Collaborative Knowledge Base Embedding for Recommender Systems

Fuzheng Zhang[†], Nicholas Jing Yuan[†], Defu Lian[‡], Xing Xie[†],Wei-Ying Ma[†]
[†]Microsoft Research
[‡]Big Data Research Center, University of Electronic Science and Technology of China
{fuzzhang,nicholas.yuan,xingx,wyma}@microsoft.com,
dove.ustc@gmail.com

ABSTRACT

Among different recommendation techniques, collaborative filtering usually suffer from limited performance due to the sparsity of user-item interactions. To address the issues, auxiliary information is usually used to boost the performance. Due to the rapid collection of information on the web, the knowledge base provides heterogeneous information including both structured and unstructured data with different semantics, which can be consumed by various applications. In this paper, we investigate how to leverage the heterogeneous information in a knowledge base to improve the quality of recommender systems. First, by exploiting the knowledge base, we design three components to extract items' semantic representations from structural content, textual content and visual content, respectively. To be specific, we adopt a heterogeneous network embedding method, termed as TransR, to extract items' structural representations by considering the heterogeneity of both nodes and relationships. We apply stacked denoising auto-encoders and stacked convolutional auto-encoders, which are two types of deep learning based embedding techniques, to extract items' textual representations and visual representations, respectively. Finally, we propose our final integrated framework, which is termed as **C**ollaborative **K**nowledge Base **E**mbedding (**CKE**), to jointly learn the latent representations in collaborative filtering as well as items' semantic representations from the knowledge base. To evaluate the performance of each embedding component as well as the whole system, we conduct extensive experiments with two real-world datasets from different scenarios. The results reveal that our approaches outperform several widely adopted state-of-the-art recommendation methods.

Keywords

Recommender Systems, Knowledge Base Embedding, Collaborative Joint Learning

1. INTRODUCTION

Due to the explosive growth of information, recommender systems have been playing an increasingly important role in online services. Among different recommendation strategies, collaborative filtering (CF) based methods, which make use of historical interactions or preferences, have made significant success [23]. However, CF methods usually suffer from limited performance when user-item interactions are very sparse, which is very common for scenarios such as online shopping where the item set is extremely large. In addition, CF methods can not recommend new items since these items have never received any feedbacks from users in the past. To tackle these problems, hybrid recommender systems, which combine collaborative filtering and auxiliary information such as item content, can usually achieve better recommendation results and have gained increasing popularity in recent years [2].

Over the past years, more and more semantic data are published following the Linked Data principles[1], by connecting various information from different topic domains such as people, books, musics, movies and geographical locations in a unified global data space. These heterogeneous data, interlinked with each other, forms a huge information resource repository called knowledge base. Several typical knowledge bases have been constructed, including academic projects such as YAGO[2], NELL[3], DBpedia[4], and DeepDive[5], as well as commercial projects, such as Microsoft's Satori[6] and Google's Knowledge Graph[7]. Using the heterogeneous connected information from the knowledge base can help to develop insights on problems which are difficult to uncover with data from a single domain [6]. To date, information retrieval [9], community detection [25], sentiment analysis [4] - to name a few - are the noteworthy applications that successfully leverage the knowledge base.

Actually, since a knowledge base provides rich information including both structured and unstructured data with different semantics, the usage of the knowledge base within the context of hybrid recommender systems are attracting increasing attention. For example, Yu et al. [30] uses a heterogeneous information network to represent users, items, item attributes, and the interlinked relationships in a knowledge base. They extract meta-path based latent features from the network structure and apply Bayesian ranking optimization based collaborative filtering to solve the entity recommendation problem. Grad-Gyenge et al. [11] extended collaborative filtering by adopting a spreading activation based technique to incorporate a knowledge base's network features for recommender systems' rating prediction task. However, previous studies have not

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: **http://dx.doi.org/10.1145/2939672.2939673**

[1]http://linkeddata.org/
[2]https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
[3]http://rtw.ml.cmu.edu/rtw/
[4]http://wiki.dbpedia.org/
[5]http://i.stanford.edu/hazy/deepdive/
[6]http://searchengineland.com/library/bing/bing-satori
[7]http://www.google.com/insidesearch/features/search/knowledge.html

fully exploited the potential of the knowledge base since they suffer from the following limitations: 1) only utilize the single network structure information of the knowledge base while ignore other important signals such as items' textual and visual information. 2) rely on heavy and tedious feature engineering process to extract features from the knowledge base.

