Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources

Yongfeng Zhang[1*], Qingyao Ai[1*], Xu Chen[2], W. Bruce Croft[1]
[1]College of Information and Computer Sciences, University of Massachusetts Amherst, MA 01003
[2]School of Software, Tsinghua University, Beijing, 10084, China
{yongfeng,aiqy,croft}@cs.umass.edu,xu-ch14@mails.tsinghua.edu.cn

ABSTRACT

The Web has accumulated a rich source of information, such as text, image, rating, etc, which represent different aspects of user preferences. However, the heterogeneous nature of this information makes it difficult for recommender systems to leverage in a unified framework to boost the performance. Recently, the rapid development of representation learning techniques provides an approach to this problem. By translating the various information sources into a unified representation space, it becomes possible to integrate heterogeneous information for informed recommendation.

In this work, we propose a *Joint Representation Learning* (JRL) framework for top-N recommendation. In this framework, each type of information source (review text, product image, numerical rating, etc) is adopted to learn the corresponding user and item representations based on available (deep) representation learning architectures. Representations from different sources are integrated with an extra layer to obtain the joint representations for users and items. In the end, both the per-source and the joint representations are trained as a whole using pair-wise learning to rank for top-N recommendation. We analyze how information propagates among different information sources in a gradient-descent learning paradigm, based on which we further propose an *extendable* version of the JRL framework (eJRL), which is rigorously extendable to new information sources to avoid model re-training in practice.

By representing users and items into embeddings offline, and using a simple vector multiplication for ranking score calculation online, our framework also has the advantage of fast online prediction compared with other deep learning approaches to recommendation that learn a complex prediction network for online calculation.

KEYWORDS

Recommender Systems; Representation Learning; Heterogeneous Information Processing; Top-N Recommendation

1 INTRODUCTION

For many years, user to item numerical ratings have been the most frequently used user-item interactions for personalized recommendation, and they have served as the underpinning of most Matrix Factorization (MF)-based [22] Collaborative Filtering (CF) [32] algorithms. Recently, researchers have found or argued that information sources beyond ratings are extremely helpful in user/item profiling and personalized recommendation, but these information sources come in very different and heterogeneous forms, e.g., textual reviews, visual images, or even sound tracks.

Different types of feedbacks describe different aspects of user preferences – numerical ratings indicate users' overall attitude towards a product; textual reviews are able to express user opinions towards various product features [9, 27, 42]; and product images reveal users' preferences on different visual fashions [18, 28]. Intuitively, heterogeneous information sources can be complementary with each other for user profiling, which can help to promote personalized recommendation when integrated properly. However, the nature of heterogeneity makes it difficult to fuse ratings, reviews, images, and other information sources in a unified way.

Previous work on this topic falls into the category of hybrid recommendation [6, 7], which generally includes two research lines – the hybridization of algorithms, and the hybridization of heterogeneous information sources. The first research line attempts to integrate different recommendation techniques for improved performance, e.g., integrating content- [30] and CF-based [14] algorithms. Different algorithms can be assembled by various strategies such as weighting, switching, mixing, cascading, or meta-level hybridization [7, 16]. However, these approaches put less attention on leveraging heterogeneous information sources. They also require significant efforts on model design and selection because different strategies are needed for different algorithms, especially when content-based methods are involved.

More recently, the second research line has attracted much attention from the research community, and algorithms are proposed to leverage the power of different information sources. One trend is to augment numerical ratings with textual reviews for recommendation [8], which includes topic modeling [3, 26, 27, 34], sentiment analysis [9, 13, 39, 42–44], and (deep) neural network [2, 36, 41, 45] approaches for review modeling. Researchers have also jointly considered ratings and images for product recommendation [17, 18, 28], video signals for key frame recommendation [10], audio signals for music recommendation [35, 37], and knowledge bases for movie/book recommendation [40]. Yet existing approaches are usually restricted to limited information sources (e.g., rating plus another information source), or require the pre-existence of domain knowledge for recommendation.

Fortunately, recent advances on representation learning [5] has shed light on this problem, which makes it possible to learn the representations of very different information sources in a shared representation space. This further makes it possible to design a

CIKM'17, November 6–10, 2017, Singapore.
 ISBN 978-1-4503-4918-5/17/11…$15.00 [*equal contribution]
DOI: https://doi.org/10.1145/3132847.3132892

unified recommendation framework that is capable of providing recommendation lists based on a variety of user feedbacks.

In this work, we focus on ranking-based recommendation to construct top-N recommendation lists. Referring to principles of multi-view machine learning [33], we propose a Joint Representation Learning (JRL) approach as a general framework for recommendation, which builds representation learning on top of pair-wise learning to rank for top-N recommendation (Figure 1). In this framework, each kind of information source is considered as a *view*, and each piece of information therein (e.g., a piece of review) is considered as an *entity*. In each view, entities are represented as embeddings based on available (deep) representation learning architectures. These entity embeddings (shown as gray intermediate embeddings in Figure 1) are connected to users and items based on the observed user-item interactions, which give us the user and item representations in the corresponding view. Representations from each view are further mapped to a shared semantic space with a full connection to obtain the integrated user/item representations, which are multiplied to produce the ranking scores for pair-wise learning to rank. For model learning, we randomly select an item that has no interaction with the target user to construct positive-negative item pairs, and parameters and representations in the whole framework are learned simultaneously in an end-to-end manner on the ranking loss.

To avoid model re-training when adding new information sources (i.e., views), we analyze the information propagation among views in the JRL framework. With this, we further propose an *extendable* version of the JRL framework (called eJRL), which theoretically allows integrating new views without re-training the existing views – an extremely favorable property in real-world systems where frequent model retraining on large-scale data is not always practical.

In the following, we first review related work in Section 2. In Section 3 we propose the JRL approach for recommendation, including the modeling of each view, the unified framework, and the top-N recommendation strategy. We examine the framework extendability and propose eJRL in Section 4, and provide the experimental results and analyses in Section 5. Section 6 concludes the work and points out some of the future research directions.

2 RELATED WORK

Researchers and practitioners have known for years that incorporating richer information source beyond numerical ratings can promote recommendation performance. This leads to research on hybrid recommendation techniques [6, 7]. At the very beginning, hybrid recommendation mostly referred to the integration of both content-based [30] and CF-based [14] techniques for recommendation, where the integration can come in various forms such as weighting, switching, mixing, etc, depending on the available content profiles and meta-recommendation algorithms.

More recently, many Web applications have accumulated a large amount of heterogeneous information sources about users, items, and their interactions, which helped to extend the concept of hybrid recommendation to the integration of different information sources.

A popular research line is the joint modeling of numerical ratings and textual reviews for recommendation. The earliest approach constructs manually defined ontologies from free-text reviews to

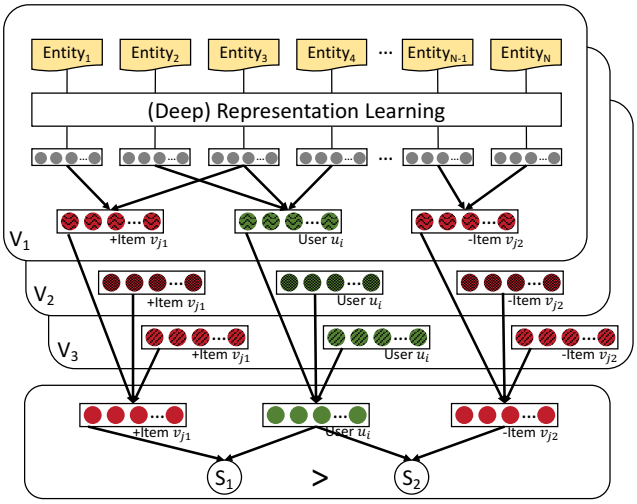


Figure 1: Overview of the Joint Representation Learning (JRL) framework for top-N recommendation. Each type of information source is denoted as a view V_i, and user-item interactions in each view are adopted to learn the corresponding user/item representations. In this figure, a positive item is one that the user previously purchased, so the user and item are linked to the common entity of this interaction (i.e., rating, review, or image of the product); instead, a negative item is one that the user did not purchase before, thus they are not linked to any common entity.

help rating prediction [1], but it requires extensive human labor on domain-related ontology construction. To address the problem, researchers have been relying on automatic review analysis techniques for recommendation. Specifically, [3, 26, 27, 34] conduct topic discovery from the reviews and integrate it with the latent factors from ratings to estimate user preferences for recommendation, which achieved better performance in rating prediction. Furthermore, [13, 39] proposed leveraging probabilistic graphical models to include more flexible prior knowledge for review modeling. To capture the local semantic information in reviews, [41] proposed to integrate traditional matrix factorization with word2vec [29] for user profiling and rating prediction.

