242	0,4346
LCE-SSF	0,3135	0,2934	**0,2671**	0,2308	0,1720	0,0490	0,0766	**0,1383**	0,2330	0,4187
ItemAttributeKNN	0,1440	0,1343	0,1188	0,1040	0,0826	0,0193	0,0299	0,0508	0,0892	0,1749
Random	0,0139	0,0140	0,0144	0,0147	0,0140	0,0017	0,0029	0,0062	0,0130	0,0302
BPR	0,1488	0,1400	0,1253	0,1105	0,0885	0,0390	0,0591	0,1011	0,1661	0,2996
WARP	0,1628	0,1587	0,1502	0,1387	0,1159	0,0457	0,0725	0,1305	0,2270	0,4237
LCE (φ Cosine)	0,3071	0,2907	0,2608	0,2264	0,1744	0,0484	0,0756	0,1370	0,2325	0,4376
LCE (φ Pearson)	0,3130	0,2939	**0,2671**	0,2306	0,1721	0,0488	**0,0771**	0,1381	0,2329	0,4197
LCE (φ TET)	**0,3151**	**0,2961**	0,2665	**0,2336**	**0,1803**	**0,0491**	0,0754	0,1342	**0,2342**	**0,4475**
LCE ($\mathbf{X}_i$ = Cosine)	0,2687	0,2574	0,2461	0,2259	0,1771	0,0384	0,0613	0,1207	0,2235	0,4322
LCE ($\mathbf{X}_i$ = Pearson)	0,2285	0,2066	0,1825	0,1628	0,1346	0,0336	0,0520	0,0923	0,1658	0,3383
MovieLens^{1M}	Precision@3	Precision@5	Precision@10	Precision@20	Precision@50	Recall@3	Recall@5	Recall@10	Recall@20	Recall@50
LCE	0,4141	0,4024	0,3585	0,3037	0,2282	0,0608	0,0967	0,1628	0,2569	0,4357
LCE-SSF	0,4060	0,3740	0,3242	0,2722	0,2049	0,0574	0,0853	0,1402	0,2213	0,3797
ItemAttributeKNN	0,2552	0,2307	0,1950	0,1606	0,1191	0,0322	0,0476	0,0786	0,1237	0,2140
Random	0,0092	0,0093	0,0091	0,0092	0,0093	0,0007	0,0012	0,0025	0,0052	0,0134
BPR	0,1426	0,1370	0,1277	0,1152	0,0944	0,0263	0,0410	0,0735	0,1248	0,2308
WARP	0,1495	0,1459	0,1396	0,1299	0,1118	0,0277	0,0443	0,0821	0,1455	0,2834
LCE (φ Cosine)	0,4302	0,4081	0,3587	0,3015	0,2263	0,0634	0,0985	0,1633	0,2560	0,4333
LCE (φ Pearson)	0,4064	0,3741	0,3244	0,2724	0,2051	0,0575	0,0852	0,1403	0,2215	0,3799
LCE (φ TET)	**0,4303**	**0,4083**	**0,3615**	**0,3073**	**0,2319**	**0,0641**	**0,0989**	**0,1638**	**0,2584**	**0,4416**
LCE ($\mathbf{X}_i$ = Cosine)	0,3607	0,3555	0,3279	0,2893	0,2248	0,0511	0,0828	0,1451	0,2402	0,4239
LCE ($\mathbf{X}_i$ = Pearson)	0,3321	0,3102	0,2685	0,2338	0,1880	0,0451	0,0695	0,1164	0,1918	0,3533
BX-Books	Precision@3	Precision@5	Precision@10	Precision@20	Precision@50	Recall@3	Recall@5	Recall@10	Recall@20	Recall@50
LCE	0,0036	0,0037	0,0041	0,0044	0,0046	0,0013	0,0020	0,0044	0,0095	0,0255
LCE-SSF	0,0045	0,0045	0,0046	0,0046	0,0045	0,0014	0,0023	0,0048	0,0096	0,0236
ItemAttributeKNN	0,0033	0,0031	0,0030	0,0027	0,0024	0,0014	0,0023	0,0045	0,0078	0,0161
Random	0,0008	0,0008	0,0008	0,0007	0,0008	0,0002	0,0004	0,0007	0,0014	0,0039
BPR	0,0065	0,0057	0,0049	0,0042	0,0032	0,0062	0,0090	0,0153	0,0260	0,0506
WARP	**0,0146**	**0,0136**	**0,0126**	**0,0112**	**0,0089**	**0,0162**	**0,0251**	**0,0444**	**0,0760**	**0,1462**
LCE (φ Cosine)	0,0039	0,0041	0,0042	0,0044	0,0042	0,0014	0,0021	0,0042	0,0093	0,0228
LCE (φ Pearson)	0,0037	0,0039	0,0041	0,0041	0,0042	0,0012	0,0022	0,0043	0,0087	0,0235
LCE (φ TET)	0,0039	0,0040	0,0044	0,0045	0,0044	0,0015	0,0022	0,0049	0,0101	0,0252
LCE ($\mathbf{X}_i$ = Cosine)	0,0040	0,0041	0,0046	0,0047	0,0044	0,0014	0,0023	0,0049	0,0099	0,0243
LCE ($\mathbf{X}_i$ = Pearson)	0,0040	0,0043	0,0050	0,0046	0,0045	0,0013	0,0024	0,0054	0,0098	0,0247

