Hierarchical Latent Relation Modeling for Collaborative Metric Learning

Viet-Anh Tran*
Deezer Research
France

Guillaume Salha-Galvan
Deezer Research & LIX, École Polytechnique
France

Romain Hennequin
Deezer Research
France

Manuel Moussallam
Deezer Research
France

ABSTRACT

Collaborative Metric Learning (CML) recently emerged as a powerful paradigm for recommendation based on implicit feedback collaborative filtering. However, standard CML methods learn fixed user and item representations, which fails to capture the complex interests of users. Existing extensions of CML also either ignore the heterogeneity of *user-item* relations, i.e. that a user can simultaneously like very different items, or the latent *item-item* relations, i.e. that a user's preference for an item depends, not only on its intrinsic characteristics, but also on items they previously interacted with. In this paper, we present a hierarchical CML model that jointly captures latent *user-item* and *item-item* relations from implicit data. Our approach is inspired by translation mechanisms from knowledge graph embedding and leverages memory-based attention networks. We empirically show the relevance of this joint relational modeling, by outperforming existing CML models on recommendation tasks on several real-world datasets. Our experiments also emphasize the limits of current CML relational models on very sparse datasets.

CCS CONCEPTS

• **Information systems** → *Recommender systems*; • **Computing methodologies** → *Learning latent representations.*

KEYWORDS

Collaborative Metric Learning, Relation Modeling, Attention Mechanisms, Recommender Systems

ACM Reference Format:
Viet-Anh Tran, Guillaume Salha-Galvan, Romain Hennequin, and Manuel Moussallam. 2021. Hierarchical Latent Relation Modeling for Collaborative Metric Learning. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3460231.3474230

*Contact author: research@deezer.com

1 INTRODUCTION

Nowadays, it is common for users to face the situation of *"too much content, too little time"* when listening to music or watching videos on streaming services with large catalogs [5, 24], or when purchasing products on e-commerce websites [23]. These online services heavily rely on recommender systems to address this information overload, by identifying the most relevant content for each user. They are known to be key components to attract and engage users, and are also central to all enjoyable passive experience relying on generated content [3, 24, 35].

A prevalent paradigm to recommend personalized content on these services is Collaborative Filtering (CF). Operating on *implicit feedback data*, such as streams or skips for music streaming, CF methods assume that users who had similar interests in the past will tend to share similar interests in the future [3, 17]. In particular, Matrix Factorization (MF) remains one of the most popular CF baselines [12, 17, 22]. It consists in learning low-dimensional vector representations of users and items from the factorization of an interaction matrix summarizing feedback data, and subsequently modeling the future user–item similarities from inner products between these vectors.

However, MF has recently been competed by *metric learning* methods [6, 7, 11, 14, 21, 32, 36, 38] such as Collaborative Metric Learning (CML) [11], where similarities are instead measured by Euclidean or Hyperbolic distances [31]. Specifically, CML adopts a *triplet loss* strategy [27, 33] to learn vector representations ensuring, as explained in Section 2, that distances between users and their *negative items* (which they did not interact with) are larger than distances between users and their *positive items* (which they interacted with). Relying on the assumption that users should be closer to items they like, this approach implicitly learns user-user and item-item similarities by satisfying the triangle inequality [11].

CML nonetheless suffers from an inherent limitation: each user is represented *by a single vector*, which does not fit the many-to-many nature of real-world recommendation problems [20, 30, 37, 38]. In particular:

- Standard CML fails to capture the *heterogeneity in user-item relations* [20, 30, 38]. Indeed, due to the triangular inequality, a user cannot be simultaneously represented as close to two items that are themselves distant. This is undesirable as, in reality, a same user could simultaneously like very different movies or music genres.

- Moreover, CML does not model the fact that a user's preference for an item depends, not only on its intrinsic characteristics, but also on items they previously interacted with [37], which we refer to as *item-item relations*.

While some extensions of CML, described thereafter in Section 2, aimed at modeling user-item relations [20, 30, 38], they did not consider the item-item ones. On the contrary, recent efforts on item-item relations-aware CML [37] ignored user-item relations. In this paper, we argue that *jointly* modeling both user-item and item-item relations can actually help revealing the underlying spectrum of user preferences, and thus lead to better performances in recommendation. More specifically, our work builds upon the assumption, presented in Section 3, that there exists *a hierarchical structure* in different relation types, and that *user-item relations are built on top of item-item relations*.

This assumption is materialized in the form of a neural architecture, named Hierarchical Latent Relation (HLR), that leverages the recent advances in attention mechanisms, augmented memory networks as well as metric learning to explicitly learn adaptive user-item and item-item relations lurked in implicit feedback data. We provide comprehensive experiments on four real-world datasets, demonstrating the relevance of our proposed approach and its empirical effectiveness, in terms of recommendation accuracy, ranking quality, and popularity bias [24]. Along with this paper, we also publicly release our source code on GitHub to ensure the reproducibility of our results.

This paper is organized as follows. In Section 2, we formulate our recommendation problem more precisely, and recall key concepts on CML and its existing extensions. In Section 3, we present our hierarchical system to jointly model user-item and item-item relations. We report and discuss our experiments in Section 4, and we conclude in Section 5.