Users may express explicit sentiments toward products or aspects in reviews. To capture this important signal of user preference, researchers have also applied sentiment analysis on textual reviews for recommendation. In particular, [42] uses multi-matrix factorization to generate explainable recommendations based on phrase-level sentiment analysis, and [9] further captures user interests on multi-category product features with a learning to rank manner. Furthermore, [4] proposed a Sentiment Utility Logistic Model to recommend not only items but also item aspects to provide better experience for the users, and [43] jointly considered sentiment, aspect, and regional features based on geographical topic modeling for location-aware recommendation.

Beyond numerical ratings and textual reviews, visual images reveal users' visual preference towards different items. For example, [28] proposed image-based recommendation to provide co-purchase recommendations, and [18] proposed a Visual-based Bayesian Personalized Ranking (VBPR) approach for top-N recommendation.

In [10], the authors jointly modeled time-synchronized comments and key frame images for personalized key frame recommendation in video websites.

A related research trend in recent years is leveraging deep learning for recommendation. For example, [25, 38] adopted denoising auto-encoders for recommendation, [46] developed a neural autoregressive approach for collaborative filtering, and [19, 20] generalized matrix factorization and factorization machines for neural collaborative filtering. On considering information sources beyond ratings, [2, 36, 41, 45] adopted deep textual modeling on reviews for recommendation, [35, 37] leveraged deep audio embeddings for music recommendation, [40] incorporated knowledge base embedding for recommendation, and [15] studied cross-domain recommendation with deep user modeling based on web search queries.

Though achieving better performance against modeling ratings alone, previous models (including deep approaches) are usually limited to pre-selected information sources or domain knowledge, thus researchers have to develop different models for different types of user-item interactions. Recent promising advances on representation learning [5] shed light on this problem. With well established representation learning theories on texts [24, 29], images [23], audios [21], and many others, we can conduct joint representation learning on heterogeneous information sources in a shared space for more informed recommendation. Furthermore, by building representation learning on top of pair-wise learning to rank techniques [31], we are able to achieve highly promoted top-N recommendation performance, which is closely related to the business values in real-world recommender systems.

3 JOINT REPRESENTATION LEARNING

In this section, we describe the Joint Representation Learning (JRL) framework for recommendation. We first provide a simple overview of the framework, and then adopt three heterogeneous information sources (textual review, visual image, and numerical rating) to describe how the framework can be developed in practice. After that, we discuss and prove how different selections of ranking loss functions and multi-view representation merge functions affect the extendability of the framework.

3.1 Framework Overview

Let V_i be the i-th view in the framework. Specifically, we take V_1 for reviews, V_2 for images, and V_3 for ratings in this work. Let u refer to a specific user, and v to a specific item. In each view V_i, we have e^i_{uv} denoting the entity corresponding to user u and item v. For example, we have d_{uv} and r_{uv} referring to the review and rating given by user u towards item v, and in the image view, p_{uv} refers to the image that user u examined on item v. The entity representations $\mathbf{e}^k_{uv}$ (grey vectors in Figure 1 and 2) are linked to users and items in different views to obtain the user/item representations, which will be described in detail in the following subsections.

The JRL framework for recommendation consists of the following four components:

- Learn the representation $\mathbf{e}^k_{uv}$ of each entity e^k_{uv} in view V_k. When back-propagation is incorporated, this step will introduce an objective function $\mathcal{L}_k(\Theta_k)$.

Table 1: A summary of key notations in this work. Note that all vectors are denoted with bold lowercases.

V_k	The k-th view of the framework, specifically V_1, V_2, V_3 refer to review, image, and rating views in this work
u, v	An arbitrary user or item in the system
e^k_{uv}, $\mathbf{e}^k_{uv}$	An entity (and its representation) corresponding to user u and item v in view V_k. This is a general notation, and an entity can be a review, image, rating, or others
d_{uv}, $\mathbf{d}_{uv}$	The review user u written to item v in view V_1, and its representation learned by PV-DBOW
p_{uv}, $\mathbf{p}_{uv}$	The image that user u examined on item v in view V_2, and its representation learned by CNN
r_{uv}, $\mathbf{r}_u$, $\mathbf{r}_v$	The rating user u rated on item v in view V_3, and its corresponding user and item representations learned by multi-layer perceptron
m, n, N_k	Number of users and items in the system, as well as the total number of entities in view V_k
$\mathcal{R}$	The set of all observed user-item interactions
$\mathbf{u}_k$, $\mathbf{v}_k$	Representations of user u and item v in V_k
$\mathbf{u}$, $\mathbf{v}$	Integrated representations of user u and item v
$\mathcal{L}_k(\Theta_k)$	Regularization function for learning entity representations $\mathbf{e}^k_{uv}$ in view V_k, and Θ_k is the parameter set
λ_k	Regularization coefficient for $\mathcal{L}_k(\Theta_k)$
$\mathcal{W}_k$	The set of connection weight parameters from entity representations to user/item representations in view V_k
$f(\cdot)$	The merge function to get the integrated user/item representations from per-view user/item representations, i.e., $\mathbf{u} = f(\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3\cdots)$, $\mathbf{v} = f(\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3\cdots)$
$g(\cdot)$	The ranking objective function for pair-wise learning to rank, e.g., $g(u, v^+, v^-) = \sigma(\mathbf{u}^\top\mathbf{v}^+ - \mathbf{u}^\top\mathbf{v}^-)$

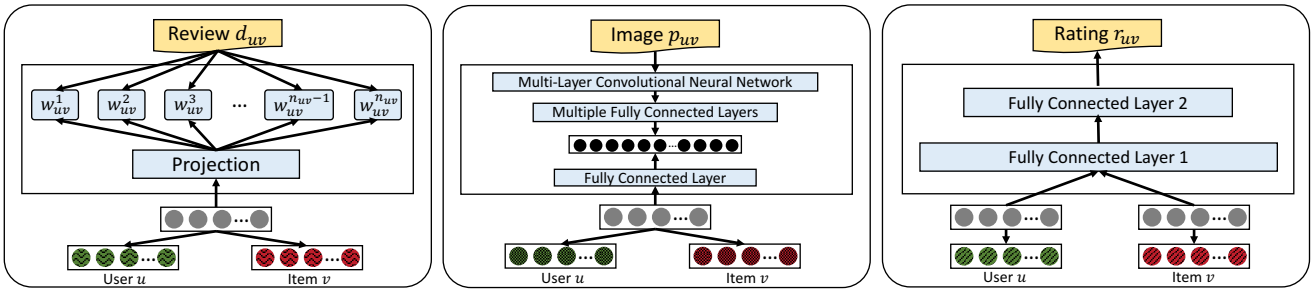
- In each view V_k, obtain user/item representations $\mathbf{u}_k, \mathbf{v}_k$ by linear combinations of entity representations $\mathbf{e}^k_{uv}$, and the learned connection weight parameters are denoted as $\mathcal{W}_k$, which is a set of connection weight parameters.
- Obtain the final user/item representations based on the merge function: $\mathbf{u} = f(\mathbf{u}_1\mathbf{u}_2\mathbf{u}_3\cdots), \mathbf{v} = f(\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3\cdots)$.
- Optimize the pair-wise ranking objective function $g(u, v^+, v^-)$ with objectives of each view $\mathcal{L}_k(\Theta_k)$ as regularizer.