Table 4: Performance of our model and baselines on different datasets in terms of Precision@K & Recall@K



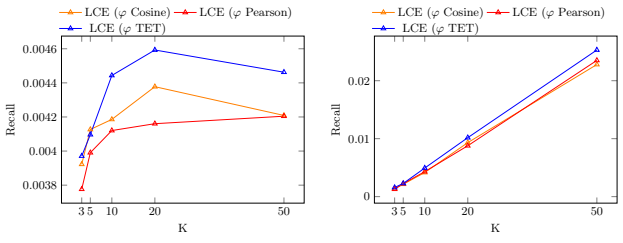
(a) Precision@K **(b)** Recall@K

Figure 8: Comparison of LCE using different graph regularization methods on the MovieLens^{100K} dataset.



(a) Precision@K **(b)** Recall@K

Figure 9: Comparison of LCE using different graph regularization methods on the MovieLens^{1M} dataset.



(a) Precision@K **(b)** Recall@K

Figure 10: Comparison of LCE using different graph regularization methods on the BX-Books dataset.

with roughly 2.5%. In general, we see improvement across the board when adding graph regularization, only Pearson shows worse performance when K=50 in terms of both precision and recall.

We observe the results for the MovieLens^{1M} dataset on figure 9, and notice that cosine performs better compared to the performance on figure 8, and that Pearson performs relatively worse. Most importantly, we see that TETs as a graph regularization term performs the best comparatively, even if only slightly. The difference in performance for TET compared with cosine slightly increases with K, going from 0.02% to 2.46%, which is arguable insignificant.

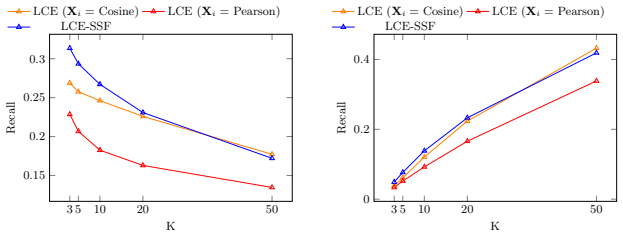
As for the performance on the BX-Books dataset, we observe on figure 10 that TET, again, generally gives us the best performance. In detail, we see on figure 10a that TET outperforms cosine between 1,23% to 6,15% in terms of precision depending on K, except for when K=5, then cosine outperforms TET with 0,75%. This difference is even big-

ger if we look at the performance in terms of recall on figure 10b, where we see TET outperform cosine for all K, ranging from 6,45% to 18,67% difference in recall. However, these results are so low in general that the difference is arguable insignificant in perspective.

