Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking

Yi Tay
Nanyang Technological University
Singapore
ytay017@e.ntu.edu.sg

Luu Anh Tuan
Institute for Infocomm Research
Singapore
at.luu@i2r.a-star.edu.sg

Siu Cheung Hui
Nanyang Technological University
Singapore
asschui@ntu.edu.sg

ABSTRACT

This paper proposes a new neural architecture for collaborative ranking with implicit feedback. Our model, LRML (*Latent Relational Metric Learning*) is a novel metric learning approach for recommendation. More specifically, instead of simple push-pull mechanisms between user and item pairs, we propose to learn latent relations that describe each user item interaction. This helps to alleviate the potential geometric inflexibility of existing metric learning approaches. This enables not only better performance but also a greater extent of modeling capability, allowing our model to scale to a larger number of interactions. In order to do so, we employ a augmented memory module and learn to attend over these memory blocks to construct latent relations. The memory-based attention module is controlled by the user-item interaction, making the learned relation vector specific to each user-item pair. Hence, this can be interpreted as learning an exclusive and optimal relational translation for each user-item interaction. The proposed architecture demonstrates the state-of-the-art performance across multiple recommendation benchmarks. LRML outperforms other metric learning models by 6% – 7.5% in terms of Hits@10 and nDCG@10 on large datasets such as Netflix and MovieLens20M. Moreover, qualitative studies also demonstrate evidence that our proposed model is able to infer and encode explicit sentiment, temporal and attribute information despite being only trained on implicit feedback. As such, this ascertains the ability of LRML to uncover hidden relational structure within implicit datasets.

KEYWORDS

Collaborative Filtering; Recommender Systems; Neural Networks

ACM Reference Format:
Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3178876.3186154

1 INTRODUCTION

The modern age is a world of information overload. The explosion of information, also referred to as the era of big data, is a huge motivator for the research and development of practical recommender systems. Generally, the key problem that these systems are aiming to solve is the inevitable conundrum of 'too much content, too little time' that is commonly faced by users. After all, there are easily million of movies, thousands of songs and hundreds of books to choose from at any given time. An effective recommender system ameliorates this problem by delivering the most relevant content to the user.

Our work is targeted at recommender systems that operate on implicit data (e.g., clicks, likes, bookmarks) and are known as collaborative filtering (CF) systems [27]. In this setting, Matrix Factorization (MF) remains as one of the most popular baselines which has inspired a considerable number of variations [7, 8, 13, 25]. The general idea of MF is as follows: Users and items are represented as a matrix and subsequently factorized into latent components which can also be interpreted as modeling the relationships between users and items using the inner product. As such, this allows missing values to be inferred which provides an approximate solution to the recommendation problem.

Recently, Hseih et al. [9] revealed the potential implications pertaining to the usage of inner product to model user-item relationships. Their argument is constructed upon the fact that inner product violates the triangle inequality which is essential to model the fine-grained preferences of users. Instead, the authors proposed a metric-based learning scheme that minimizes the distance between user and item vectors (p and q) of positive interactions. Simultaneously, this also learns user-user similarity and item-item similarity in vector space. As evidence to their assertions, their proposed algorithm, the collaborative metric learning (CML) algorithm [9] demonstrates highly competitive performance on many benchmark datasets.

Despite the success of CML, it faces several weaknesses. Firstly, the scoring function of CML is clearly *geometrically restrictive*. Given a user-item interaction, CML tries to fit the pair into the same point in vector space. Considering the *many-to-many* nature of the collaborative ranking problem, enforcing a good fit in vector space can be really challenging from a geometric perspective especially since the optimal point of each user and item is now a single point in vector space. Intuitively, this tries to fit a user and all his interacted items onto the same point, i.e., geometrically congestive and inflexible. While it is possible to learn user-user and item-item similarity clusters, this comes at the expense of precision and accuracy in ranking problems especially pertaining to large datasets whereby there can be millions of interactions. Secondly and by taking a more theoretically grounded angle, CML is an *ill-posed algebraic system* [36] which further reinforces and aggravates the problem of geometric inflexibility. A proof and more details are described in the related work section.

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186154

In this work, we propose a flexible and adaptive metric learning algorithm for collaborative filtering and ranking. Our model, LRML (Latent Relational Metric Learning) learns adaptive relation vectors between user and item interactions, finding an optimal translation vector between each interaction pair. Needless to say, our work is highly inspired by recent advances in NLP which include the highly celebrated word embeddings [18] and knowledge graph embeddings [2, 15, 32] which popularized the concept of semantic translation in vector space. In our proposed approach, we assume that there exist a latent relational structure within the implicit interaction data and therefore, we aim to model the latent relationships between users and items by inducing relation vectors.

Overall, our key intuition can be described as follows: For each user and item interaction, we learn a vector r that explains this relationship, i.e., the relation vector r connects the user vector to the item vector. Ideally, this vector r should capture the hidden semantics between each implicit interaction and is learned over an auxiliary memory module via a neural attention mechanism. The auxiliary memory module can be interpreted as a memory store of concepts in which, upon linear combination, constructs a relation vector. The content addressing of this memory module is user and item dependent, which ensures sufficient flexibility in geometric space. Apart from the clear benefits of an interpretable attention module, LRML can also be considered as an improvement to the CML algorithm [9]. Our approach solves the geometric inflexibility problem by means of adaptive (user-item specific) translations in vector space. This allows for a greater extent of flexibility and modeling capability in metric space which enables our model to scale to larger datasets with easily millions of interactions.

1.1 Our Contributions

Motivated by the success of deep learning, both generally and in the field of recommender systems, our ideas are materialized in the form of a neural network architecture that leverages the recent advancements of neural attention mechanisms and augmented memory modules [28]. Overall, the prime contributions of this paper are:

- We present LRML (Latent Relational Metric Learning), a novel, end-to-end neural network architecture for collaborative filtering and ranking on implicit interaction data. For the first time, we adopt *user and item specific* latent relation vectors to model the relationship between user-item interactions.
- We propose a novel *Latent Relational Attentive Memory* (LRAM) module in order to generate the latent relation vectors. The LRAM module provides improvements in terms of flexibility and modeling capability of the algorithm. Moreover, the neural attention also gives greater insight and interpretability of the model.
- We evaluate our proposed LRML on **ten** publicly available benchmark datasets. This includes large, web-scale datasets like Netflix Prize and MovieLens20M. Our proposed approach demonstrates highly competitive results on all datasets, outperforming not only CML but many other strong baselines such as NeuMF [7]. Moreover, on large datasets, we obtain $6\% - 7.5\%$ gain in performance over CML and other models.
- We performed extensive qualitative analysis. Upon inspection of the attention weights, our proposed LRML is capable of inferring explicit information such as ratings (e.g., 1-5 stars), temporal and item attribute information despite being only trained on implicit binary data. This ascertains the capability of LRML in unraveling hidden latent structure within seemingly non-relational datasets.