Finally, the JRL framework optimizes the following abstract objective function for top-N recommendation:

$$\underset{\forall k:\{\mathcal{W}_k,\Theta_k\}}{\text{maximize}}\ \mathcal{L} = \sum_{(u,v^+)\in\mathcal{R}} g(u, v^+, v^-) + \sum_k \lambda_k \mathcal{L}_k(\Theta_k) \quad (1)$$

where (u, v^+) is a positive user-item pair indicating that user u purchased item v^+, while (u, v^-) is a negative pair where user u did not purchase v^-. In this framework, all observed user-item interaction pairs in $\mathcal{R}$ are treated as positive pairs, while negative pairs are randomly sampled from those items that a user did not purchase before.

We will show that different choices of merge function $f(\cdot)$ and ranking loss function $g(\cdot)$ will affect how information from different views are shared. Specifically, we will show that when $f(\cdot)$ and $g(\cdot)$ obtain certain separation properties, the JRL framework is extendable to new views without retraining of the existing views. Key notations adopted in this section are summarized in Table 1.



(a) Representation Learning of Reviews (b) Representation Learning of Images (c) Representation Learning of Ratings

Figure 2: The representation learning architectures for different views. We consider the textual review, visual image, and numerical rating views in this work, but the framework is extendable to more heterogeneous information sources as additional views. Each subfigure examples the learning of a single entity therein, and different entities in the same view are learned using the same architecture. In each view, entity representations are connected to users and items according to the corresponding user-item interactions in that view.

3.2 Modeling of Textual Reviews

We adopt the PV-DBOW model [24] to learn the review representations in V_1, which is shown in Figure 2(a). PV-DBOW assumes the independence between words in a document and uses the document to predict each observed word in it. Let d_{uv} be the review of user u on item v, with words $\{w_{uv}^1, w_{uv}^2 \cdots w_{uv}^{n_{uv}}\}$ in it. The representations of user u, item v, word w, review d_{uv} in this view V_1 are defined as $\mathbf{u}_1, \mathbf{v}_1, \mathbf{w}, \mathbf{d}_{uv} \in \mathbb{R}^K$, respectively, where K is the dimension (i.e., embedding size) of representations.

Specifically, each review d_{uv} is first projected into a semantic space and then trained to predict its words. With the bag-of-words assumption, the generative probability of word w in review d_{uv} is calculated by a softmax function over the whole vocabulary $\mathcal{V}$:

$$P(w|d_{uv}) = \frac{\exp(\mathbf{w}^\top \mathbf{d}_{uv})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{w}'^\top \mathbf{d}_{uv})} \tag{2}$$

To reduce the cost of gradient computation of Eq.(2) given a large vocabulary, we take the negative sampling strategy [29], which randomly samples several words according to a predefined noise distribution and uses these words to approximate the denominator of Eq.(2). With negative sampling, the global objective of PV-DBOW that sums over all possible word-review pairs is:

$$\begin{aligned}\mathcal{L}_1(\mathbf{w}, \mathbf{d}_{uv}) = &\sum_{w \in \mathcal{V}} \sum_{(u,v) \in \mathcal{R}} \mathrm{f}_{w,d_{uv}} \log \sigma(\mathbf{w}^\top \mathbf{d}_{uv}) \\ &+ \sum_{w \in \mathcal{V}} \sum_{(u,v) \in \mathcal{R}} \mathrm{f}_{w,d_{uv}} \left(t \cdot \mathbb{E}_{w_N \sim P_\mathcal{V}} \log \sigma(-\mathbf{w}_N^\top \mathbf{d}_{uv}) \right)\end{aligned} \tag{3}$$

where $\mathrm{f}_{w,d_{uv}}$ is the frequency of word-review pair, which is 0 when d_{uv} does not mention w. t is the number of negative samples, which is set as 5 in this work according to the guidance of [29]. $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function, and $\mathbb{E}_{w_N \sim P_\mathcal{V}}[\log \sigma(-\mathbf{w}_N^\top \mathbf{d}_{uv})]$ is the expected value of $\log \sigma(-\mathbf{w}_N^\top \mathbf{d}_{uv})$ given the noise distribution $P_\mathcal{V}$, for which we take the 3/4rd unigram distribution as in [29].

After the review representations $\mathbf{d}_{uv}$ are obtained, each $\mathbf{d}_{uv}$ is connected to its corresponding user u and item v, thus the user and item representations $\mathbf{u}_1, \mathbf{v}_1$ in V_1 are generated from their corresponding reviews. Weights of these connections consist a set $\mathcal{W}_1$, which is learned as parameter in the final objective function.

3.3 Modeling of Visual Images

We adopt frequently used DNN architectures to learn the representations in the image view V_2, as shown in Figure 2(b). Let p_{uv} denote the image that user u examined on item v. Same as [28], the image features $\vec{p}_{uv}$ (black intermediate vectors in Figure 2(b)) are calculated using the Caffe deep learning framework with 5 convolutional layers followed by 3 fully-connected layers, which was pre-trained on 1.2 million ImageNet (ILSVRC2010) images. Output of the second fully-connected layer are used as the intermediate image vectors $\vec{p}_{uv}$, whose dimension is 4096[1].

To learn our compressed image representations $\mathbf{p}_{uv}$, we adopt a fully connected layer to predict the image features $\vec{p}_{uv}$. Let A and $\mathbf{b}$ be the weight matrix and bias vector to be learned, this introduces the following objective function,

$$\mathcal{L}_2(A, \mathbf{b}, \mathbf{p}_{uv}) = \sum_{(u,v) \in \mathcal{R}} \left(\phi(A \cdot \mathbf{p}_{uv} + \mathbf{b}) - \vec{p}_{uv} \right)^2 \tag{4}$$

where $\phi(\cdot)$ is the activation function that can be sigmoid, hyperbolic tangent (tanh), exponential linear units (ELU), or others. It is known that the sigmoid function restricts each neuron to be in (0,1), which may limit the performance and may also suffer from saturation, where neurons stop learning when their output is near either 0 or 1. The tanh function only partly alleviates the problem because $\tanh(x/2) = 2\sigma(x) - 1$. As a result, we take the state-of-the-art ELU function in this work to avoid this problem [11].

Similar to reviews, after the image representations $\mathbf{p}_{uv}$ are obtained, each $\mathbf{p}_{uv}$ is also connected to its corresponding user u and item v, so that the user and item representations $\mathbf{u}_2, \mathbf{v}_2$ in view V_2 are generated from their corresponding images. Weights of these connections consist a parameter set $\mathcal{W}_2$ to be learned.

3.4 Modeling of Numerical Ratings

In this subsection, we present the modeling of the rating view in our framework, as shown in Figure 2(c). Let r_{uv} be the rating that user u scores on item v. In contrast to the modeling of reviews and images, we do not apply a single representation to each rating

[1] In experiments, the Amazon review dataset has pre-trained the image features and each image is already represented as a 4096-dimensional vector $\vec{p}_{uv}$, so we used these vectors directly to learn our image representations $\mathbf{p}_{uv}$.

number, but use a pair of representations $\mathbf{r}_u, \mathbf{r}_v$ corresponding to the rating user and the rated item to predict the ratings, with the following two-layer fully connected neural network to model the underlying non-linear correlations:

$$\hat{r}_{uv} = \phi\left(U_2 \cdot \phi\Big(U_1(\mathbf{r}_u \odot \mathbf{r}_v) + \mathbf{c}_1\Big) + \mathbf{c}_2\right) \tag{5}$$

where $\odot$ is element-wise multiplication, $\phi(\cdot)$ is also the ELU activation function, and U_1, U_2, $\mathbf{c}_1$, $\mathbf{c}_2$ are the weight and bias parameters to be learned. This gives us the following objective function,

$$\mathcal{L}_3(U_1, U_2, \mathbf{c}_1, \mathbf{c}_2, \mathbf{r}_u, \mathbf{r}_v) = \sum_{(u,v)\in\mathcal{R}} (\hat{r}_{uv} - r_{uv})^2 \tag{6}$$

Because the entity representations of ratings have already been mapped to each user and item, we take the entity representations directly as the user/item representations in view V_3, i.e., $\mathbf{u}_3 = \mathbf{r}_u$, $\mathbf{v}_3 = \mathbf{r}_v$, and connection weight set $\mathcal{W}_3$ is predefined as all 1's.