2 BACKGROUND

2.1 Problem Formulation

In this paper, we consider a top-K recommendation problem on an online service, with a set of users $\mathcal{U}$ and a set of recommendable items $\mathcal{V}$. We observe some positive implicit and binary user feedback (e.g., clicks, streams or view history logs) on items, gathered in a set $\mathcal{S} = \{(u, v) | u \in \mathcal{U}, v \in \mathcal{V}\}$. If we interpret $\mathcal{U}$ and $\mathcal{V}$ as two disjoint and independent sets of vertices in a bipartite user-item graph $\mathcal{G}$, then $\mathcal{S}$ represents the ensemble of edges that connect a vertex in $\mathcal{U}$ to one in $\mathcal{V}$. Missing interactions do not necessarily imply a negative feedback; in most cases, the user is simply unaware of the items' existence. We use $\mathcal{N}_u \subseteq \mathcal{V}$ to denote the set of items that user u previously interacted with. In such a setting, we aim at recommending $K > 0$ items $v \in \mathcal{V} \setminus \mathcal{N}_u$ to each user u, for which they should positively interact with. The problem can thus be framed as a missing link prediction task in the bipartite graph $\mathcal{G}$.

2.2 Collaborative Metric Learning

To address this top-K recommendation problem, CML [11] learns a *metric space* encoding $\mathcal{S}$ and representing both users and items as low-dimensional vectors. In such a space, user-item similarities are measured with the Euclidean distance $d(u, v) = ||p_u - q_v||_2^2$, where $|| \cdot ||_2$ denotes the L2-norm, and where p_u and q_v are the respective representation vectors of user u and item v in the metric space. At its heart, CML aims at pulling closer matching user-item pairs in $\mathcal{S}$, and at pushing away non-matching pairs, up to a certain margin $m > 0$. Formally, let us denote by $\mathcal{T}$ the ensemble of *triplets* (u, v, v^-) such that the user u interacted with item v and did not interact with item v^-, i.e. $(u, v) \in \mathcal{S}$ and $(u, v^-) \notin \mathcal{S}$. Let us denote by p_u , q_v and q_{v^-} their respective vectors. Then, CML essentially aims at solving:

$$\min_{||p_u||_2 \leq 1, ||q_v||_2 \leq 1, ||q_{v^-}||_2 \leq 1} \sum_{(u,v,v^-) \in \mathcal{T}} \Big[||p_u - q_v||_2^2 - ||p_u - q_{v^-}||_2^2 + m \Big]_+ \tag{1}$$

where $m > 0$ is a fixed margin hyperparameter, and $[x]_+ = max(0, x)$ denotes the standard hinge loss. Finally, the top-K items recommended to the user u will correspond to its K closest ones in the resulting space.

2.3 Modeling User-Item Relations

2.3.1 On the Limits of Single Vector Representations. Despite promising results against popular alternatives including MF [10, 12, 22], standard CML suffers from the fact that each user is represented *by a single vector* in the metric space. As mentioned in Section 1, the *heterogeneity in user preferences* can not be satisfactorily handled under such restriction, as the triangular inequality imposes that p_u cannot be simultaneously close to two items that are distant. For instance, a user could simultaneously like metal and classical music, whereas songs from these two music genres are likely to be far from each other in a CML-based metric space. Also, a single distance hardly captures the fact that a user might like a song or a movie for multiple reasons, such as its genre, its director, or its casting.

2.3.2 User Translation in Vector Space. To overcome these limitations, recent works [20, 30] leveraged the concept of *translation in vector space* to learn an adaptive *relation vector* r_{uv} for each user-item pair (u, v). This concept takes inspiration from reasoning over semantics in a Knowledge Graph (KG) embedding [4], where some entities and relations from a KG are represented in a vector space. As an example, let us consider a KG connecting geographical entities such as countries and cities. Given a fact/an edge (e.g. "u is the capital of v"), the semantic relation ("capital") would be captured by a translation vector r, in the sense that the embedding vectors of u and v, say p_u and p_v, would verify $p_u + r \approx p_v$. Then, the plausibility of a semantic link between two new entities (e.g. "is w the capital of x?") is measured from the distance between $p_w + r$ and p_x. TransE [4] is a famous example of such a translational distance model.

2.3.3 User-Item Relation Modeling for CML. Transposing this concept to recommendation, extensions of CML [20, 30] proposed to learn metric spaces by minimizing $||p_u + r_{uv} - q_v||_2^2$ terms in equation (1), with a relation vector r_{uv} for each user-item interaction. Notably, Tay et al. [30] introduced Latent Relation Metric Learning (LRML), which learns r_{uv} vectors by means of a weighted adaptive translation over an augmented memory via neural attention. This permits translating a user u's vector ($p_u \rightarrow p_u + r_{uv}$) based on the candidate item v, and therefore capturing heterogeneities in user preferences contrary to CML. Zhou et al. [38] then further extended this idea to multiple user-item relation types.

Another related model is Collaborative Translational Metric Learning (TransCF) [20], whose authors argued that the above approach might be prone to *overfitting* due to the large number of parameters involved, and that it does not allow the collaborative information to be explicitly modeled as no parameters are shared among users and items. Authors of TransCF instead took inspiration from neighborhood–based CF algorithms [13, 16], that assume that similar users display similar item preferences and that similar items are consumed by similar users. TransCF employs the neighborhood information of users and items to learn translation vectors r_{uv}. More precisely, each item is regarded as the average taste of users having interacted with that item. Likewise, each user's taste can be represented by the average representation of items this user interacted with. Given the respective representations of an item and a user from the neighborhood perspective, the relation vector r_{uv} is modeled by a simple element-wise vector product of these two vectors.

2.4 Modeling Item-Item Relations

2.4.1 On the Limits of Single Vector Representations. Besides user-item relations, single vector representations also fail to capture latent *item-item relations*. This refers to situations where a user's preference for an item depends, not only on its characteristics, but also on items they previously interacted with [37]. For instance, some users could decide to interact with an item because of its similarity with several items in their favorites. The aforementioned LRML and TransCF extensions of CML also ignore this aspect. As they focus on a single user-item interaction at a time, they can not capture relations between a candidate item and items that a user previously interacted with.