5.2.3 RQ3: Similarity

We find it natural that in order to investigate the impact of the TET induced structural similarity matrix in combination with LCE, we should compare its performance with other similarity matrices, e.g. Pearson and cosine similarity. The models used in these experiments use different item-item matrices as input, i.e. LCE ($\mathbf{X}_i$ = Cosine) denotes that the item-item matrix has been computed using cosine similarity. Note that LCE-SSF is the same as LCE ($\mathbf{X}_i$ = TET).

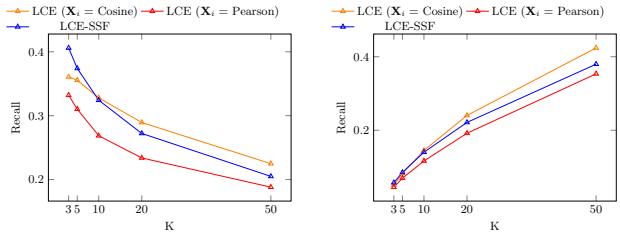
If we look at figure 11, we can see that the LCE model using the item-item matrix constructed using the TETs structural similarity outperforms the other similarity matrices. In detail, we see that on figure 11a that LCE-SSF displays between 16,65%-2,17% higher precision than cosine, and on figure 11b that LCE-SSF outperforms cosine with 27,42% - 4,24%, when K=$[3, 5, 10, 20]$. We note that Pearson performs the worst of the models, with a considerable difference in performance. This may seem counter-intuitive, given the positive results using Pearson as a graph regularization term. However, since we introduce a non-negativity constraint on the item-item matrix in section 4.4, we cannot utilize the negative distances that Pearson provides, and as such we set all negative values to zero. The pattern between the methods using cosine and TET hints at TETs being better at recommending when K is low. We observe the same pattern when performing experiments on MovieLens^{1M}, as seen on figure 12, where we outperform cosine with 12.55% when K=3, and 5,2% when K=5. However, on this larger dataset cosine performs the best in terms of both precision and recall when K=$[10, 20, 50]$, where this only occurred on the dataset MovieLens^{100K} when K=50. Additionally, we see that the difference between LCE-SSF, cosine and Pearson decrease with the increase in the dataset.



(a) Precision@K **(b)** Recall@K

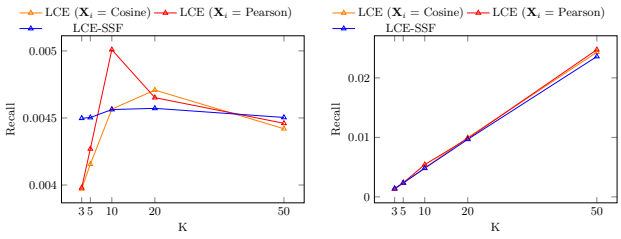
Figure 11: Comparison of LCE using different item similarity matrices on the MovieLens^{100k} dataset.

The results for the experiments ran on the BX-Books dataset is shown on figure 13, where we notice a new, more randomly seeming pattern. Specifically, we note that precision for TET on figure 13a is relatively stable, while cosine and Pearson steadily increase, before decreasing again when K=50.



(a) Precision@K **(b)** Recall@K

Figure 12: Comparison of LCE using different item similarity matrices on the MovieLens^{1M} dataset.



(a) Precision@K **(b)** Recall@K

Figure 13: Comparison of LCE using different item similarity matrices on the BX-Books dataset.

Furthermore, we see on figure 13b that the recall values are much closer together for all methods (due to the scaling of the figure), not displaying the same disparities as seen on figure 13a. The trend of TETs performing the best when K is low persists, displaying a somewhat large difference in performance, however given the overall low performance its significance is arguable.

An explanation for the low precision and recall values in general on the BX-Books dataset, besides the sparsity of interactions, could be the sparsity of descriptive (meaningful) features for this dataset. Compared to the MovieLens dataset, books only have one category, while a movie could consist of many genres. Another approach to recommendation on this dataset could be to use the words in a books description as features, as done in the original LCE paper with news articles and emails.