3.5 Integrated Recommendation Strategy

In this subsection, we integrate the above three views in a unified pair-wise learning to rank framework. We already have the user and item representations from each of the three views above, which are $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ and $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, respectively. With merge function $f(\cdot)$, we obtain the integrated user and item representations $\mathbf{u}$ and $\mathbf{v}$,

$$\mathbf{u} = f(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3),\ \mathbf{v} = f(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \tag{7}$$

Common selections of merge functions can be used in this step, such as concatenation (i.e., $\mathbf{u} = [\mathbf{u}_1^\intercal \mathbf{u}_2^\intercal \mathbf{u}_3^\intercal]^\intercal$) or average (i.e., $\mathbf{u} = \frac{1}{3}(\mathbf{u}_1 + \mathbf{u}_2 + \mathbf{u}_3)$). In this work, we adopt the simple concatenation function to guarantee the extendability of the framework, which will be analyzed in the next section.

To conduct pair-wise learning to rank, we consider all the observed user-item purchases $(u, v) \in \mathcal{R}$ as positive pairs, which are identically denoted as (u, v^+). For each positive pair (u, v^+), we randomly select a negative item v^- that the user did not purchase before to construct a triplet (u, v^+, v^-) for training, where $g(u, v^+, v^-) = g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-)$ denotes the ranking loss function based on the integrated user and item representations. Generally, $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-)$ can be any function as long as its maximization or minimization leads to higher rankings of v^+ against v^- for user u, but its functional form directly affects whether the framework is easily extendable to new views. For here, we primarily adopt the sigmoid function $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-) = \sigma(\mathbf{u}^\intercal \mathbf{v}^+ - \mathbf{u}^\intercal \mathbf{v}^-)$ for model learning. In the next section, we will discuss how different selections of $g(\cdot)$ result in different extendable properties of this framework.

By integrating the objective functions in Eq.(3)(4)(6) as well as the objective function for pair-wise learning to rank into Eq.(1), we have the objective function for 3-view JRL as,

$$\begin{aligned}
&\underset{\mathcal{W},\Theta}{\text{maximize}}\ \mathcal{L} = \sum_{(u,v)\in\mathcal{R}} g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-) + \lambda_1\mathcal{L}_1 - \lambda_2\mathcal{L}_2 - \lambda_3\mathcal{L}_3 \\
&= \sum_{(u,v)\in\mathcal{R}} \Big\{\sigma(\mathbf{u}^\intercal \mathbf{v}^+ - \mathbf{u}^\intercal \mathbf{v}^-) - \lambda_2\Big(\phi(A \cdot \mathbf{p}_{uv} + \mathbf{b}) - \vec{p}_{uv}\Big)^2 \\
&+ \lambda_1 \sum_{w\in\mathcal{V}} \mathrm{f}_{w,d_{uv}}\Big(\log\sigma(\mathbf{w}^\intercal \mathbf{d}_{uv}) + t \cdot \mathbb{E}_{w_N\sim P_\mathcal{V}} \log\sigma\Big(-\mathbf{w}_N^\intercal \mathbf{d}_{uv}\Big)\Big) \\
&- \lambda_3\Big(\phi\left(U_2 \cdot \phi\Big(U_1(\mathbf{r}_u \odot \mathbf{r}_v) + \mathbf{c}_1\Big) + \mathbf{c}_2\right) - r_{uv}\Big)^2\Big\}
\end{aligned} \tag{8}$$

where the optimization parameters $\mathcal{W} = \{\mathcal{W}_1, \mathcal{W}_2, \mathcal{W}_3\}$ and $\Theta = \{\Theta_1, \Theta_2, \Theta_3\} = \{\{\mathbf{w}, \mathbf{d}_{uv}\}, \{A, \mathbf{b}, \mathbf{p}_{uv}\}, \{U_1, U_2, \mathbf{c}_1, \mathbf{c}_2, \mathbf{r}_u, \mathbf{r}_v\}\}$. Besides, $\lambda_1, \lambda_2, \lambda_3 > 0$ are regularization coefficients. Note that the image and rating regularizer have negative coefficients because their objectives need to be minimized instead of being maximized. Besides, the parameters of each view V_k are restricted to its own part of objective function $\mathcal{L}_k$, and the only interaction of different views lies in the ranking objective $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-)$, where $\mathbf{u} = f(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ and $\mathbf{v} = f(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ integrate representations learned from different views. This affects the gradient on different parameters and further affects the extendability of the framework, which will be analyzed in the next section.

Eq.(8) can be easily optimized based on Stochastic Gradient Descent (SGD) in well-developed deep learning infrastructures. Once we obtain the integrated user/item representations $\mathbf{u}$ and $\mathbf{v}$, the personalized recommendation list for each user is constructed by ranking all the candidate items in descending order of $s = \mathbf{u}^\intercal \mathbf{v}$.

4 FRAMEWORK EXTENDABILITY

Intuitively, our JRL framework is already extendable to new views in practice, in the sense that when a new view is added to the framework, there is no need to redesign the architecture of existing views – we only need to design a model to learn the user/item representations from the new information source, add it to the integrated user/item representations with the merge function $f(\cdot)$, and finally retrain the whole framework for personalized recommendation.

But in this section, we discuss the extendability of the framework in a more rigorous sense – we expect that there should not only be no need to redesign the architecture of existing views, but also no need to retrain the parameters of existing views, so that the already trained model parameters can still be used even a new view is added into the framework – which is a very favorable property for real-world systems.

To this end, we analyze the learning process of the framework by examining the updating gradients of different parameters, based on which we take a closer look at how information from different views is shared during model learning. Further more, by selecting a proper merge function $f(\cdot)$ and ranking objective function $g(\cdot)$, we propose a rigorously extendable version of the JRL framework (denoted as eJRL) for recommendation.

4.1 Information Propagation among Views

Careful readers may have realized that our JRL framework does not involve common parameters across views – each view has its own optimization parameter set Θ_k for objective function $\mathcal{L}_k$, as well as its own connection weight parameter set $\mathcal{W}_k$ to map entity representations to user/item representations within the view. This raises the question of how information is transferred amongst different views, so as to take advantage of the power of multi-view machine learning. We examine this problem by looking into the model learning process based on (stochastic) gradient descent methods.