2.4.2 Item-Item Relation Modeling for CML. Recent efforts from Zhang et al. [37] aimed at learning adaptive relation vectors r_{uv} that capture such latent item-item relations. Authors proposed Adaptive Collaborative Metric Learning (AdaCML), an extension of the popular FISM model [13]. As in FISM, users are represented by a linear combination of the items they interacted with in the past. However, while FISM considers that all items contribute equally when estimating the similarity between a user and a target item, AdaCML learns the impact of each historical item via an attention mechanism. Attention weights depend on each target item and are derived from inner products of this item with historical items in users' favorite lists. The final relation vector r_{uv} is then derived from a weighted sum of all historical items that the user interacted with. We emphasize that AdaCML only concentrates on item-item relations, and does not model the heterogeneity lurked in each *user-item* interaction.

3 HIERARCHICAL LATENT RELATION MODELING

In this section, we introduce our proposed Hierarchical Latent Relation (HLR) model, and its extension HLR++, to jointly learn user-item and item-item relations. Their overall architectures are summarized in Figure 1 and Figure 2.

3.1 Motivations

From Sections 2.3 and 2.4, we conclude that existing extensions of CML capturing user-item relations [20, 30] did not consider the item-item ones. Simultaneously, extensions capturing item-item relations [37] did not consider the user-item ones. In this paper, we argue that user-item and item-item relations can actually complement and influence each other. This would imply that *jointly* learning both relations could help reveal fine-grained user preferences, leading to better performances in recommendation.

In the remainder of this section, we leverage attention mechanisms and augmented memory networks to learn user-item and item-item relations in CML. Our approach is based on the assumption that *there exists a hierarchical structure* in different relation types and, more specifically, that *user-item relations are built on top of item-item relations*. The assumption is motivated by the idea that users might predominantly forge their opinion about a new item from this item's relations to others they already interacted with. For instance, some users' opinion on a new French rap song released on a music streaming service could be influenced by their former interactions with French and rap songs.

3.2 HLR

Following the notations of Section 2.2, we continue to denote by p_u and q_v the respective low-dimensional representation vectors of some user u and item v. These vectors are of dimension d, with $d \ll |\mathcal{U}|$ and $d \ll |\mathcal{V}|$.

3.2.1 Item-Item Modeling. Following the above assumption, our proposed HLR starts by modeling latent item-item relations. When recommending a candidate item to a user, relations between this item and the user's previous interactions with items will be cornerstones of HLR; they are represented as M in Figures 1 and 2. We model these relations through a *key-value memory network*, as proposed in [30]. At this stage, we introduce N *keys* vectors $k_i \in \mathbb{R}^d$, stacked up in the matrix $\mathbf{K} \in \mathbb{R}^{N \times d}$, and that HLR will learn. Here, N is a hyperparameter representing the number of keys which, in a nutshell, should represent different possible "types of relations". We also introduce the *memory* vectors $m_i \in \mathbb{R}^d$, stacked up in the matrix $\mathbf{M} \in \mathbb{R}^{N \times d}$. They are the corresponding *value vectors* of these relation types, as described thereafter.

Key Addressing **K**. First, HLR learns which relations exist between two items, as follows:

(1) Given an item pair (v_1, v_2), a Hadamard product[1] is used to extract the joint embedding between these items:

$$s_{v_1,v_2} = q_{v_1} \odot q_{v_2} \quad (2)$$

where the generated *joint embedding* vector $s_{v_1,v_2} \in \mathbb{R}^d$ is of the same dimension as q_{v_1} and q_{v_2}.

(2) Next, we use s_{v_1,v_2} to learn which relation types in **K** express the interaction between the two items through an attention vector $w_{v_1,v_2} \in \mathbb{R}^N$, in which each element $w_{i,v_1,v_2} \in \mathbb{R}$

[1]We note that other options are also viable to compute joint embedding vectors s_{v_1,v_2}, such as some vector averaging or a multi-layer perceptron processing q_{v_1} and q_{v_2}. In this work, we use Hadamard products for consistency with LRML [30], who underlined the empirical effectiveness of this simple approach.