6. DISCUSSION

We devote this section to discuss some of the considerations and choices made in regards to writing this paper.

6.1 TET design choices

6.1.1 Numerical values

In the original count-of-counts method, *Jaeger et al.* [9], only Boolean attributes are considered. We extend their formulation by including numerical values as node attributes, but doing this poses some problems. It becomes harder to choose good candidates for the bias β_0 and weights $\beta_1, ..., \beta_m$ as the numerical features can vary significantly in their range of values. We normalize all numerical features to handle this and manually set the weight and biases. *Jaeger et al.* [9] experiments with learning the weights and biases from e.g. using stochastic gradient descent. We did not look into this.

On the other hand, we are able to include additional features without having to design intervals for numerical features, which would not be possible if we only allowed Boolean attributes. In that case, given a numerical feature in a TET such as budget, we would design a TET with the nodes e.g. "$\leq$ 100.000", "$\geq$ 100.000", returning 1 or 0 depending on the evaluation.

6.1.2 EMD with sibling nodes

When calculating EMD between items' NHTs, we consider the individual histograms of nodes in the NHT, which follows the structure of its TET specification. Intuitively, when comparing two items, the overall EMD is determined by the difference in histogram values e.g. two movies where their genres differ. However, for movies that follows the specification from figure 2, genre nodes account for the majority of the nodes and as such, account for most of the difference between two movies. This makes sense from a purely structural perspective, but means that other nodes besides the genre nodes are "less important" in the EMD calculation. Dividing by the number of siblings nodes for a given NHT could solve this problem, but it depends on how the TET specification is structured. In order to, for example, scale the importance of the single genre nodes shown in figure 2, they would have to be nested s.t. they are children of a node *has_genres(m,g)*. Otherwise, we are effectively just scaling down the EMD of genres and their sibling nodes, since they would have the same number of siblings.

6.2 Choice of datasets and features

It is evident from the experiments section that the choice of datasets, which features to include and how to preprocess the data, have great significance. Good performance of the models obviously depends on how sparse or dense the dataset is, in order to obtain accurate predictions for the users. However, for LCE-SSF or the LCE variant that uses TET similarity as graph regularization, it is also important that we can construct meaningful TETs from the input features. For MovieLens, movies are relatively easy to compare as there are fewer total features, e.g. genres, and some movies also overlap in terms of their genres. For BX-books, every book only has a single category, and there is no overlap (except for exact matches), making them harder to compare. Also, the total number of features for BX-books is much greater compared to MovieLens, resulting in a much larger TET specification.

6.3 Practical Limitations

While LCE-SSF does outperform other baselines on various settings, we need to keep in mind the additional trade-offs LCE-SSF incurs in relation to standard LCE. For example, in standard LCE, $\mathbf{X_i} \in \mathcal{R}^{n \times f}$ is a matrix of n items and f number of features describing them. In LCE-SSF, $\mathbf{X_i}$ instead has dimensions $\mathcal{R}^{n \times n}$. This means that introducing new items to LCE-SSF requires calculating the item-item similarity vector $1 \times n$, where n increases for every new item introduced, as opposed to having a fixed set of features. As such, the matrix $\mathbf{X_i}$ can quickly become very large. One way to circumvent this is to introduce a threshold for the matrix $\mathbf{X_i}$ s.t. for any two items $\mathbf{X_i}[\mathbf{i}, \mathbf{j}]$ whose similarity is e.g. ≥ 10.0, will be represented by a nil value. This would result in a sparse representation of $\mathbf{X_i}$

Another thing to consider is the calculation of the item-item similarity itself. In standard matrix factorization, it is sufficient to consider user-item interactions, along with their ratings. After factorization, one can then quickly obtain the predictions by multiplying the decomposed matrices together.

In LCE-SSF, computing the item-item similarity matrix for factorization requires comparing the histograms between items NHTs. This step takes a varying amount of time, but depends on the size of the TET specification, i.e. the number of histograms to compare. In real applications, this delay may not be acceptable.