For the optimization of Θ_k of the k-th view, we have,

$$\frac{\partial\mathcal{L}}{\partial\Theta_k} = \lambda_k \frac{\partial\mathcal{L}_k}{\Theta_k} \tag{9}$$

which only involves variables and parameters of the k-th view itself. While for the optimization of $\mathcal{W}_k$, we have,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathcal{W}_k} &= \sum_{(u,v)\in\mathcal{R}} \left(\frac{\partial g}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathcal{W}_k} + \frac{\partial g}{\partial \mathbf{v}^+} \frac{\partial \mathbf{v}^+}{\partial \mathcal{W}_k} + \frac{\partial g}{\partial \mathbf{v}^-} \frac{\partial \mathbf{v}^-}{\partial \mathcal{W}_k} \right) \\ &= \sum_{(u,v)\in\mathcal{R}} \left(\frac{\partial g}{\partial \mathbf{u}} \frac{\partial f}{\partial \mathbf{u}_k} \frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k} + \frac{\partial g}{\partial \mathbf{v}^+} \frac{\partial f}{\partial \mathbf{v}_k^+} \frac{\partial \mathbf{v}_k^+}{\partial \mathcal{W}_k} + \frac{\partial g}{\partial \mathbf{v}^-} \frac{\partial f}{\partial \mathbf{v}_k^-} \frac{\partial \mathbf{v}_k^-}{\partial \mathcal{W}_k} \right) \\ &\triangleq \sum_{(u,v)\in\mathcal{R}} \left(h(\mathbf{u}) \frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k} + h(\mathbf{v}^+) \frac{\partial \mathbf{v}_k^+}{\partial \mathcal{W}_k} + h(\mathbf{v}^-) \frac{\partial \mathbf{v}_k^-}{\partial \mathcal{W}_k} \right)\end{aligned} \tag{10}$$

where the second equality holds for $\mathbf{u} = f(\mathbf{u}_1, \mathbf{u}_2, \cdots)$ because that,

$$\frac{\partial g}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathcal{W}_k} = \frac{\partial g}{\partial \mathbf{u}} \left(\frac{\partial f}{\partial \mathbf{u}_k} \frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k} + \sum_{j\neq k} \frac{\partial f}{\partial \mathbf{u}_j} \frac{\partial \mathbf{u}_j}{\partial \mathcal{W}_k} \right) = \frac{\partial g}{\partial \mathbf{u}} \frac{\partial f}{\partial \mathbf{u}_k} \frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k} \tag{11}$$

and that similar results can be obtained for $\mathbf{v}^+ = f(\mathbf{v}_1^+, \mathbf{v}_2^+, \cdots)$ and $\mathbf{v}^- = f(\mathbf{v}_1^-, \mathbf{v}_2^-, \cdots)$. The last equality is for notational purpose by defining the following discriminant,

$$h(\mathbf{x}) = \frac{\partial g}{\partial \mathbf{x}} \frac{\partial f}{\partial \mathbf{x}_k}, \ \forall \mathbf{x} \in \{\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-\} \tag{12}$$

where $\mathbf{x}$ represents any integrated user/item representation, and $\mathbf{x}_k$ is the corresponding representation from view V_k.

We see that for each additive component in Eq.(10), the last multiplier (i.e., $\frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k}$, $\frac{\partial \mathbf{v}_k^+}{\partial \mathcal{W}_k}$, and $\frac{\partial \mathbf{v}_k^-}{\partial \mathcal{W}_k}$) is only related to view V_k itself, while discriminant part $h(\cdot)$ may involve variables and parameters from other views, which makes it possible for different views to share information during the model learning process. For example, when we adopt sigmoid function $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-) = \sigma(\mathbf{u}^\top \mathbf{v}^+ - \mathbf{u}^\top \mathbf{v}^-)$ as in Eq.(8), and adopt the concatenation function $f(\mathbf{u}_1, \mathbf{u}_2, \cdots) = [\mathbf{u}_1^\top \mathbf{u}_2^\top \cdots]^\top$ as examined before, then we have,

$$h(\mathbf{u}) = \frac{\partial g}{\partial \mathbf{u}} \frac{\partial f}{\partial \mathbf{u}_k} = \sigma'(\mathbf{u}^\top \mathbf{v}^+ - \mathbf{u}^\top \mathbf{v}^-)(\mathbf{v}_k^+ - \mathbf{v}_k^-)^\top \tag{13}$$

which is related not only to view V_k, but also variables from other views because $\mathbf{u} = [\mathbf{u}_1^\top \mathbf{u}_2^\top \cdots]^\top$, $\mathbf{v}^+ = [\mathbf{v}_1^{+\top} \mathbf{v}_2^{+\top} \cdots]^\top$, and $\mathbf{v}^- = [\mathbf{v}_1^{-\top} \mathbf{v}_2^{-\top} \cdots]^\top$. This is also true for $h(\mathbf{v}^+) = \sigma'(\mathbf{u}^\top \mathbf{v}^+ - \mathbf{u}^\top \mathbf{v}^-)\mathbf{u}_k^\top$ and $h(\mathbf{v}^-) = \sigma'(\mathbf{u}^\top \mathbf{v}^+ - \mathbf{u}^\top \mathbf{v}^-)(-\mathbf{u}_k^\top)$.

As a result, the learning of user/item representations in view V_k based on Eq.(10) would rely on information (i.e., user/item representations) from other views, which eventually propagates information among different views.

4.2 Extendable JRL (eJRL)

Although information propagation among views helps to boost the performance of joint representation learning, it also prevents the framework from being extendable, that is, whenever a new view is added, we need to retrain the whole model because information from the new view will affect the parameters of the existing views, which is not favorable in real-world systems that usually need to train models based on massive data. However, we show that by selecting certain combinations of the merge function $f(\cdot)$ and ranking objective function $g(\cdot)$, we can obtain an extendable version of the JRL framework.

The key is to examine the discriminant $h(\mathbf{x}) = \frac{\partial g}{\partial \mathbf{x}} \frac{\partial f}{\partial \mathbf{x}_k}$ in Eq.(12). If for any $\mathbf{x}$, $h(\mathbf{x})$ contains no variable from other views beyond V_k, then the gradient $\frac{\partial \mathcal{L}}{\partial \mathcal{W}_k}$ in Eq.(10) will also be independent from other views. For example, when $f(\mathbf{u}_1, \mathbf{u}_2, \cdots) = [\mathbf{u}_1^\top, \mathbf{u}_2^\top, \cdots]^\top$ and $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-) = \sum_k \sigma(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-)$, we have,

$$\begin{aligned} &h(\mathbf{u}) = \sigma'(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-)(\mathbf{v}_k^+ - \mathbf{v}_k^-)^\top \\ &h(\mathbf{v}^+) = \sigma'(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-)\mathbf{u}_k^\top, \ h(\mathbf{v}^-) = \sigma'(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-)(-\mathbf{u}_k^\top) \end{aligned} \tag{14}$$

By substituting Eq.(14) into Eq.(10), we obtain the gradient on parameter set $\mathcal{W}_k$ as,

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}_k} = \sum_{(u,v)\in\mathcal{R}} \sigma'(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-) \left((\mathbf{v}_k^+ - \mathbf{v}_k^-)^\top \frac{\partial \mathbf{u}_k}{\partial \mathcal{W}_k} + \mathbf{u}_k^\top \frac{\partial \mathbf{v}_k^+}{\partial \mathcal{W}_k} - \mathbf{u}_k^\top \frac{\partial \mathbf{v}_k^-}{\partial \mathcal{W}_k} \right) \tag{15}$$

which only contains variables from view V_k itself. Combining Eq.(9) and Eq.(15), the parameters $\{\Theta_k, \mathcal{W}_k\}$ of view V_k will converge to the same solution even if we retrain the whole model with a new view added into the framework, as long as we adopt gradient descent algorithms (e.g., SGD, BFGS, Adam, etc) for optimization. In this case, we can actually fix the parameters of existing views and only gradient on the parameters of the newly added view, which makes the framework easily extendable in practice.

We denote this model under $g(\mathbf{u}, \mathbf{v}^+, \mathbf{v}^-) = \sum_k \sigma(\mathbf{u}_k^\top \mathbf{v}_k^+ - \mathbf{u}_k^\top \mathbf{v}_k^-)$ and $f(\mathbf{u}_1, \mathbf{u}_2, \cdots) = [\mathbf{u}_1^\top, \mathbf{u}_2^\top, \cdots]^\top$ as the Extendable Joint Representation Learning (eJRL) framework. Although it prevents information propagation among different views, it still significantly outperforms baseline methods because of the integration of multiple information sources for recommendation. We will report the performance of both JRL and eJRL in the experiments.