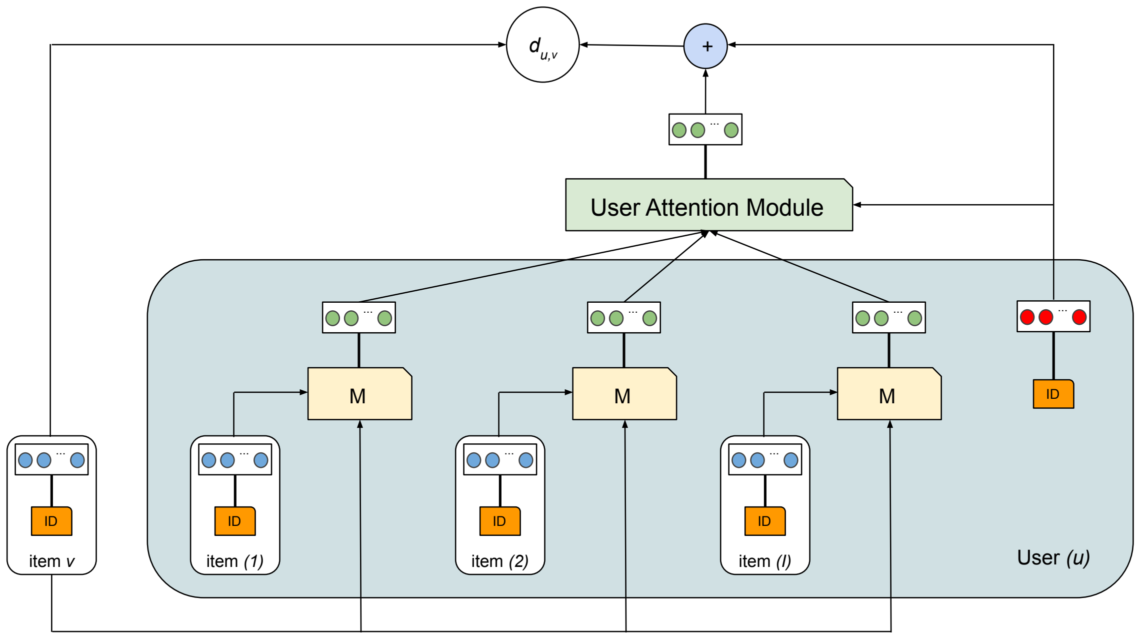

Figure 1: Architecture of HLR

verifies:

$$w_{i,v_1,v_2} = s^T_{v_1,v_2} k_i \tag{3}$$

where we recall that k_i are the d-dimensional key vectors from $\mathbf{K}$. Then, attention weights in w_{v_1,v_2} are normalized into a probability distribution, using the *softmax* function. This permits assessing the relative importance of each of the N relations types for a given item pair (v_1, v_2).

Latent Relations via Memories $\mathbf{M}$. We use the attention vector w_{v_1,v_2} to compute a *weighted* sum of *memory vectors* $m_i \in \mathbb{R}^d$ from the matrix $\mathbf{M} \in \mathbb{R}^{N\times d}$. Each m_i corresponds to the representation vector (to learn) for the relation type $k_i \in \mathbf{K}$. They are building blocks selected to form the *latent relation vector* $r_{v_1,v_2} \in \mathbb{R}^d$ for each item pair (v_1, v_2). Formally, the latent relation vector r_{v_1,v_2} of the (v_1, v_2) pair will be:

$$r_{v_1,v_2} = \sum_{i=1}^{N} w^T_{i,v_1,v_2} m_i. \tag{4}$$

3.2.2 *User-Item Modeling.* Then, we learn latent user-item relations, on top of the item-item ones. More specifically, HLR incorporates an *user attention module*, responsible for constructing an adaptive *latent relation vector* $\bar{r}_{u,v} \in \mathbb{R}^d$ between each user u and item v, as follows:

$$\bar{r}_{u,v} = \sum_{j\in\mathcal{N}_u} \alpha(u,v,j) r_{v,j}. \tag{5}$$

We recall that $\mathcal{N}_u \subseteq \mathcal{V}$ is the set of items that user u previously interacted with. In equation (5), $\alpha(u,v,j) \in [0,1]$ denotes the attention weight which aims at capturing the contribution of the item-item relation $r_{v,j}$ to the final user-item latent relation between user u and item v. More precisely, we define $\alpha(u,v,j)$ with the standard *softmax* function:

$$\alpha(u,v,j) = \frac{e^{p_u^T r_{v,j}}}{\sum_{h\in\mathcal{N}_u} e^{p_u^T r_{v,h}}}. \tag{6}$$

A large $\alpha(u,v,j)$ indicates a high influence of the relation $r_{v,j}$ in the final representation of the user-item relation $\bar{r}_{u,v}$. Finally, for recommendation tasks, *user-item similarities* for each user-item pair (u,v) will be evaluated as follows:

$$d(u,v) = ||p_u + \bar{r}_{u,v} - q_v||_2^2. \tag{7}$$

3.2.3 *Optimization.* As other CML-based methods [10, 20, 30], we minimize a *triplet loss* pulling closer matching user-item pairs in $\mathcal{S}$, and pushing away non-matching pairs up to an $m > 0$ distance. Equation (1) is redefined as:

$$\mathcal{L} = \sum_{(u,v,v^-)\in\mathcal{T}} \Big[||p_u + \bar{r}_{u,v} - q_v||_2^2 - ||p_u + \bar{r}_{u,v^-} - q_{v^-}||_2^2 + m \Big]_+ \tag{8}$$

where, as in equation (1), $\mathcal{T}$ denotes the ensemble of *triplets* (u,v,v^-) such as $(u,v) \in \mathcal{S}$ and $(u,v^-) \notin \mathcal{S}$. Intuitively, such a loss penalizes deviations of $p_u + \bar{r}_{u,v}$ from the vector q_v. In practice, latent vectors, keys $\mathbf{K}$ and memories $\mathbf{M}$ are learned by iteratively minimizing $\mathcal{L}$, using gradient descent [8], and a triplet sampling scheme described in Section 4.3.