2 BACKGROUND

Our work is concerned with collaborative filtering[1] with implicit feedback. We first formulate the problem and discuss the existing algorithms that are aimed at solving this problem. Then, we elaborate on the potential weaknesses of the collaborative metric learning algorithm.

2.1 Implicit Collaborative Filtering

The task of implicit collaborative filtering is concerned with learning via implicit interaction data, e.g., clicks, bookmarks, likes, etc. Let **P** be the set of all users and **Q** be the set of all items. The problem of implicit CF can be described as follows:

$$y_{ui} = \begin{cases} 1 \text{ , if interaction <user,item> exists} \\ 0 \text{ , otherwise} \end{cases} \quad (1)$$

where $\mathbf{Y} \in \mathbb{R}^{|P| \times |Q|}$ is the user-item interaction matrix. Implicit CF models the interaction of users and items and on that note, it is good to bear in mind that a value of 0 does not necessarily imply negative feedback. In most cases, the user is unaware of the existence of the item which forms the cornerstone of the recommendation problem, i.e., estimating the scores of the unobserved entries in **Y**. Across the past decade, Matrix Factorization (MF) techniques are highly popular algorithms for collaborative filtering and have spurred on a huge number of variations [7, 8]. Since MF does not belong to the core focus of our work, we omit the technical descriptions of MF for the sake of brevity and refer interested readers to [7, 9] for more details.

2.2 Collaborative Metric Learning (CML)

CML [9] is a recently incepted algorithm for CF and has, despite its simplicity, demonstrated highly competitive performance on several benchmarks. The key intuition is that CML operates in metric space, i.e., it minimizes the distance between each user-item interaction in Euclidean space. The scoring function of CML is defined as:

$$s(p, q) = \| p - q \|_2^2 \quad (2)$$

where p, q are the user and item vectors respectively. CML learns via a pairwise hinge loss, which is reminiscent of the Bayesian Personalized Ranking (BPR) [25]. CML obeys the triangle inequality which, according to the authors, is a prerequisite for fine-grained fitting of users and items in vector space.

CML, however, is not without flaws. As mentioned, the scoring function of CML is *geometrically restrictive* since the objective function tries to fit each user-item pair into the same point in vector space. Unfortunately, this intrinsic geometric inflexibility causes

[1] In this paper, we use the terms collaborative filtering and collaborative ranking interchangeably.

adverse repercussions when the dataset is large or dense since CML tries to force all of a user's item interactions onto the same point. Secondly and by taking a more theoretically grounded angle, we show that CML is an *ill-posed algebraic system* [36] which further reinforces and aggravates the problem of geometric inflexibility. The following proof elaborates on this issue.

Theorem 2.1. *The objective function of CML: $s(p, q) = \|p - q\|_2^2$ can be considered as an ill-posed algebraic system when there is a large number of interactions.*

Proof. Let d be the dimensions of vectors p and q. From an algebraic perspective, each user-item interaction can be regarded as the equation $p - q = 0$. By considering $p_i - q_i$, where i is the index of the vectors p and q, the number of equations for each interaction is d. Let N be the total number of interactions, the total number of equations is therefore $N \times d$. On the other hand, the number of free variables is only $(|P| + |Q|) \times d$. Since $N \ggg d(|P| + |Q|)$ in most settings, CML is an ill-posed algebraic system. □

Since it is not uncommon for implicit recommendation datasets to contain millions of interactions while having significantly much lesser unique items and users, we can consider, from a mathematical perspective, that CML proposes an ill-posed algebraic system. This introduces instability when training and optimizing the objective function of CML.

2.3 Translating in Vector Space

Our proposed approach, LRML, ameliorates the flaws of CML by means of adaptive translation. Since our adaptive translation is learned as a weighted representation (over an augmented memory via neural attention), this introduces an extremely large number of possibilities for the user and item vectors to be translated in vector space. More specifically, the attention vector (learned via a softmax function) learns a continuous weighted representation of the augmented memory. As such, this significantly expands the flexibility of the metric learning algorithm. In LRML, the user vector is now adaptively translated based on the target item (and vice versa). As such, this allows LRML to avoid the above-mentioned flaws of CML, and enables more precise and fine-grained fitting in vector space.

Translating in vector space takes inspiration from NLP and in particular, reasoning over semantics (knowledge graphs). In this area, a highly seminal work by Bordes et al. (TransE) [2] proposed translations in vector space to model the relationships between entities in a knowledge graph. Word embeddings [18] are also known to exhibit *semantic translation* in vector space whereby the relationships between two words can be explained by a relation vector. The domain of CF that models users and items, and represents them as an interaction matrix is highly related to graph and network embeddings [17, 21, 29].

To the best of our knowledge, our work is the first work that extends the 2D structure of user-item CF into a 3D structure by assuming a latent relational (3D) structure. Intuitively, this can be also interpreted as inducing a latent knowledge graph from the user-item interaction graph.

Figure 1 depicts the key difference between LRML and CML - while CML tries to place user and item into the same spot in vector space, LRML learns to fit user and item with adaptive, trainable latent vectors. More specifically, LRML learns an optimal translation between each user-item interaction. Recall in Section 2.2, we have previously established that CML suffers from instability (due to being an ill-posed algebraic system) along with geometric inflexibility, i.e., the push-pull effects from too many interactions. In order to alleviate this weakness, our proposed approach adopts attentive and adaptive user-item specific translations that benefit from the vast number of possibilities of learning weighted (linearly combined) representations.