5 EXPERIMENTS

In this section, we provide and analyze the experimental results to study the performance of our (e)JRL framework. We first provide the dataset descriptions and experimental setup, and then present the evaluations and interpretations of our observations.

5.1 Dataset Description

We adopt the Amazon review dataset[2] for experiments. It covers user interactions (review, rating, helpfulness votes, etc) on items as well as the item metadata (descriptions, price, brand, image features, etc) on 24 product categories spanning May 1996 - July 2014, and each product category consists a sub-dataset. We adopt five product categories of different sizes and sparsity, and take the standard five-core dataset for experiment. Some statistics of the datasets are shown in Table 2.

The number of interactions in Table 2 refers to the total number of reviews or ratings in each dataset. Each item is accompanied with an image, which has already been processed into a 4096-dimensional vector in the dataset, and we take these vectors to fit the user/item representations, as explained in Section 3.3.

For each dataset, we randomly select 70% of the interactions from each user to construct the training set, and adopt the remaining 30% for testing. Because we take the 5-core dataset where each user

[2] http://jmcauley.ucsd.edu/data/amazon/

Table 2: Basic statistics of the experimental datasets.

Dataset	#users	#items	#interactions	sparsity
Movies	123,960	50,052	1,697,533	0.0274%
CDs	75,258	64,421	1,097,592	0.0226%
Clothing	39,387	23,033	278,677	0.0307%
Cell Phones	27,879	10,429	194,439	0.0669%
Beauty	22,363	12,101	198,502	0.0734%

has at least 5 interactions, thus we have at least 3 interactions per user for training, and at least 2 interactions per user for testing.

5.2 Experimental Setup

We compare and analyze our JRL and eJRL framework, as well as our framework using each single view (review, image, or rating), with the following baselines[3].

• **BPR**: The Bayesian Personalized Ranking approach for recommendation, which is one of the state-of-the art ranking-based method for top-N recommendation with numerical ratings. Specifically, we use BPR-MF for model learning [31].

• **BPR-HFT**: The Hidden Factors and Topics model is one of the state-of-the-art methods for rating prediction with textual reviews [27], but it is not specifically designed for top-N recommendation. We apply Bayesian personalized ranking on top of HFT for better top-N recommendation while using textual reviews.

• **VBPR**: The Visual Bayesian Personalized Ranking method for recommendation, which is the state-of-the-art method for recommendation based on visual images of the products [18].

• **DeepCoNN**: The Deep Cooperative Neural Networks model for recommendation, which models users and items jointly using review text for rating prediction [45].

• **CKE**: The Collaborative Knowledge-base Embedding model for recommendation. We adopt the textual product description and product image knowledge for model implementation [40].

In general, we have considered both shallow (BPR, BPR-HFT, VBPR) and deep (DeepCoNN, CKE) models. Besides, they cover different information sources, including ratings (BPR), reviews (BPR-HFT, DeepCoNN), and images (VBPR, CKE), respectively.

To simulate a practical application, we train the different views independently and then merge the views for the eJRL framework; while for JRL, different views are trained as a whole. We adopt stochastic gradient descent with batch size 64 and train each model for 20 epochs. The learning rate is 0.5 multiplied with the number of trained instances divided by the total number of training instances, which dynamically shrinks during the learning procedure. We set the number of negative samples $t = 5$ in Eq.(3), and clip the global norm of gradients from each batch with 5. Unless otherwise specified, we primarily set the embedding size (length of vectors $\mathbf{d}_{uv}, \mathbf{p}_{uv}, \mathbf{r}_u, \mathbf{r}_v$, and $\mathbf{u}_k, \mathbf{v}_k$) as 300, and set the regularization coefficients as $\lambda_1 = \lambda_3 = 1, \lambda_2 = 0.0001$ ($\lambda_2 = 0.001$ for clothing dataset). Performance on different parameter settings will be analyzed in Section 5.6 and 5.7.

We conduct five-fold cross-validation on training set to tune the best hyper-parameters of each baseline. Specifically, the number of topics is 10 for BPR-HFT, and the dimension of latent factor (or embedding size) is 100 for baselines. The regularization coefficient $\lambda = 10$ works the best for BPR and VBPR. Optimization for baselines terminate until convergence or 150 learning epochs.

[3]Source code of our models are available at https://github.com/evison/JRL

5.3 Evaluation Measures

For evaluation, we adopt the following four representative top-N recommendation measures:

• **Precision**: Percentage of correctly recommended items in a user's recommendation list, averaged across all testing users.

• **Recall**: Percentage of purchased items that are really recommended in the list, and it is also averaged across all testing users.

• **NDCG**: The most frequently used list evaluation measure that takes into account the position of correctly recommended items. NDCG is averaged across all the testing users.

• **HT**: Hit Ratio, which is the percentage of users that have at least one correctly recommended item in their list.

We provide top-N recommendation list for each user in the testing set, where N=10 is taken to report the numbers and compare different algorithms.

5.4 Single-view Performance

We first look into the performance of our framework when using each single view, i.e., when only one of the regularization coefficients λ_i in Eq.(8) is non-zero. Results are shown in Table 3.

We see that both of the deep baselines (DeepCoNN and CKE) are better than any of the shallow baselines, which is in accordance with the observations in [40, 45]. Besides, we see that CKE performs better than DeepCoNN on most datasets. The reason can be that CKE incorporates more multimodal information (both product description and image) for product embedding, while DeepCoNN only takes advantage of textual reviews. Another reason is that DeepCoNN adopts point-wise learning for recommendation, while CKE is based on pair-wise learning to rank, and the latter has been frequently observed to be better than point-wise methods on top-N recommendation tasks [12].

We first compare our model with the shallow baselines to analyze the advantage of deep representation learning methods. Specifically, we compare our review-based model with BPR-HFT – the baseline that also models textual reviews for recommendation. The improvement mainly comes from two aspects: 1) embedding-based representation learning gives higher degree of freedom to find the word/document representations, instead of pre-assuming a fixed number of topics as in topic modeling (used by BPR-HFT); and 2), by word embedding, our model can better capture the semantic similarity between words and phrases, which helps to aggregate user preferences from multiple reviews that use different but semantically similar expressions.

Using images alone in our framework also outperforms VBPR – the baseline that models images for recommendation. Considering the difference of image modeling in our framework from that of VBPR, this implies that by connecting both users and items to the image representations directly (see Figure 2(b)), our model profiles the users and items in the same semantic space spanned by images, which can better capture the user-item similarity than the latent affine space used in VBPR.

However, when using the rating view alone, our framework did not outperform the best baseline – it was only comparable to BPR

Table 3: Summary of performance for baselines and our framework with single- and multi-view settings (note: all numbers in the table are percentage numbers with '%' omitted). The first block shows the shallow baseline performance, where the starred numbers are the best shallow baseline performance among the four; the second block shows deep baseline performance; the third block shows the results of our framework with each single view (Review, Image, Rating) and multi-views (JRL, eJRL). The last block shows the percentage increment (or decrement for negative numbers) of our results against the best baseline (i.e., CKE). All increments/decrements are significant at p=0.001. Underlined numbers show the best single view among all three views, while bolded numbers are the best performance of each column.