3.3 HLR++

In our work, we also consider an extension of HLR, denoted HLR++. As illustrated in Figure 2, HLR++ adds an *item attention module* on top of HLR. It acts as a symmetric component w.r.t. the *user attention module* from Section 3.2.2. This module is responsible for additionally building item-item relations on top of user-item relations. It is motivated by the following postulate: some items might

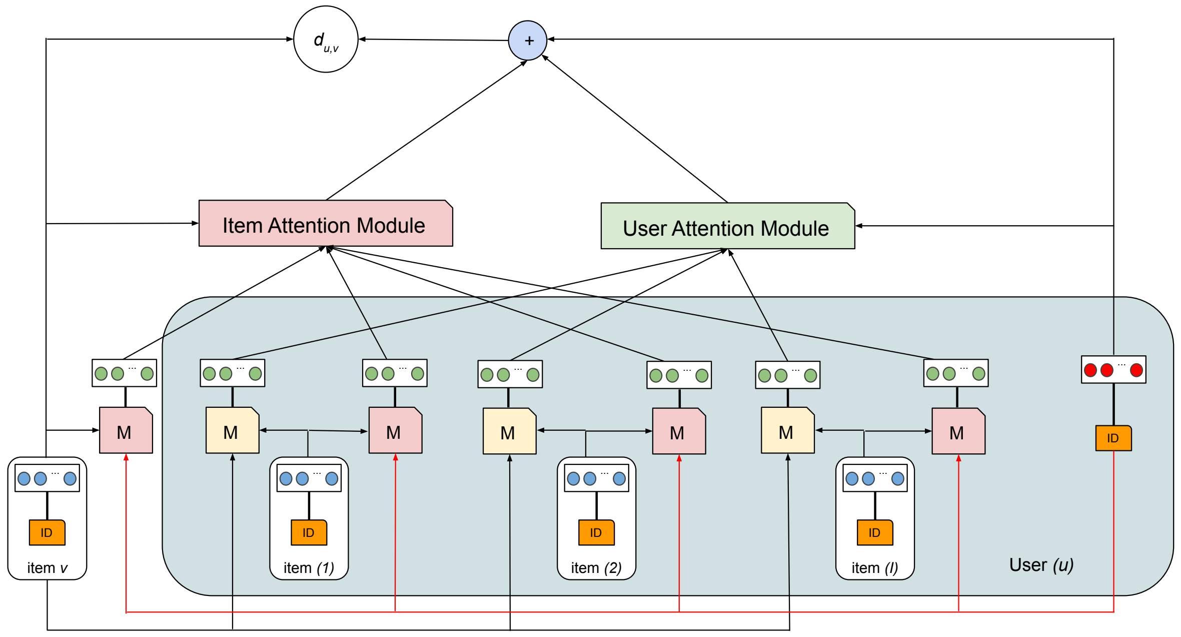

Figure 2: Architecture of HLR++

be related, not only because of their characteristics, but also because some users consider them together *in a specific context*. For instance, a user might listen to specific songs while running. In HLR++:

- First, we model item-item relations by memory networks, as in HLR.
- Then, the *item attention module*, together with the pre-existing *user attention module*, symmetrically construct two user-item relation vectors, say $\overline{r}^I_{u,v}$ and $\overline{r}^U_{u,v}$, as in Section 3.2.2 - but averaging over users of a same item for $\overline{r}^I_{u,v}$.
- Finally, the adaptive relation vector $\overline{r}_{u,v}$ between a user u and a candidate item v is derived by summing up the outputs of both user and item attention modules:

$$\overline{r}_{u,v} = \overline{r}^I_{u,v} + \overline{r}^U_{u,v}. \tag{9}$$

4 EXPERIMENTAL EVALUATION

In this section, we empirically evaluate and discuss the performance of our HLR and HLR++ models on recommendation.

4.1 Datasets

For our experimental evaluation, we consider four real-world datasets covering various domains:

- *MovieLens* [9]: this dataset gathers user ratings for movies, collected from a movie recommendation service. We binarize explicit rating data on a 5-star scale, keeping the ratings of 4 or higher as positive implicit feedback.
- *Echonest* [1]: the Echo Nest Taste Profile dataset contains user playcounts for songs from their Million Song Dataset (MSD). The playcount data is binarized by considering values of six streams or higher as implicit feedback.
- *Yelp* [34]: online reviews of businesses on a 5-star scale. Ratings of 4 or 5 are binarized and are implicit feedback.
- *Amazon book* [19]: a dataset of consumption records with reviews from *Amazon.com*. We use 5-star ratings from the book category. Ratings are binarized by keeping ratings of 4 and 5 as implicit feedback.

We implement a k-*core* pre-processing step on each dataset, consisting in a recursive filter until all users and items have at least k interactions, as proposed in [29]. Specifically, we set $k = 10$ for all datasets except for Yelp ($k = 5$). Table 1 reports some statistics on our four datasets, after this pre-processing step. They are ordered by *density* of the user-item interaction matrix, which is a commonly used factor to assess the difficulty of the recommendation task [26].

4.2 Evaluation Methodology

We assess the performance of HLR and HLR++ on these four datasets, for the top-K recommendation task introduced in Section 2.1, with $K = 10$. We closely follow the evaluation protocol proposed by Sun et al. [29]. Specifically, datasets are splitted as follows: 80% of user interactions are used for training, 10% for validation and the last 10% for test. Models must correctly recommend an ordered list of $K = 10$ items for which each user should interact with positively, in the validation or test set. We report five standard evaluation metrics: the Precision and the Recall, for prediction accuracy, as well as the Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) scores as measures of ranking quality. Last, as collaborative filtering is known to be prone to *popularity biases* [28], consisting in recommending more popular content, we furthermore report the *median popularity* of recommended items. The popularity of an item is defined as the number of users who interacted with it.

Table 1: Datasets statistics after pre-processing

Dataset	Number of users	Number of items	Number of ratings	Density	Median number of interactions per user
MovieLens	129 757	11 508	9 911 879	0.664%	98
Echonest	131 495	39 874	2 476 013	0.047%	24
Yelp	82 166	71 949	2 077 093	0.035%	23
Amazon	109 730	96 421	1 405 671	0.030%	16

4.3 Models

4.3.1 List of Baselines. We compare our proposed HLR and HLR++ to several baselines. Foremost, we consider MF-ALS [16], a standard MF method modeling the user-item relations using inner products, from the factorization of the training interaction matrix using *alternating least squares* (ALS) [17]. We also consider CML [11], the standard collaborative metric learning approach which does not include any relational translation between user and item vectors. Then, we consider the LRML and TransCF extensions of CML, described in Section 2.2, and capturing heterogeneities in *user-item* interactions. Last, we implement AdaCML [37], described in Section 2.3, which models *item-item* relations.