Finally, introducing a new item with e.g. a new genre, s.t. it is not in the current TET specification, requires calculating the entire item-item similarity matrix again. This is because the items TET's must have the same structure in order to compare the items.

7. CONCLUSION AND FUTURE WORK

To improve on item cold-start recommendation, we have in this work proposed a combination of methods, LCE and TETs, that combined is a recommender system that utilizes structural similarity as content information in combination with collaborative information in a unified matrix factorization framework. Given the results seen in table 4 we can conclude that LCE, when utilizing TETs, generally outperforms LCE that does not use TETs. However, even though the precision for some of the results is not a large increase, it still showcases the potential of TETs. Especially, if used in conjunction with a dataset whose features are well suited to be described using TETs.

As for possible future work, testing different TET specifications, with and without adding new features, could be interesting for comparison purposes. Specifically, utilizing features that can be counted using the free variables in the TETs, such as the number of male and female actors in a movie. We hypothesize that using another approach than normalization for handling numerical values in leaf nodes in TETs, such as manually encoding the selected features e.g. votes into intervals that corresponds to the number of histogram bins, would ensure more evenly distributed histograms at the leaf level and lead to an increase in performance. Based on the results we conclude that including the expressive power of TETs in combination with methods can potentially lead to more powerful methods. Therefore, trying TETs with other methods could be interesting, or simply testing on other datasets. It could also be beneficial, especially for larger datasets, to utilize k-nearest-neighbors as done in ItemAttributeKNN, and set all entries for each items nearest neighbors in the item-item matrix to 1, and all non-neighbors to 0. This would provide, in theory, stronger signals for similar items while removing any potential noise from having many items. and it would allow for the use of sparse matrices. This could help alleviate the practical limitations of scaling our model mentioned in 6.3. Lastly, it would be necessary to perform online evaluation of our proposed model to truly verify any improvements in user satisfaction, but as this is not possible for us, we limit ourselves to offline evaluation.

Acknowledgements

We would like to thank our supervisors, Manfred Jaeger and Peter Dolog, for providing valuable feedback throughout this project.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99.

[2] D. Alberani. Python package: Imdbpy.

[3] Y. Cao, X. Wang, X. He, H. Zikun, and T.-S. Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. 02 2019. doi: 10.1145/3308558.3313705.

[4] C. C. Chen, Y.-H. Wan, M.-C. Chung, and Y.-C. Sun. An effective recommendation method for cold start new users using trust and distrust networks. *Information Sciences*, 224:19–36, 2013. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2012.10.037. URL `https://www.sciencedirect.com/science/article/pii/S0020025512007074`.

[5] A. Cichocki, R. Zdunek, A. Phan, and S. Amari. Nonnegative matrix and tensor factorizations - applications to exploratory multi-way data analysis and blind source separation. *IEEE Signal Processing Magazine*, 25:142–145, 2009.

[6] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *2010 IEEE International Conference on Data Mining*, pages 176–185. IEEE, 2010.

[7] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[8] M. Jaeger, M. Lippi, A. Passerini, and P. Frasconi. Type extension trees for feature construction and learning in relational domains. *Artificial Intelligence*, 204: 30–55, 2013.

[9] M. Jaeger, M. Lippi, G. Pellegrini, and A. Passerini. Counts-of-counts similarity for prediction and search in relational data. *Data mining and knowledge discovery*, 33(5):1254–1297, 2019.

[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009. doi: 10.1109/MC.2009.263.

[11] M. Kula. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.

[12] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *MIT Press*, Neural Information Processing Systems 13:556–562, 2001.

[13] J. Li, M. Jing, K. Lu, L. Zhu, Y. Yang, and Z. Huang. From zero-shot learning to cold-start recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4189–4196, Jul. 2019. doi: 10.1609/aaai.v33i01.33014189. URL `https://ojs.aaai.org/index.php/AAAI/article/view/4324`.

[14] H. Ling and K. Okada. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE transactions on pattern analysis and machine intelligence*, 29(5):840–853, 2007.