Dataset	Movies				CDs				Clothing				Cell Phones				Beauty			
Measures(%)	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec
BPR	1.267	1.988	4.421	0.528	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185	1.998	3.258	5.273	0.595	2.753	4.241	8.241	*1.143
BPR-HFT	*2.092	*3.255	*6.378	*0.776	*2.661	*3.570	*9.926	*1.268	*1.067	*1.819	*2.872	*0.297	*3.151	*5.307	*8.125	*0.860	*2.934	*4.459	*8.268	1.132
VBPR	0.849	1.534	2.976	0.324	0.631	0.845	2.930	0.328	0.560	0.968	1.557	0.166	1.797	3.489	5.002	0.507	1.901	2.786	5.961	0.902
DeepCoNN	3.800	4.671	10.522	0.886	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
CKE	4.091	5.466	11.053	1.319	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
Review	4.222	6.145	12.958	1.465	5.286	7.454	16.592	2.079	1.270	2.211	3.527	0.336	4.184	7.275	10.632	1.062	4.216	6.766	12.422	1.467
Image	2.648	4.035	9.489	1.048	3.191	4.564	11.547	1.379	1.393	2.481	3.773	0.354	3.777	6.439	9.444	0.932	3.310	5.288	10.280	1.211
Rating	0.432	0.700	2.242	0.234	0.528	0.747	2.394	0.248	0.377	0.732	1.219	0.112	1.506	2.706	3.845	0.369	0.876	1.442	3.322	0.313
eJRL	**4.405**	6.289	**13.292**	**1.521**	5.023	6.973	16.081	2.002	1.523	2.679	4.182	0.396	4.185	7.130	10.531	1.054	3.896	6.010	11.090	1.355
JRL	4.334	**6.334**	13.245	1.492	**5.378**	**7.545**	**16.774**	**2.085**	**1.735**	**2.989**	**4.634**	**0.442**	**4.364**	**7.510**	**10.940**	**1.096**	**4.396**	**6.949**	**12.776**	**1.546**
Review-Impr	3.20	12.43	17.24	11.09	14.43	14.99	14.11	16.86	-15.46	-11.90	-17.51	-13.36	4.72	3.86	-1.64	-0.73	13.42	13.94	12.49	6.98
Image-Impr	-35.28	-26.19	-14.15	-20.58	-30.92	-29.59	-20.59	-22.50	-7.24	-1.12	-11.75	-8.84	-5.47	-8.08	-12.63	-12.89	-10.96	-10.95	-6.91	-11.67
eJRL-Impr	7.67	15.07	20.26	15.33	8.72	7.57	10.59	12.54	1.41	6.75	-2.19	1.92	4.74	1.79	-2.57	-1.52	4.81	1.21	0.42	-1.14
JRL-Impr	5.92	15.89	19.84	13.08	16.40	16.39	15.36	17.21	15.52	19.12	8.38	13.87	9.23	7.21	1.21	2.41	18.27	17.03	15.69	12.76

on some datasets. Because of the sparsity of rating interactions and the huge parameter complexity in the training stage of our model, this observation is not surprising, and it also implies that the power of (deep) representation learning structures can be better leveraged with the availability of large-scale unstructured data of rich (textual or visual) semantics, such as text or image.

We also see that Review performs better than Image on most datasets except clothing, which is expected because users' preferences to clothes are largely affected by their visual fashions, which is also observed in VBPR [18]. But for other domains such as movie or CD, it is relatively difficult to make judgements only based on the appearance of a movie poster or a CD cover, while the textual comments from users may reveal more details about these products.

Similar observations can be found when comparing our single-view model with the best baseline (CKE), shown in the last block of Table 3. We see that Review achieves better performance than CKE on all datasets except Clothing and (partly) Cell Phones, because the review information not only contains users' description of products, but also their personalized preferences and sentiments on these products, while the product text description information used by CKE is non-personalized. On Clothing dataset, however, image is the most informative source for consumer decisions, as a result, CKE is better for its adoption of both text and image information. However, by incorporating image, review, and rating information sources together, our unified JRL or eJRL models are better than the best CKE baseline, which will be analyzed in the next subsection.

5.5 Multi-view Performance

We further look into our framework when integrating all three views. Generally, we see that on most datasets, eJRL is better than CKE, and JRL is even better than eJRL for nearly all the cases.

Intuitively, our ranking-based unification of different views, plus the adoption of concatenation function $f(\mathbf{u}_1, \mathbf{u}_2, \cdots) = [\mathbf{u}_1^\intercal, \mathbf{u}_2^\intercal, \cdots]^\intercal$ as the merge function, together help to gain better performance when integrating different views. Mathematically, this is equivalent to adding up the ranking scores $s_k = \mathbf{u}_k^\intercal \mathbf{v}_k$ from each view to rerank the top-N recommendation lists produced by different views. As a result, an item tends to gain a higher score in the final recommendation list as long as it achieves a high score from one view, i.e., as long as the user preference on this item is appropriately profiled by one information source.

Furthermore, by allowing information propagation among views during model learning, each view borrows representations from other views to fit the ranking objective $g(\cdot)$, which helps the JRL framework to achieve even better performance than eJRL. However, the performance of eJRL is generally comparable to JRL on most cases, and they are both significantly better than the best CKE baseline. As a result, eJRL has advantages in practice given its extendability property.

We also find that selecting proper regularization coefficients in Eq.(8) is important to gain better integrated performance on JRL and eJRL than the single-view versions – especially the image view regularizer λ_2. We analyze the parameter sensitivity and explain the reasons in the following subsections.

5.6 Impact of Regularization Coefficients

During the experiments, we find that the performance of the (e)JRL framework is relatively stable in terms of review and rating regularization coefficients λ_1 and λ_3, but the image regularization coefficient λ_2 largely affects the results. As a result, we fix $\lambda_1 = \lambda_3 = 1$ as the default value, and vary the values of λ_2 to study the effect.

Figure 3 shows the performance of Review, Image, eJRL, and JRL on the five datasets in terms of NDCG. Observations on other evaluation measures were similar. Note that only the performance of Image, eJRL and JRL change with different values of λ_2, because other model variations do not involve this parameter.

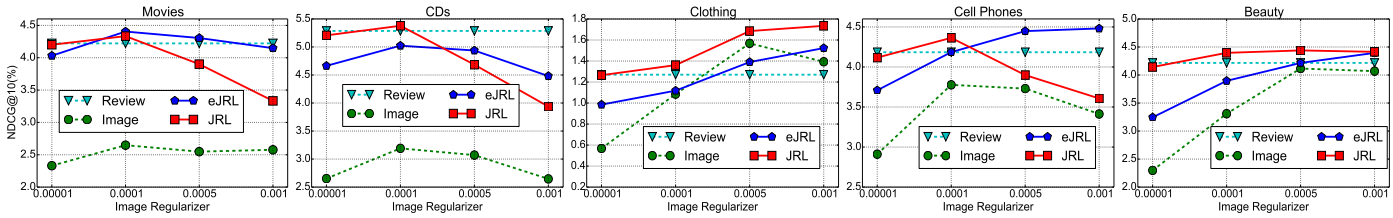


Figure 3: Relationship of NDCG vs the Image Regularization Coefficient λ_2 under the default embedding size 300 and other regularizers $\lambda_1 = \lambda_3 = 1$. Note that only the performance of Image, eJRL, and JRL approaches change with different selections with λ_2, because other model variations do not involve this parameter. We also plot Review performance for reference.

We see that on most datasets (Movies, CDs, Cell Phones, Beauty), the performance of our model increases with the increase of λ_2 at the beginning, and then drops when λ_2 further increases, which indicates that a proper λ_2 value is required for the model to gain the best performance. When λ_2 is too small, the image regularization term vanishes, which prevents the model from fitting accurate user preferences based on visual images, while if λ_2 is too large, the image regularizer dominates the gradient direction during SGD learning. Specifically, we see that the best selection of λ_2 is 0.0001 for JRL on all these four datasets, which is actually not a coincidence – by referring to Eq.(8), we see that the image regularizer component is an ℓ_2-norm of a 4096-dimensional vector, while the remaining components are scalars. Though the scale of each element of the image vector is comparable to the other components of the objective function, they amount to approximately 4000 times when added into an ℓ_2-norm. As a result, a coefficient on the magnitude of 0.0001 is needed to rescale the image regularizer to be comparable with other components in the range of $0 \sim 1$. Otherwise, the model would be dominated by the image view and will only gradient descend towards the image features in SGD, which is verified by our observations when tracing the learning procedure. This observation indicates that it is important to make sure that different views are comparable on scale when adding new information sources into the JRL framework.