4.3.2 Implementation Details. We use Adam optimizer [15] for all models. To avoid reconstructing all triplets in $\mathcal{L}$, we adopt a standard *triplet sampling* strategy [11] that approximates losses: specifically, each training instance (u, v) is paired with a single negative sample (u, v^-) randomly picked among all items v^- with which the user u did not interact.

Models are trained for a maximum of 100 epochs. Parameters on the best epoch, i.e. parameters with minimum loss on the validation set, are used for final models. The embedding size d is fixed to 100 while the batch size is set to 1000. Other hyperparameters are tuned on the validation set by grid search: learning rates are amongst $\{0.0002, 0.0005, 0.00075, 0.001\}$; the number of memory slices N is in $\{5, 10, 20, 50\}$; the margin m is in $\{0.2, 0.5, 0.75, 1.0\}$.

Along with this paper, we publicly release our source code[2] to ensure the reproducibility of our results. Our repository includes an exhaustive table reporting the best hyperparameters for all models. It also reports running times on our machines, showing that HLR/HLR++ behave on par with the other item-item based model AdaCML.

4.4 Experimental Results

Table 2 reports average performances of all models, along with standard deviations over five runs to measure randomness in the training process. Overall, our approach reaches competitive results on three datasets out of four (MovieLens, Echonest and Yelp), while the AdaCML baseline ranks first on the Amazon dataset. We thereafter interpret these results.

4.4.1 On the Relevance of Relation Modeling. We observe that most methods incorporating *relation modeling* (LRML, AdaCML and our HLR and HLR++) significantly outperform those who do not (MF-ALS and CML) on the densest MovieLens dataset. AdaCML and, even more, HLR and HLR++, remain consistently superior on the sparser Echonest and Yelp datasets. For instance, HLR++ reaches top MRR and NDCG scores of 25.70% and 13.88% respectively on Echonest, vs. 15.82% and 8.85% for the standard CML. Our experiments also emphasize how the empirical benefits of relation modeling *tend to decrease on sparser datasets*. Specifically, it is narrowed on Yelp and, on Amazon, relation modeling did not bring any benefit (apart for AdaCML - see after). We postulate that, on such datasets with few feedback, modeling user-item or item-item relations by latent memories could actually bring noise and be more prone to overfitting.

4.4.2 On User-Item Modeling vs Item-Item Modeling. AdaCML, modeling only item-item relations, outperforms LRML and TransCF, modeling only user-item relations, on all four datasets (e.g. with a 10.62% MAP score on MovieLens, vs. 9.40% for LRML and 2.94% for TransCF on this same dataset). Learning latent relations between items thus seems to better complement the existing user-item interactions from original data, for this recommendation task. This highlights how user preferences for some items can indeed strongly depend on items they previously interacted with.

4.4.3 On Hierarchical Latent Relation Modeling. HLR and HLR++ outperform all baselines on the MovieLens, Echonest and Yelp datasets. This emphasizes the benefits of *jointly* modeling both user-item and item-item latent relations. This also tends to confirm the relevance of our assumption (i.e. that user-item relations are built on top of item-item relations) and therefore of our proposed hierarchical architecture. Nevertheless, we acknowledge that, on the very sparse Amazon dataset, HLR and HLR++ are outperformed by AdaCML. Our explanation is twofold. First of all, modeling user-item relations might be unhelpful - not to say harmful - on this last dataset. This would explain why LRML and TransCF also fail to surpass a standard CML, contrary to AdaCML that only focuses on item-item relations. Secondly, AdaCML adopts a relatively simple strategy (relying on unweighted averages of inner products) that, despite being surpassed by HLR and HLR++ on the three other datasets, might be less prone to overfitting in very sparse settings such as Amazon.

4.4.4 On HLR++ vs HLR. The additional *item attention module* of HLR++ provides visible gains compared to HLR on MovieLens and Echonest (e.g. with a 18.26% Recall on MovieLens, vs 17.63% for HLR). This tends to show that items relations could also be built upon user-item ones, for example when considering specific *contexts* (see Section 3.3). Regarding Yelp and Amazon, however, this item attention module does not show any improvement. These

[2]https://github.com/deezer/recsys21-hlr

Table 2: Top-10 item recommendation using HLR, HLR++ and baseline models.