Finally, we note that another translation-based recommendation model, TransRec was recently proposed by He et al. [6] in which the authors proposed to use translations to model sequential data. While TransRec also utilizes the translation principle, LRML is a completely different model. Firstly, TransRec learns translations for sequential recommendation, e..g, the 2nd item a user interacts with is represented by a translation of the first. Secondly, the overall goals of LRML is different, i.e., LRML utilizes translations for flexible and adaptive metric learning. Thirdly, LRML uses neural attention to learn latent relations, which is also an feature that is absent in TransRec.

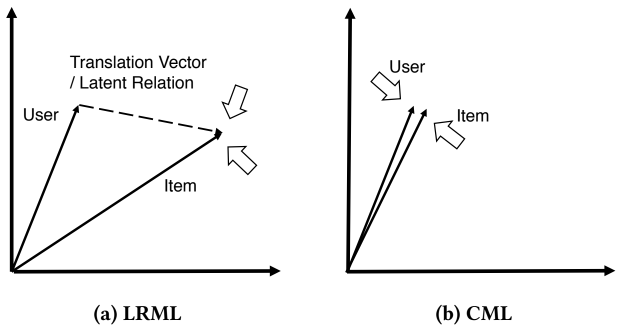

Figure 1: Geometric Comparisons of Latent Relational Metric Learning (LRML) and CML (Collaborative Metric Learning) for Modeling User-Item Relationships in Metric Space.

2.4 Deep Learning

In this section, we provide some preliminaries about deep learning for recommendation.

2.4.1 Deep Learning for Recommendation. In the recent years we can easily observe the emerging numbers of neural network models that have been designed for a diverse range of recommendation tasks. Notably, Recurrent neural networks [38] and convolutional neural networks [30] have been exploited for sequence aware recommendations. There is also an emerging line of work focusing on representation learning using reviews, e.g., Deep Co-operative Networks (DeepCoNN) [43]. A recent work, the Multi-Pointer Co-Attention Networks [35] is the state-of-the-art review-based CF model that uses pointer-based attention for representation learning. Autoencoder based models [14, 42] have also been proposed for CF. In the more closely related domain of collaborative filtering on implicit feedback, Neural Matrix Factorization (NeuMF) [7] is a recent state-of-the-art deep learning model that learns the interaction function between user and item using deep neural networks. NeuMF is a combined framework that concatenates the inner-product-based MF with a multi-layered perceptron (MLP). A comprehensive review can be found at [41].

2.4.2 Neural Attention. Our work borrows inspiration from the recent advances in deep learning. Specifically, LRML uses a neural attention mechanism over an augmented memory module to generate latent vectors. Neural attention mechanisms are popular in the fields of computer vision [20, 39] and NLP [16, 23, 26, 31, 34] and are known to improve performance and interpretability of deep learning models. The key idea of attention is to learn a weighted representation across multiple samples (or embeddings), reducing noise and selecting more informative features for the final prediction. Attention operates using a *softmax* function, which converts the attention vector into a probability distribution. Subsequently, this vector is then used to learn a weighted sum of a sequence of vectors.

Notably, attention mechanisms have been also recently adopted for collaborative filtering problems particularly for content-based recommendations such as multimedia recommendation [3]. However, the novelty of our model lies in the difference whereby our model adopts neural attention to generate latent relation vectors over an augmented memory module. This is fundamentally different from content-based attention models which learn to attend over features and learn to predict. While the key idea of attentive selection is similar, the goal of our model is to find hidden relational structure by leveraging attention mechanisms. Moreover, the inner mechanism of our proposed LRAM is highly reminiscent of end-to-end memory networks [28] and key-value memory networks [19] which are the competitive models for question answering, machine comprehension and aspect-aware sentiment analysis [33].

3 OUR PROPOSED MODEL

In this section, we introduce LRML, our novel deep learning recommendation architecture. The overall model architecture is described in Figure 2. LRML aims to model user and item pairs using relation vectors. This is what we refer to as the translational principle, i.e., $p + r \approx q$. Note that the relation vector r is what separates our model from simple metric learning approaches like CML which operate via $p \approx q$. Let us begin with a simple high-level overview of our model:

(1) Users and items are converted to dense vector representations using an Embedding Layer (a look-up layer). p and q are the user and item vectors respectively.
(2) Given p and q, a relation vector r is generated using a neural attention mechanism over an augmented memory matrix $\mathbf{M}$. The relation vector, r, is a weighted representation over a trainable LRAM module. r is dependent on user and item, and is learned to best explain the relationship between user and item.
(3) Our model optimizes for $\| p + r - q \| \approx 0$ using pairwise ranking (hinge loss) and negative sampling.

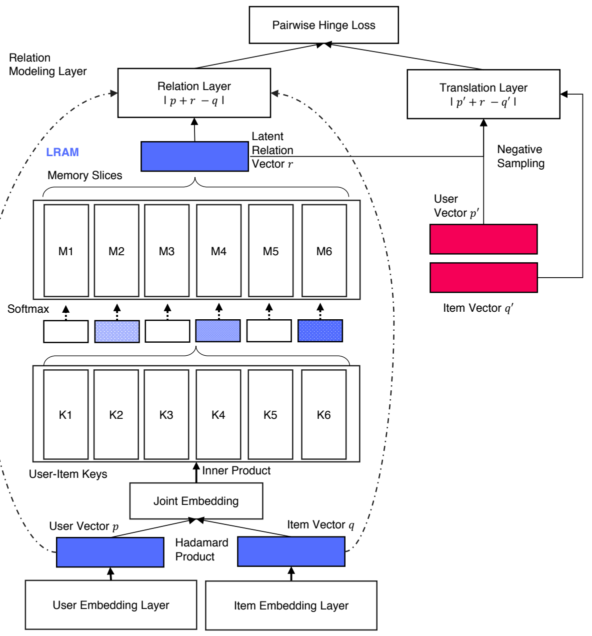

Figure 2: Illustration of our proposed LRML architecture, an end-to-end differentiable neural architecture. LRML is characterized by its key-addressed LRAM module which learns user-item specific relation vectors. The size of the memory N=6 slices in this example.