For eJRL, on the other hand, different views do not interact during model learning, and this is why we see that eJRL is better than JRL when λ_2 is large. However, for most datasets in Figure 3, the globally best performance is still achieved when using JRL with the appropriate image regularization coefficient.

We also see that on the Clothing dataset, the best image regularization coefficient is $\lambda_2 = 0.001$. Together with the observation in Section 5.4 that clothing is the only dataset where Image achieved better performance than Review on single-view modeling, this implies that it would be beneficial to strengthen the best view when integrating different views so as to achieve further promoted performance in multi-view modeling.

5.7 Impact of Embedding Size

We study the effect of different embedding sizes in this subsection. To do so, we fix λ_2 as default values (0.001 for clothing and 0.0001 for other datasets), and then tune the embedding size from 10 to 500. We plot the results on Movies and CDs in Figure 4, and observations on other datasets were similar.

We see that the performances of both JRL and eJRL gradually increase with the increase of embedding size, and then tend to be

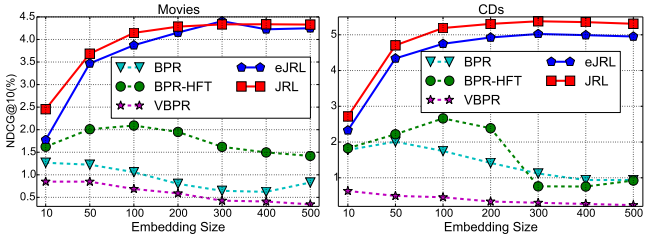


Figure 4: Relationship of NDCG vs embedding size (Movies and CDs) under default regularizer coefficient $\lambda_2 = 0.0001$.

stable when the embedding size is sufficiently large, and this is why we adopt 300 as the default embedding size in the previous experiments. This observation is similar to previous work on matrix factorization algorithms, however, MF algorithms usually achieve stable performance with several tens or at most a hundred of latent factors, while the performance of our JRL and eJRL frameworks keep increasing until a few hundred factors, which implies that with the availability of large-scale user-generated data, our frameworks are able to learn more complex user-item interactions with more parameters than matrix factorization.

For the baseline algorithms, we see that the best performance is usually achieved with smaller embedding sizes, for example, the BPR-HFT method achieves its best performance when using 100 latent factors, and BPR and VBPR perform best with even smaller embedding size. Because of their limited ability to model complex user-item interactions in multimodal data, these methods could not benefit from an increased parameter complexity, instead, they may even result in over-fitting when too many latent factors are used for model learning. This observation, on the other hand, further verifies the advantage of our joint representation learning approach for recommendation. The only trade-off here is recommendation quality and the speed of model training. In our experiments, both the JRL and eJRL models can be trained within 10 hours on a single GPU for most datasets and parameter settings, which is sufficient to support daily-level model updating in practice.

5.8 Discussions

Another advantage of our deep representation learning approach to recommendation is its ability of fast online prediction. This is because we adopt a simple vector multiplication $s = \mathbf{u}^\top \mathbf{v}$ to calculate ranking scores online, as a result, once the user and item embeddings are calculated and stored in the offline, the online prediction and recommendation procedure only involve vector multiplications, which is similar to conventional latent factor models such as matrix factorization. This is more efficient than other deep learning

approaches to recommendation that train a complex prediction network for online ranking score calculation.

6 CONCLUSIONS AND FUTURE WORK

In this work, we proposed a Joint Representation Learning (JRL) framework based on multi-view machine learning, which is capable of incorporating heterogeneous information sources for top-N recommendation by learning user/item representations in a unified space. We analyzed how information is propagated among different views in a gradient-based model learning paradigm, and further proposed a rigorously extendable version of the JRL framework (eJRL), which makes it possible to integrate new views (i.e., information sources) without re-training of existing views. Experiments on various datasets verified the effectiveness of both JRL and eJRL.

In contrast to previous work that mostly focuses on rating prediction tasks, our work reveals the significant potential for improvement on top-N recommendation tasks brought about by the power of representation learning architectures, and there is even more room for further improvements. In the future, we will consider alternative representation learning architectures to model reviews, images, and ratings. Specifically, we would like to capture the word sequential information and their local semantics to better model the textual reviews for recommendation, design structures to further promote the rating view in top-N recommendation, and finally incorporate more information sources such as sound tracks or even videos towards a unified, multi-view informed practical recommendation system.

ACKNOWLEDGEMENT

We thank the reviewers for the careful reviews and constructive suggestions. This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1160894. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Silvana Aciar, Debbie Zhang, Simeon Simoff, and John Debenham. 2007. Informed recommender: Basing recommendations on consumer product reviews. *IEEE Intelligent systems* 22, 3 (2007), 39–47.
[2] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *RecSys*. 147–154.
[3] Yang Bao, Hui Fang, and Jie Zhang. 2014. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation.. In *AAAI*.
[4] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect Based Recommendations: Recommending Items with the Most Valuable Aspects Based on User Reviews. In *KDD*. 717–725.
[5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *PAMI* (2013).
[6] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.
[7] Robin Burke. 2007. Hybrid web recommender systems. In *The adaptive web*. Springer, 377–408.
[8] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Model. & User-Adapt. Inter.* (2015).
[9] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *SIGIR*.
[10] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In *SIGIR*. 315–324.
[11] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. Fast and accurate deep network learning by exponential linear units (elus). *ICLR* (2016).
[12] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*.
[13] 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 *KDD*.
[14] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends in HCI* 4, 2 (2011).
[15] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*.
[16] Asela Gunawardana and Christopher Meek. 2009. A unified approach to building hybrid recommender systems. In *RecSys*. 117–124.
[17] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*.
[18] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *AAAI*.
[19] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*.
[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.
[21] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, and others. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Sig. Proc.* (2012).
[22] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[24] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[25] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM*. ACM, 811–820.
[26] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 105–112.
[27] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.
[28] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM.
[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[30] M. J. Pazzani and D. Billsus. 2007. Content-Based Recommendation Systems. *The Adaptive Web LNCS* (2007), 325–341.
[31] Steffen Rendle, C. Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.
[32] Xiaoyuan Su and Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence* 4 (2009).
[33] Shiliang Sun. 2013. A survey of multi-view machine learning. *Neural Computing and Applications* 23, 7-8 (2013), 2031–2038.
[34] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-boosted latent topics: understanding users and items with ratings and reviews. In *IJCAI*.
[35] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.
[36] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. ACM, 1235–1244.
[37] Xinxi Wang and Ye Wang. 2014. Improving content-based and hybrid music recommendation using deep learning. In *ACM Multimedia*. 627–636.
[38] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*.
[39] Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*. 199–208.
[40] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.
[41] Wei Zhang, Quan Yuan, Jiawei Han, and Jianyong Wang. 2016. Collaborative Multi-Level Embedding Learning from Reviews for Rating Prediction. In *IJCAI*.
[42] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit Factor Models for Explainable Recommendation based on Phrase-level Sentiment Analysis. *SIGIR* (2014), 83–92.
[43] Kaiqi Zhao, Gao Cong, Quan Yuan, and Kenny Q Zhu. 2015. Sar: a sentiment-aspect-region model for user preference analysis in geo-tagged reviews. In *ICDE*.
[44] Wayne Xin Zhao, Jinpeng Wang, Yulan He, Ji-Rong Wen, Edward Y Chang, and Xiaoming Li. 2016. Mining Product Adopter Information from Online Reviews for Improving Product Recommendation. *TKDD* 10, 3 (2016), 29.
[45] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*.
[46] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. *NIPS* (2016).