Datasets	Metrics @10 (in %)	MF-ALS	CML	LRML	TransCF	AdaCML	**HLR**	**HLR++**
MovieLens	MAP	8.41 ± 0.09	7.82 ± 0.08	9.40 ± 0.05	2.94 ± 0.04	10.62 ± 0.08	10.97 ± 0.06	**11.36 ± 0.06**
	MRR	55.32 ± 0.53	48.70 ± 0.39	56.54 ± 0.24	22.07 ± 0.21	63.38 ± 0.24	66.01 ± 0.29	**67.85 ± 0.24**
	NDCG	20.05 ± 0.28	18.29 ± 0.12	20.28 ± 0.06	8.36 ± 0.07	23.47 ± 0.09	24.31 ± 0.13	**25.07 ± 0.10**
	Precision	15.22 ± 0.17	14.94 ± 0.17	16.37 ± 0.05	7.01 ± 0.06	18.47 ± 0.09	18.80 ± 0.11	**19.29 ± 0.07**
	Recall	13.98 ± 0.29	13.22 ± 0.08	13.53 ± 0.05	6.04 ± 0.05	16.96 ± 0.12	17.63 ± 0.19	**18.26 ± 0.25**
	Popularity	4771.6 ± 56.5	1244.8 ± 45.3	1034.6 ± 18.0	1495.5 ± 28.9	835.1 ± 19.7	**827.8 ± 20.3**	885.4 ± 7.4
Echonest	MAP	1.62 ± 0.03	1.97 ± 0.05	1.95 ± 0.06	1.82 ± 0.07	2.72 ± 0.05	3.04 ± 0.04	**3.28 ± 0.04**
	MRR	13.44 ± 0.20	15.82 ± 0.3	15.70 ± 0.33	14.38 ± 0.52	21.25 ± 0.33	23.92 ± 0.3	**25.70 ± 0.27**
	NDCG	7.12 ± 0.10	8.85 ± 0.15	8.60 ± 0.13	7.90 ± 0.26	11.78 ± 0.15	13.13 ± 0.16	**13.88 ± 0.13**
	Precision	3.13 ± 0.04	4.12 ± 0.07	4.02 ± 0.04	3.59 ± 0.09	5.48 ± 0.05	6.18 ± 0.07	**6.31 ± 0.04**
	Recall	8.48 ± 0.11	11.29 ± 0.16	10.74 ± 0.06	9.81 ± 0.27	14.89 ± 0.16	16.59 ± 0.28	**16.98 ± 0.19**
	Popularity	511.7 ± 2.2	51.0 ± 1.4	56.4 ± 0.5	55.0 ± 0.6	41.4 ± 0.8	**33.0 ± 0.0**	35.3 ± 0.4
Yelp	MAP	0.44 ± 0.01	0.52 ± 0.01	0.54 ± 0.02	0.15 ± 0.01	0.57 ± 0.02	**0.63 ± 0.02**	**0.63 ± 0.01**
	MRR	4.10 ± 0.12	4.75 ± 0.1	4.99 ± 0.17	1.43 ± 0.06	5.22 ± 0.17	**5.80 ± 0.20**	**5.74 ± 0.11**
	NDCG	2.57 ± 0.07	2.37 ± 0.08	2.45 ± 0.08	0.97 ± 0.04	2.58 ± 0.04	**2.83 ± 0.08**	**2.83 ± 0.04**
	Precision	1.13 ± 0.02	1.49 ± 0.04	1.51 ± 0.05	0.48 ± 0.02	1.60 ± 0.03	**1.68 ± 0.02**	**1.67 ± 0.03**
	Recall	3.64 ± 0.10	2.89 ± 0.13	2.90 ± 0.11	1.53 ± 0.07	3.13 ± 0.06	**3.35 ± 0.10**	**3.35 ± 0.09**
	Popularity	385.7 ± 1.1	45.2 ± 0.4	46.4 ± 0.5	44.2 ± 0.4	**40.2 ± 0.4**	68.0 ± 1.1	73.2 ± 1.3
Amazon	MAP	0.98 ± 0.03	1.12 ± 0.05	1.06 ± 0.03	0.57 ± 0.01	**1.33 ± 0.03**	1.11 ± 0.03	1.03 ± 0.03
	MRR	8.27 ± 0.20	9.45 ± 0.45	9.03 ± 0.19	4.93 ± 0.09	**11.02 ± 0.23**	9.29 ± 0.4	8.71 ± 0.26
	NDCG	3.84 ± 0.11	4.72 ± 0.18	4.45 ± 0.11	2.61 ± 0.06	**5.49 ± 0.13**	4.59 ± 0.20	4.39 ± 0.14
	Precision	2.10 ± 0.05	2.64 ± 0.07	2.47 ± 0.06	1.48 ± 0.04	**2.99 ± 0.05**	2.54 ± 0.08	2.46 ± 0.02
	Recall	4.11 ± 0.17	5.47 ± 0.17	5.09 ± 0.13	3.21 ± 0.08	**6.32± 0.16**	5.26 ± 0.26	5.13 ± 0.20
	Popularity	222.7 ± 2.7	33.0 ± 0.0	35.6 ± 0.5	**26.6 ± 0.5**	31.0 ± 0.0	64.0 ± 1.8	47.6 ± 0.8

datasets might be less prone to contextual interactions, w.r.t. MovieLens (on movies) and Echonest (on music). Indeed, music consumption highly depends on the listening context which may be linked to emotions and mood, time of the day, season, event, location, weather, and user activity [18, 25]. In the case of movies, watching context can also be considered an important aspect of consumption [2]. However, in domains such as business products and books, the influence of these contexts to user behaviors could be lesser, considering that consumption times are higher than for movies and songs (several days to several weeks instead of several minutes or a couple of hours). Such a result is also in line with our previous findings on the limits of complexifying models when dealing with very sparse feedback data.

4.4.5 On Popularity. We observe that MF-ALS is more prone to *popularity biases* [28] than CML methods on all datasets. HLR and HLR++ tend to recommend *less popular* content w.r.t. other CML extensions on MovieLens and Echonest, which is often viewed as a desirable property [24, 28]. On the contrary, they recommend slightly *more popular* content on Yelp and Amazon; they nonetheless preserve significant reductions in popularity biases w.r.t. standard MF methods (e.g. with a 47.6 median popularity on Amazon for HLR++, above the 26.6 of TransCF but way below the 222.7 of MF-ALS).