3.1 Embedding Layer

LRML accepts a user-item pair $< user,item >$ as an input. Inputs of users and items are represented as vectors encoded via one-hot encoding corresponding to a unique index key belonging to each user and item. At the embedding layer, this one-hot encoded vector is converted into a low-dimensional real-valued dense vector representation. In order to do so, this one hot vector is multiplied with the embedding matrices $\mathbf{P} \in \mathbb{R}^{d \times |U|}$ and $\mathbf{Q} \in \mathbb{R}^{d \times |I|}$ which store the user and item embeddings respectively. d is the dimensionality of the user and item embeddings while $|U|$ and $|I|$ are the total number of users and items respectively. The output of this layer is a pair of embeddings $< \vec{p}, \vec{q} >$ which are the user and item embeddings respectively.

3.2 LRAM - Latent Relational Attentive Memory Module

One of the primary goals of our LRML model is to induce latent relations between user-item pairs. However, explicit semantic relations between user-item pairs are not available in traditional CF. As such, we introduce the *Latent Relational Attentive Memory* (LRAM) module. The LRAM module is a centralized memory store in which latent relations are built upon. The memory matrix of the LRAM module is represented as $\mathbf{M} \in \mathbb{R}^{N \times d}$ where d is the dimensionality of the user-item embeddings and N is a user-specified hyperparameter that controls the expressiveness and capacity of the LRAM module. In matrix $\mathbf{M}$, we refer to each row slice $m_i \in \mathbb{R}^d$ as a memory slice. The input to LRAM is a user-item pair $< p, q >$. The LRAM module returns the vector r of equal dimensionality as p and q.

3.2.1 Joint User-Item Embedding. Given the user-item pair, $< \vec{p}, \vec{q} >$, the LRAM module first applies the following steps to learn a joint embedding of users and items:

$$s = p \odot q \tag{3}$$

where $\odot$ is simply the Hadamard product (or element-wise multiplication). The generated vector $s \in \mathbb{R}^d$ is of the same dimension of p and q. Note that while other functions such as the multi-layered perceptron $MLP(p, q)$ are also viable, we found that a simple Hadamard product performs better.

3.2.2 User-Item Key Addressing. Next, using the joint user-item embedding, we aim to learn an attention vector a. The attention vector is learned from $\mathbf{K} \in \mathbb{R}^{N \times d}$ which we refer to as the key matrix. Each element of the attention vector a can be defined as:

$$a_i = s^T k_i \tag{4}$$

where $k_i \in \mathbf{K} \in \mathbb{R}^{N \times d}$ and the generated vector $a \in \mathbb{R}^d$ is of the same dimensions of p, q and s. In order to normalize a to a probability distribution, we can simply use the Softmax function:

$$Softmax(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}. \tag{5}$$

Since our attention mechanism utilizes the softmax function, it ensures that our network is end-to-end differentiable.

3.2.3 Generating Latent Relations via Memory-based Attention. Finally, in order to generate the latent relation vector r, we use the attention vector a to generate a weighted representation of $\mathbf{M}$, i.e., adaptively selecting relevant pieces of information from the memory matrix $\mathbf{M}$.

$$r = \sum_i a_i m_i \tag{6}$$

The output of the LRAM module is a user and item specific latent relation vector r. The latent relation vector is a weighted representation of $\mathbf{M}$. Intuitively, the memory matrix $\mathbf{M}$ can be interpreted as a store of conceptual building blocks that can be used to describe the relationships between users and items. The mechanism design of the LRAM module is inspired by Memory Networks and can also be interpreted as neural attentions which give our model improved interpretability. Note that the LRAM module is part of LRML and is trained end-to-end. Finally, the total number of parameters added by the LRAM module is merely $2 \times N \times d$ parameters and since typically we set $N < 100$ in our experiments, the parameter cost incurred by the LRAM module is negligible.

3.3 Optimization and Learning

In this section, we introduce the final layer of the network, the objective function and the regularization employed in our training scheme. LRML is end-to-end differentiable since it utilizes soft attention over the LRAM module. As such, we are able to simply train it via stochastic gradient descent (SGD) methods.

3.3.1 Relational Modeling Layer. For each user-item pair p and q, the scoring function is defined as:

$$s(p, q) = || \, p + r - q \, ||_2^2 \tag{7}$$

where r is the latent relation vector constructed from the LRAM module and $||.||_2^2$ is essentially the L2 norm of the vector $p + r - q$. Intuitively, this score function penalizes any deviation of (p+r) from the vector q.

3.3.2 Objective Function. LRML adopts the pairwise ranking loss or hinge loss for optimization. For each positive user-item pair $< p, q >$, we sample a corrupted pair which we denote as $< p', q' >$. Similar to the positive example, the corrupted pair of user and item goes through the same user and item embedding layer respectively. The pairwise ranking / hinge loss is defined as follows:

$$L = \sum_{(p,q) \in \Delta} \sum_{(p',q') \notin \Delta} max(0, s(p, q) + \lambda - s(p', q')) \tag{8}$$

where Δ is the set of all user-item pairs, λ is the margin that separates the golden pairs and corrupted samples. $max(0, x)$ is also known as the *relu* function. Note that we use the same (generated) latent relation vector for the negative example. This is motivated by our early empirical results whereby performance was much better over generating a separate relation vector for the negative example.

3.3.3 Regularization. Finally, we apply regularization by normalizing all user and item embeddings to be constrained within the Euclidean ball. At the end of each mini-batch, we apply a constraint of $\|p_*\|_2 \leq 1$ and $\|q_*\|_2 \leq 1$ for regularization and preventing overfitting. In order to enforce this, we can manually project all embeddings to the unit ball either at the beginning or after each training iteration.

4 PERFORMANCE EVALUATION

In this section, we evaluate our proposed LRML against other state-of-the-art algorithms. Our experimental evaluation is designed to answer several research questions (**RQs**).

- **RQ1:** Does LRML outperform other baselines and state-of-the-art methods for collaborative ranking?
- **RQ2:** How does the relative performance of LRML and CML differ across different dataset sizes?
- **RQ3:** What is the scalability and runtime of LRML compared to other baselines?
- **RQ4:** What is the LRAM module learning? Are we able to derive qualitative insights about the inner workings of LRML?
- **RQ5:** What do the relation vectors represent? Are they meaningful?

4.1 Datasets

In the spirit of experimental rigor, we conduct our evaluation across a wide spectrum of datasets.