[15] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, page 452–461, Arlington, Virginia, USA, 2009. AUAI Press. ISBN 9780974903958.

[16] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)*, pages 59–66. IEEE, 1998.

[17] M. Saveski and A. Mantrach. Item cold-start recommendations:learning local collective embeddings (proof for theorem 1).

[18] M. Saveski and A. Mantrach. Item cold-start recommendations: Learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 89–96, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326681. doi: 10.1145/2645710.2645751. URL `https://doi.org/10.1145/2645710.2645751`.

[19] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 89–96, 2014.

[20] S. Sedhain, A. Menon, S. Sanner, L. Xie, and D. Braziunas. Low-rank linear cold-start recommendation from social data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10758`.

[21] unknown. Earth mover's distance. URL `https://en.wikipedia.org/wiki/Earth_mover%27s_distance`.

[22] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 417–426, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360142. doi: 10.1145/3269206.3271739. URL `https://doi.org/10.1145/3269206.3271739`.

[23] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, page 2764–2770. AAAI Press, 2011. ISBN 9781577355151.

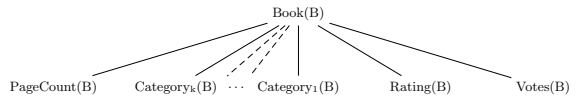


Figure 14: Describes the TET for a movie. The Category_k and Category_1 describes the $1, 2, ..., k$ categories that a book can have.

[24] C.-N. Ziegler, Cai-Nicolas, S. McNee, S. M, Konstan, J. A, Lausen, and Georg. Improving recommendation lists through topic diversification. 01 2005. doi: 10.1145/1060745.1060754.

APPENDIX
A. FIGURES

Summary

In this paper we combine a statistical relational learning model, namely Type Extension Trees(TETs), together with the well known matrix factorization model Local Collective Embeddings(LCE). We explore whether we improve the content-based signals of the LCE model, by combining it with the item-item similarities defined on TETs. We do this as an attempt to tackle the cold-start item problem.

The paper starts off with an introduction where we narrow down the domain and motivate the problem that in cold-start scenarios, it is important to optimize the content-based component of a hybrid model.

In section 2, we talk about the related works for our paper, where we talk about the papers on TETs and LCE, but also talk about the baselines in our experiments, which consists of two matrix factorization models that can handle cold-start scenarios.

We proceed to define a problem definition in section 3, where we explain how we want to calculate the item-item similarity and how we want to use that for scoring how likely a user is to show interest in the new item.

Then section 4 we define our model, by first going through the formal definitions for TETs, how it is structured and how to use it when finding the similarities between items. The calculation of item similarity is done by calculating the distances between TETs, which is done using histograms to reduce time- and space-complexity.

We proceed to define the LCE model, where we use the item-item similarities from the TETs as input together with the user-item interaction matrix. We start by showing how it is calculated, followed by how to include graph regularization and lastly how the method is optimized.

Section 5 is our experiments section, where we start out by talking about the chosen datasets and how we extend the datasets with additional features, since it allows for better similarity calculation. We proceed to explaining the baselines we evaluate on the datasets, followed by explaining the metrics used for evaluation, namely precision and recall. The different baselines require some preprocessing and we explain how this is done e.g. normalizing the data to give an even distribution.

The second part of section 5 consists of the results, where we show the results and try to reason for why they look as they do. The results shows that our proposed combination does show some increase in performance on all datasets, atleast compared to LCE. The experiments are set up using k-folds cross-validation, where we consider 5 folds.

In section 6 we go through a discussion about some design choices we made in the project e.g. that we include numerical values in the TETs, which is not available in the original implementation. This makes it much simpler to construct some features in the TETs.

In section 7 the conclusion shows that using the item similarity given by TETs together with LCE, shows an increase in performance compared to the original implementation of LCE. The best performing being LCE using TET similarity for graph regularization. Finally, we suggest trying different specification to better make use of the counts-of-counts structure that the TETs provide.