5 CONCLUSION

In this paper, we introduced a hierarchical CML model that, contrary to previous efforts, jointly models latent user-item and item-item relations from implicit data. Our approach leverages memory-based attention networks and builds upon the assumption that user-item relations are built on top of the item-item ones in a hierarchical manner. We emphasized the empirical effectiveness of our approach at capturing the complex interests of users for recommendation, outperforming popular alternatives in terms of prediction accuracy and ranking quality on several real-world datasets. On the other hand, we also pointed out that relation modeling does not always improve standard CML, especially on sparse datasets with very few interaction data. This leaves the door open for future research on such a challenging setting, where we believe simple approaches might surpass complex models in providing more reliable recommendations.

REFERENCES

[1] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference.* 909–916.

[2] Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli, Alfonso Miola, and Giuseppe Sansonetti. 2011. Context-aware movie recommendation based on signal processing and machine learning. In *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation.* 5–10.

[3] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Gutiérrez A. 2013. Recommender Systems Survey. *Know.-Based Sys.* 46 (2013), 109–132.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *Advances in Neural Information Processing Systems.* 1–9.

[5] Léa Briand, Guillaume Salha-Galvan, Walid Bendada, Mathieu Morlon, and Viet-Anh Tran. 2021. A Semi-Personalized System for User Cold Start Recommendation on Music Streaming Apps. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.*

[6] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 714–722.

[7] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *Proceedings of the 24th International Conference on Artificial*

Intelligence. 2069–2075.

[8] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep Learning.* MIT Press.

[9] F Maxwell Harper and Joseph A Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. on Int. Int. Sys.* 5, 4 (2015), 1–19.

[10] Xiangnan He, Hanwang Zhang Lizi Liao, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *In Proceedings of the 26th International Conference on World Wide Web.* 173–182.

[11] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the World Wide Web Conference.* 193–201.

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining.* 263–272.

[13] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: Factored Item Similarity Models for Top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 659–667.

[14] Mohammad Khoshneshin and Nick W. Street. 2010. Collaborative Filtering via Euclidean Embedding. In *Proceedings of the ACM Conference on Recommender Systems.* 87–94.

[15] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proc. of the 3rd Int. Conference on Learning Representations.*

[16] Yehuda Koren. 2008. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 426–434.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[18] Kaminskas Marius and Francesco Ricci. 2012. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review* 6, 2-3 (2012), 89–119.

[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th ACM SIGIR Conference on Research and Development in Information Retrieval.* 43–52.

[20] Chanyoung Park, Donghyun Kim, Xing Xie, and Hwanjo Yu. 2018. Collaborative Translational Metric Learning. In *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM).*

[21] Bibek Paudel, Sandro Luck, and Abraham Bernstein. 2019. Loss Aversion in Recommender Systems: Utilizing Negative User Preference to Improve Recommendation Quality. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining.*

[22] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence.*

[23] J Ben Schafer, Joseph A Konstan, and John Riedl. 2001. E-commerce Recommendation Applications. *Data Mining and Knowledge Discovery* 5, 1 (2001), 115–153.

[24] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current Challenges and Visions in Music Recommender Systems Research. *International Journal of Multimedia Information Retrieval* 7, 2 (2018), 95–116.

[25] Volokhin Sergey and Eugene Agichtein. 2018. Understanding music listening intents during daily activities with implications for contextual music recommendation. In *Proceedings of the 2018 Conference on Human Information Interaction & Retrieval.* 313–316.

[26] Joao Felipe Silva, Natanael Moura Junior, and Luiz Caloba. 2018. Effects of Data Sparsity on Recommender Systems based on Collaborative Filtering. In *Proceedings of the 2018 International Joint Conference on Neural Networks.* 1–8.

[27] Kun Song, Feiping Nie, Junwei Han, and Xuelong Li. 2017. Parameter Free Large Margin Nearest Neighbor for Distance Metric Learning. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence.*

[28] Harald Steck. 2011. Item Popularity and Recommendation Accuracy. In *Proceedings of the Fifth ACM Conference on Recommender systems.* 125–132.

[29] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. 2020. Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems.* 23–32.

[30] Yi Tay, Anh Tuan Luu, and Cheung Hui Siu. 2018. Latent Relational Metric Learning via Memory-Based Attention for Collaborative Ranking. In *Proceedings of the 2018 World Wide Web Conference.* 729–739.

[31] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. A Boosting Metric Learning Approach in Hyperbolic Space for Recommender Systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining.* 609–617.

[32] Viet-Anh Tran, Romain Hennequin, Jimena Royo-Letelier, and Manuel Moussallam. 2019. Improving Collaborative Metric Learning with Efficient Negative Sampling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1201–1204.

[33] Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research* 10 (2009), 207–244.

[34] Yelp. 2014. Dataset available on: https://www.yelp.com/dataset/. (2014).

[35] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning based Recommender System: A Survey and New Perspectives. *Comput. Surveys* 52, 1 (2019), 1–38.

[36] Shuai Zhang, Lina Yao, Yi Tay, Xiwei Xu, Xiang Zhang, and Liming Zhu. 2018. Metric Factorization: Recommendation beyond Matrix Factorization. *arXiv preprint arXiv:1802.04606.*

[37] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Junhua Fang, Lei Zhao, Victor S. Sheng, and Zhiming Cui. 2019. AdaCML: Adaptive Collaborative Metric Learning for Recommendation. In *Proc. of the International Conference on Database Systems for Advanced Applications.* 301–316.

[38] Xiao Zhou, Danyang Liu, Jianxun Lian, and Xing Xie. 2019. Collaborative Metric Learning with Memory Network for Multi-Relational Recommender Systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence.* 4454–4460.