- **Netflix Prize** - Since the entire Netflix Prize dataset is extremely large, we construct a subset of the famous Netflix Prize dataset. Specifially, we only considered movie-item ratings from the year 2005 and filtered users who had less than 100 interactions.

- **MovieLens** - A widely adopted benchmark dataset[2] for collaborative filtering in the application domain of recommending movies to users. Specifically, we use two configurations of this benchmark dataset, namely MovieLens1M and MovieLens20M [5].
- **IMDb** - A movie recommendation dataset obtained from IMDb that was introduced in [4].
- **LastFM** - This dataset contains social networking, tagging, and music artist listening information from Last.fm online music system[3].
- **Books** - This is a book recommendation dataset that was used in [44].
- **Delicious** - This dataset contains social networking, bookmarking, and tagging information from a set of 2K users from Delicious Social Bookmarking System[4]. This dataset, along with the lastFM dataset, originated from the Hetrec 2011 workshop[5].
- **Meetup** - An event-based social network[6]. We use the datasets provided by [22] which include event-user pairs from NYC.
- **Twitter** - This is a check-in dataset constructed by [40] which contains users and their check-ins. There are two subsets of this dataset, namely Twitter (WW) and Twitter (USA).

In total, we evaluate our proposed algorithm on **ten** different datasets with diverse sizes and interaction densities, i.e., the percentage of non-zero values in the user-item interaction matrix. For all datasets, with the exception of the Netflix Prize dataset, we ensured that each user has at least 20 interactions. The statistics of all datasets are reported in Table 1.

Dataset	Interactions	#Users	#Items	% Density
Netflix Prize	44M	75K	13K	4.5
MovieLens20M	16M	53K	27K	1.1
MovieLens1M	1M	6K	4K	4.2
IMDb	117K	0.8K	114K	0.13
LastFM	92K	1.9K	175K	0.28
Books	285K	7.4K	291K	0.01
Delicious	43K	1.7K	69K	0.36
Meetup	11K	2.6K	16K	0.26
Twitter (WW)	18K	4K	36K	0.01
Twitter (USA)	171K	4K	36K	0.12

Table 1: Statistics of all datasets used in our experimental evaluation.

4.2 Baselines

In this section, we introduce the key baselines for comparison against our proposed LRML.

- **Bayesian Personalized Ranking** (BPR) [25] is a strong CF baseline that minimizes $\sum_i \sum_{j,k} -\log \sigma(p_i^T q_j - p_i^T q_k) + \lambda_v \|u_i\|^2 + \lambda_q \|q_j\|^2$, where (p_i, q_j) is a positive interaction and (p_i, q_k) is a negative sample.
- **Matrix Factorization** (MF) is a standard baseline for CF that models the relationship between user and item using inner products. We use the generalized version from [7] which scores user item pairs with $s(p, q) = \sigma(h^T (p \odot q))$.
- **Multi-layered Perceptron** (MLP) is the baseline neural architecture proposed in [7] in which the authors proposed to use multiple layers of nonlinearities to model the relationships between users and items.
- **Neural Matrix Factorization** (NeuMF) [7] is the state-of-the-art unified framework combining MF with MLP. NeuMF concatenates the output of MF and MLP, and uses a regression layer to predict the user item rating. Note that NeuMF uses separate embedding representations of users and items for MF and MLP.
- **Collaborative Metric Learning (CML)** [9] can be considered as the baseline of our model which does not include relational translations between user and item vectors.

Since CML and NeuMF have surpassed many other baselines such as WMF [10], eALS [8] and Factorization Machines [24], we do not further report them. Additionally, for fair comparison and due to scalability issues, we do not use WARP [37] for both CML and LRML.

4.3 Evaluation Protocol and Metrics

Our evaluation protocol follows He et al. [7] very closely. Similarly, we adopt the *leave-one-out* evaluation protocol, i.e., the testing set comprises the last item of all users. If there are no timestamps available in the dataset (e.g., *Delicious* and *LastFM*), then the test sample is randomly sampled. A single item from each user is also sampled to form the development set. Since it is too time consuming to rank all items for every user, we randomly sampled 100 items that have no interactions with the target user and ranked the test item with respect to these 100 items. This is in concert with many works [1, 7, 8, 25]. Since our problem is essentially formulated as learning-to-rank, we judge the performance of our model based on the popular and widely adopted standard metrics used in information retrieval and recommender systems: **normalized discounted cumulative gain** (nDCG@10) [11] and **Hit Ratio** (H@10). Intuitively, the nDCG@10 metric is a position-aware ranking metric while H@10 metric simply considers whether the ground truth is ranked amongst the top 10 items. For more detailed explanations, we refer readers to [7].

4.4 Implementation Details

We implemented all models in *TensorFlow*[7] on a Linux machine. For tuning the hyperparameters, we select the model that performs best on the development set based on the nDCG metric and report the result of that model on the test set. Model parameters are saved every 50 epochs. All models are trained until convergence, i.e., if the performance (nDCG metric) on the development set does not improve after 50 epochs. Models are trained for a maximum of 500 epochs. For large datasets like *MovieLens20M* and *Netflix Prize*, we stop the training at 100 epochs. The dimensionality of

[2]https://grouplens.org/datasets/movielens/
[3]http://last.fm
[4]http://www.delicious.com
[5]https://grouplens.org/datasets/hetrec-2011/
[6]https://www.meetup.com/

[7]https://www.tensorflow.org/

	BPR		MLP		MF		NEUMF		CML		LRML	
	H@10	nDCG@10	H@10	nDCG@10	H@10	nDCG@10	H@10	nDCG@10	H@10	nDCG@10	H@10	nDCG@10
Netflix	<u>48.67</u>	<u>31.97</u>	33.77	22.34	47.07	30.25	32.27	22.59	46.12	29.48	**53.71**	**35.78**
MovieLens20M	69.68	46.68	75.81	<u>54.38</u>	72.98	49.01	75.82	54.37	<u>77.64</u>	53.01	**84.47**	**61.52**
MovieLens1M	<u>72.37</u>	53.33	48.59	33.11	68.87	49.17	68.61	50.65	72.16	<u>54.13</u>	**73.97**	**54.53**
IMDb	4.62	4.23	4.11	3.79	5.26	4.89	4.87	4.55	<u>9.47</u>	<u>7.16</u>	**11.92**	**8.45**
LastFM	<u>20.73</u>	<u>13.58</u>	7.36	3.75	18.17	12.02	14.89	9.61	19.75	12.03	**21.71**	**14.38**
Books	22.07	16.13	12.89	10.03	15.61	10.75	12.54	7.65	<u>25.86</u>	<u>18.70</u>	**26.72**	**19.43**
Delicious	78.50	77.78	77.05	73.80	78.91	78.09	78.79	78.11	<u>79.31</u>	<u>78.43</u>	**80.31**	**79.01**
Meetup	44.91	36.08	31.33	23.19	<u>47.23</u>	<u>38.29</u>	32.76	25.79	47.04	36.64	**50.19**	**40.48**
Twitter (WW)	76.39	75.27	53.33	35.68	<u>76.93</u>	75.43	<u>76.66</u>	74.86	75.86	74.72	**78.92**	**77.17**
Twitter (USA)	75.88	75.04	77.91	76.23	76.47	75.62	70.75	69.79	<u>78.30</u>	<u>76.50</u>	**79.36**	**77.85**

Table 2: Experimental results on ten benchmark datasets. Best performance is in boldface and second best is underlined. LRML achieves best performance on all datasets, outperforming many strong neural baselines. Improvement is much larger on large datasets such as Netflix Prize or MovieLens20M.

user and item embeddings d is tuned amongst $\{20, 50, 100\}$. The number of batches B is tuned amongst $\{10, 100, 1000\}$. The minimum number of batches for NetflixPrize and MovieLens20M is 100 in order to fit into the GPU RAM. We optimize all models using the Adam optimizer [12]. The learning rate for all models are tuned amongst $\{0.01, 0.005, 0.001\}$. For models that minimize the hinge loss, the margin λ is tuned amongst $\{0.1, 0.2, 0.25, 0.5\}$. For NeuMF and MLP models, we follow the configuration and architecture proposed in He et al. [7], i.e., 3 fully-connected layers with a pyramid architecture. However, for fair comparison of all models, we do not use pretrained MLP and MF models in the NeuMF model since this effectively acts as an ensemble classifier. For LRML, the number of memory slices in $\mathbf{M}$ is tuned amongst $N = \{5, 10, 20, 25, 50, 100\}$. For simplicity, each training instance is paired with only a single negative sample. All embeddings and parameters are normally initialized with a standard deviation of 0.01.

For most datasets and baselines, we found that the following hyperparameters work well: learning rate= 0.001, number of batches $B = 10$ and $\lambda = 0.2$. A larger embedding size always performs better, i.e., $d = 100$. The size of LRAM is dataset dependent. We found that setting $N = 20$ works well for most datasets (performance does not degrade going beyond 50 but does not improve either). However, we found that setting $N = 100$ works better on large datasets such as *Netflix Prize* and *MovieLens20M*.

4.5 Experimental Results

The empirical results of our proposed model and baselines on 10 benchmark datasets are reported in Table 2. Our proposed LRML performs extremely competitively on all datasets and obtain the best performance on both nDCG@10 and H@10 metrics on **all** datasets. This answers **RQ1**, showing that our proposed LRML is capable of effective collaborative ranking. Moreover, the ranking of many of the competitor baselines is fluctuating across datasets as we see the second best performance is scattered amongst different models.

4.5.1 Comparison against CML. In general, LRML outperforms CML on all datasets on both H@10 and nDCG@10 metric. We would like to draw the reader's attention to the two datasets, namely *Netflix Prize* and *MovieLens20M* datasets in which LRML obtained a clear margin in performance gain over the competitor models. This ascertains our earlier claim about the flaws of CML (not being able to scale to large datasets) and empirically proves the advantages of our proposed approach. Specifically, LRML outperforms CML by performance gains about 7.5% on *MovieLens20M* and about 6% on *Netflix Prize* on the nDCG@10 metric. The performance gains on the hit ratio (H@10) metric is also similarly high. When the dataset is smaller, the performance gains are less distinct. For example, the performance gain in MovieLens20M is much larger than in MovieLens1M. The performance gains on smaller datasets range from a marginal $1\% - 2\%$, (e.g., *Books* and *Delicious*) to reasonably large, e.g., $3\% - 4\%$ on the *Meetup* or *Twitter (WW)* datasets. As such, the concluding findings pertaining to the comparison of LRML and CML can be drawn as follows: On large datasets, the performance gain of LRML over CML is large. However, on smaller datasets, LRML at least performs equally well or sometimes reasonably better. This answers **RQ2** on the effect of dataset size on relative performance of LRML and CML. Our experimental evidence shows that our proposed LRML is effective and ascertains our usage of adaptive translations in metric learning.

4.5.2 Comparison against Other Baselines. Pertaining to the performance of the other baselines, we found that the performance of MF and BPR is extremely competitive, i.e., both MF and BPR outperform CML on several datasets. The performance of MLP, on the other hand, seem to perform reasonably well only on *MovieLens20M* and performs horribly on most datasets. Note that we also tried a non-pyramid architecture but that did not improve the performance. The performance of the model NeuMF (that combines MLPs with MF) is often better than vanilla MLP but falls short of MF in most cases. Notably, NeuMF performs reasonably well on *MovieLens20M*, *Netflix Prize* and *MovieLens1M*. This could possibly mean that the usage of dual embedding spaces (one for MF and one for MLP) might be overfitting on the smaller datasets.

4.5.3 Comparison on Runtime. Figure 3 reports the runtime (seconds taken to run a single epoch) of all models on *Netflix Prize* and *MovieLens20M*. We make several observations. First, the difference in runtime between LRML and CML is quite insignificant, i.e., LRML only spends $\approx 10s - 15s$ extra per epoch which is only a $5\% - 10\%$ increase in runtime on both datasets. On the other hand, it is still faster than models such as NeuMF and MLP. Notably, this is also contributed by the fact that MLP and NeuMF are point-wise models which do not pair negative samples with positive samples during training. Next, we also compare the runtime of LRML with different N (LRAM size) values and found that there is only minimal observable difference in runtime with $N = 50$ or $N = 100$. This was probably made insignificant by the highly optimized GPU operations and also due to the fact that the size of the matrix-vector operations in LRAM is relatively small. To answer **RQ3**, we have shown that LRML only incurs a slight computation cost over CML.

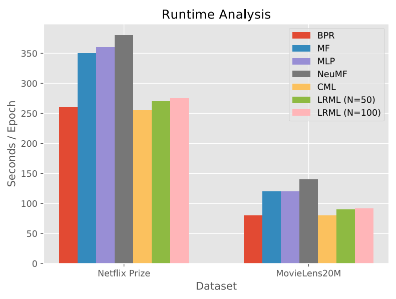

Figure 3: Runtime (seconds/epoch) of all models on Netflix Prize and MovieLens20M. Experiments were run with batch size of 100 on a Nvidia P100 GPU. LRML only incurs a small computational cost over CML. *(Best viewed in color.)*

5 DISCUSSION AND ANALYSIS

In this section, we derive qualitative insights regarding our proposed model. This section describes the discoveries that we have made while trying to understand and gain some intuition behind the performance of LRML.

5.1 RQ4: What is the LRAM module learning?

A key advantage to neural attention mechanisms is an improved interpretability since we are able to visualize the weighted importance of each memory slice with respect to any given attribute value v. This helps us to understand how the model is learning. Specifically, we investigate the attributes of *explicit rating information* and *explicit temporal information*, and show empirically via visualisation that the LRAM model learns to encode these attributes. Note that both attributes are not provided to our model at training time. In this experiment, the following steps were taken:

(1) First, we categorized all user-item pairs (p, q) according to the target attribute value v.
(2) Using (p, q) as an input, we generated the attention vector a for each user-item pair. Recall that this attention vector[8] is a probability distribution that depicts how much the model is looking at each memory slice of the LRAM module.
(3) For each attribute class $c_i \in v$, we take the mean attention vector for all user-item pairs in the category.
(4) We visualise the mean attribute vector of each attribute class to observe the correlation between attribute class and which memory slice LRML is looking at.

[8]It is good to note that, at initialization, this attention vector looks at all memory slices *almost* equally irregardless of v.

5.1.1 *LRAM Encodes Explicit Rating Information.*

On datasets like MovieLens1M, explicit ratings (1-5 stars) exist but are not provided to LRML. Surprisingly, we empirically discovered that, despite being only trained on implicit interactions, explicit rating information is actually being encoded in LRAM. Figure 4 shows the mean attention vector (i.e., a) for each rating class (1-5).

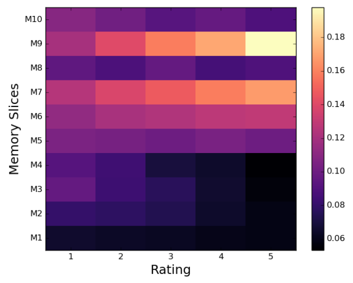

Figure 4: Attention weights over LRAM for user-item pairs of different ratings on MovieLens1M. LRML is able to model explicit rating information despite being only trained on implicit data. *(Best viewed in color.)*

The color scale represents the strength of the attention weights and each column of Figure 4 represents the mean attention vector for each rating class. As such, we are able to observe patterns and trends across different ratings by looking at the rows (from left to right). For example, the mean attention vector of rating=1 is the first vertical slice in Figure 4 and the intensity denoted by the color scale represents the attention weights.

Clearly, we observe that there is a pattern between the explicit rating score and the memory slice in which LRML is looking at. We observe that slices M2-M4 are mostly associated with bad ratings (1-2 stars) while having a high attention weight over M6, M7 and M9 signifies a good rating (4-5 stars). Moreover, there is a correlation between how much the model looks at M6, M7 and M9 and the explicit rating score. As such, it seems we are able to infer explicit rating scores solely based on how much our model is looking at each memory slice.

We believe this can be explained as follows: The goal of LRML is to find a latent relational structure between the user and item interactions. As such, while LRML is trying to assign relations between users and items via neural attention, it has learned to identify and model explicit rating sentiment from the implicit structure of the dataset.

5.1.2 *LRAM encodes temporal information.*

The second discovery is that the LRAM module actually encodes temporal information. Similar to ratings, timestamps are available on the MovieLens1M dataset but are not used to train the model. To facilitate clear visualization, we binned the timestamps into 10 separate bins in ascending order. Figure 5 shows the visualized attention weights of LRAM with respect to time. Similar to explicit rating scores, we notice that certain memory slices model the chronological order of user-item interactions. On M8, we see that the intensity of the attention weights increase along with time, i.e., by viewing the row M8 from left to right, we can observe an increasing attention weight on M8 based on the intensity scale. Moreover, the converse is true for M6 which when observed from left to right, it decreases in intensity instead. In short, there is a clear pattern in which we can quite safely ascertain that LRAM has learned to encode temporal information. Once again, it is worthy to note that LRML was not given any temporal information to begin with.

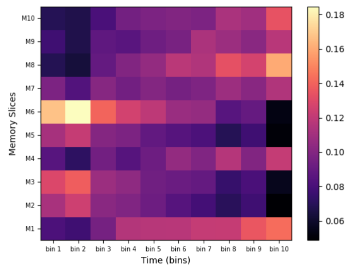

Figure 5: Attention weights over LRAM for user-item pairs for different time bins on MovieLens1M. A clear trend is found in M6 which shows that the LRAM module encodes temporal information even when no such information is provided during training. *(Best viewed in color.)*

5.2 RQ5: What do the relation vectors represent? Are they meaningful?

For each user-item pair in the test set, we generated the latent relation vector r. Next, we computed the cosine similarity between the relation vectors of all user-item pairs and selected the user-item pairs in which the cosine similarity between their relation vectors is the highest. Intuitively, this is to investigate **if similar user-item pairs might have similar relation vectors**. In order to characterize user-item pairs, we selected attributes that are available in the MovieLens1M dataset. The user attributes provided include Age, Job and Gender while only category and movie title were provided for the items. Once again, note that these attributes were not provided to our model during training. For each user-item pair, we computed the attribute matches with respect to the user-item pair with the **closest** relation vector, e.g., if <User1,Item1> and <User2,Item2> have the most similar relation vector, we compute the matches between **each** attribute of both user-item pairs. For example, we check for matches between User1 and User2 within the list of attributes such as user age, user gender and user job and item category. Ideally, the model should learn a similar relation vector for similar user-item pairs. In order to determine if the result is significant, we computed the probability of a match by random chance taking into consideration the distribution of attributes. Table 3 reports the results of this experiment.

User-Item Attribute	Match (%)	Random (%)	Diff (%)
User Age	25.81	22.17	3.64
User Job	20.06	13.71	6.35
User Gender	65.73	59.43	6.30
Item Category	51.19	43.87	7.32
Category AND Job	15.07	5.56	9.51

Table 3: Matches between user-item attributes of user-item pairs with the closest relation vector. Relation vectors encode user-item attributes without being trained on them.

We observe that the percentage of getting an attribute match is often higher than that of random chance which might signal that *similar user-item pairs have similar relation vectors*. In particular, the item category (movie genre) has the most prominent improvement over random chance (7.32%) individually while a considerable percentage of user-item pairs (15.07%) have an exact match of item category **and** job. This is 9.51% more than random chance. Additionally, we also found that (by manual inspection) there is a prominent number of job-category matches such as *(programmer, thriller)* and *(technician/engineer, thriller)*. This is intuitive since engineers and programmers can be considered as semantically related professions.

Overall, we believe that, the user and movie attributes characterize the behavior of users and therefore, there might be a hidden structure within simple implicit interaction data. By imposing and inducing architectural bias, our model learns to capture this fine-grained behavior even from simple implicit feedback data.

6 CONCLUSION

In this paper, we proposed LRML (Latent Relational Metric Learning), a novel attention-based memory-augmented neural architecture that models the relationship between users and items in metric space using latent relation vectors. LRML demonstrates the state-of-the-art performance on 10 publicly available benchmark datasets for implicit collaborative ranking. Empirical results show that relative improvement is significantly greater when the dataset is large, e.g., Netflix Prize and MovieLens20M, which is due to the geometric inflexibility of the CML algorithm. Additionally, LRML leverages the hidden and latent relational structure in the implicit user-item interaction matrix. Via qualitative analysis of the attention weights, we discovered that explicit rating information, temporal information and even item attributes are encoded within the LRAM module and relation vectors even when these information are not provided during training.

7 ACKNOWLEDGEMENTS

The authors would like to thank anonymous reviewers of WWW 2018 for their time and effort in reviewing this paper. We also thank Matt Yang and Kang KyungPhil for feedback on some typo errors in previous preprint versions.

REFERENCES

[1] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 1341–1350. https://doi.org/10.1145/3038912.3052694

[2] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* 2787–2795.

[3] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 335–344. https://doi.org/10.1145/3077136.3080797

[4] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 193–202. https://doi.org/10.1145/2623330.2623758

[5] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19. https://doi.org/10.1145/2827872

[6] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. ACM, New York, NY, USA, 161–169. https://doi.org/10.1145/3109859.3109882

[7] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 173–182. https://doi.org/10.1145/3038912.3052569

[8] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. 549–558. https://doi.org/10.1145/2911451.2911489

[9] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 193–201. https://doi.org/10.1145/3038912.3052639

[10] 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 (ICDM 2008), December 15-19, 2008, Pisa, Italy*. 263–272. https://doi.org/10.1109/ICDM.2008.22

[11] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.* 20, 4 (2002), 422–446. https://doi.org/10.1145/582415.582418

[12] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[13] 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, Las Vegas, Nevada, USA, August 24-27, 2008*. 426–434. https://doi.org/10.1145/1401890.1401944

[14] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. 305–314. https://doi.org/10.1145/3097983.3098077

[15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.* 2181–2187.

[16] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[17] Yao Ma, Zhaochun Ren, Ziheng Jiang, Jiliang Tang, and Dawei Yin. 2018. Multi-Dimensional Network Embedding with Hierarchical Structure. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18).*

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.* 3111–3119.

[19] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* (2016).

[20] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada.* 2204–2212.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 701–710. https://doi.org/10.1145/2623330.2623732

[22] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. 2015. A general graph-based model for recommendation in event-based social networks. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. 567–578. https://doi.org/10.1109/ICDE.2015.7113315

[23] Minh C. Phan, Aixin Sun, Yi Tay, Jialong Han, and Chenliang Li. 2017. NeuPL: Attention-based Semantic Matching and Pair-Linking for Entity Disambiguation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. 1667–1676. https://doi.org/10.1145/3132847.3132963

[24] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*. 995–1000. https://doi.org/10.1109/ICDM.2010.127

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*. 452–461.

[26] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* (2015).

[27] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. 285–295. https://doi.org/10.1145/371920.372071

[28] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-To-End Memory Networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada.* 2440–2448.

[29] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. 1067–1077. https://doi.org/10.1145/2736277.2741093

[30] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18).*

[31] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Learning to Attend via Word-Aspect Associative Fusion for Aspect-based Sentiment Analysis. (2017). arXiv:arXiv:1712.05403

[32] Yi Tay, Anh Tuan Luu, and Siu Cheung Hui. 2017. Non-Parametric Estimation of Multiple Embeddings for Link Prediction on Dynamic Knowledge Graphs. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 1243–1249.

[33] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2017. Dyadic Memory Networks for Aspect-based Sentiment Analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017.*

[34] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. A Compare-Propagate Architecture with Alignment Factorization for Natural Language Inference. *CoRR* abs/1801.00102 (2018). arXiv:1801.00102 http://arxiv.org/abs/1801.00102

[35] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. *CoRR* abs/1801.09251 (2018). arXiv:1801.09251 http://arxiv.org/abs/1801.09251

[36] A. N. Tikhonov and V. Y. Arsenin. 1977. Solutions of ill-posed problems. In *Mathematics of Computation 32(144)*. 491âĂŞ491.

[37] Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning* 81, 1 (2010), 21–35. https://doi.org/10.1007/s10994-010-5198-3

[38] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. 2017. Recurrent Recommender Networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 495–503.

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2048–2057.

[40] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. 2013. Who, where, when and what: discover spatio-temporal topics for twitter users. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*. 605–613. https://doi.org/10.1145/2487575.2487576

[41] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435* (2017).

[42] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Auto-encoders. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 957–960. https://doi.org/10.1145/3077136.3080689

[43] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 425–434.

[44] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*. 22–32. https://doi.org/10.1145/1060745.1060754