To address the above issues, in this paper, we propose a novel recommendation framework to integrate collaborative filtering with items' different semantic representations from the knowledge base. For a knowledge base, except for the network structure information, we also consider items' textual content and visual content (e.g., movie's poster). To avoid heavy and tedious manual feature extractions, we design three embedding components to automatically extract items' semantic representations from the knowledge base's structural content, textual content and visual content, respectively. To be specific, we first apply a network embedding approach to extract items' structural representations by considering the heterogeneity of both nodes and relationships. Next, we adopt stacked denoising auto-encoders and stacked convolutional auto-encoders, which are two types of deep learning based embedding techniques, to extract items' textual representations and visual representations, respectively. Finally, to integrate collaborative filtering with items' semantic representations from the knowledge base smoothly, we propose our final framework, which is termed as Collaborative Knowledge Base Embedding (**CKE**), to learn different representations in a unified model jointly.

Our empirical studies consist of multiple parts. First, we conduct several experiments to evaluate the performance of three knowledge base embedding components, respectively. Next, we evaluate the effectiveness of our integrated framework by comparing with several competitive baselines.

The key contributions of this paper are summarized as the following:

- To the best of our knowledge, this is the first work leveraging structural content, textual content and visual content from the knowledge base for recommender systems.
- We apply embedding methods including heterogeneous network embedding and deep learning embedding to automatically extract semantic representations from the knowledge base. The learned representations may also be used for tasks other than recommendation.
- By performing knowledge base embedding and collaborative filtering jointly, **CKE** can simultaneously extract feature representations from the knowledge base and capture the implicit relationship between users and items.
- Based on two real-word datasets, we have conducted extensive experiments to evaluate the effectiveness of our framework. The results reveal that our approaches significantly outperform baseline methods.

The rest of this paper is organized as follows. Section 2 introduces the preliminary concepts and present our recommendation problem. Section 3 gives an overview of our framework. Section 4 delves into the usage of embedding components to extract representations from the knowledge base. In Section 5, we discuss how to effectively integrate collaborative filtering with knowledge base embedding into a unified model. The empirical results are discussed in Section 6, followed by a brief review of related work in Section 7 and a conclusion of this paper in Section 8.

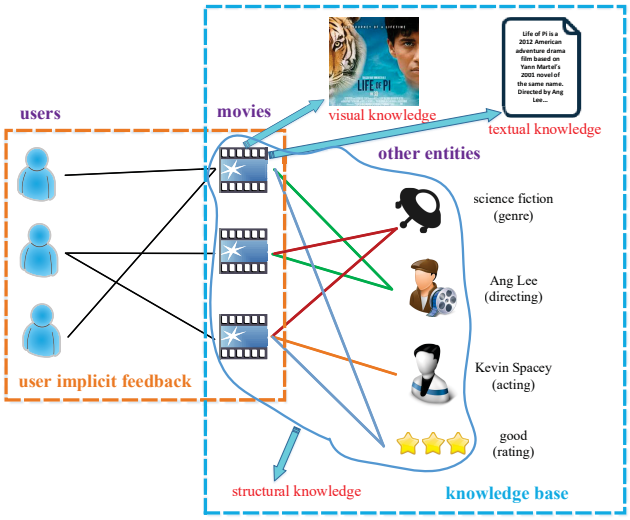


Figure 1: Illustration of a snippet of user implicit feedback data and knowledge base data.

2. PRELIMINARY

In this section, we will first clarify some terminologies used in this paper, and then explicitly present our problem.

2.1 User Implicit Feedback

The recommendation task considered in this paper are targeted for implicit feedback. Assume there are m users and n items, we define the user implicit feedback matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ as

$$R_{ij} = \begin{cases} 1, & \text{if (user } i, \text{item } j\text{) interactions have been observed;} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where the value 1 in matrix $\mathbf{R}$ represents the interactions between users and items, e.g., users watched a movie or users searched a book in the search engine. Note that the value 1 in the implicit feedback data does not mean that users actually like the items. Actually a user searched a book because he is interested in the book but he might probably dislike the book after browsing the related information on the Internet. Similarly, the value 0 in $\mathbf{R}$ does not mean that the users dislike the items, but can be regarded as a mixture of negative feedbacks (users are not interested in such items) and potential interactions (users are not aware of such items).

2.2 Knowledge Base

Actually, we are interested in leveraging the knowledge base for enhancing the quality of recommender systems, therefore items in recommender systems are mapped to entities in the knowledge base (e.g., a movie item can usually be mapped to an entity describing this movie), and these entities are termed as **item entities** in this article.

We consider the information stored in the knowledge base can be divided into three parts: structural knowledge, textual knowledge and visual knowledge. The detailed definition of each part is given as follows:

Definition 1: ***Structural knowledge***. This knowledge can be regarded as a heterogeneous network with multiple types of entities and multiple types of links to express the structure of the knowledge base. For movie recommendation, entities usually include movie items and corresponding attributes (e.g., the genre "science fiction" and the actor "Kevin Space"), and links describe the relationship between these entities (e.g., "acting" behavior and "rating" behav-

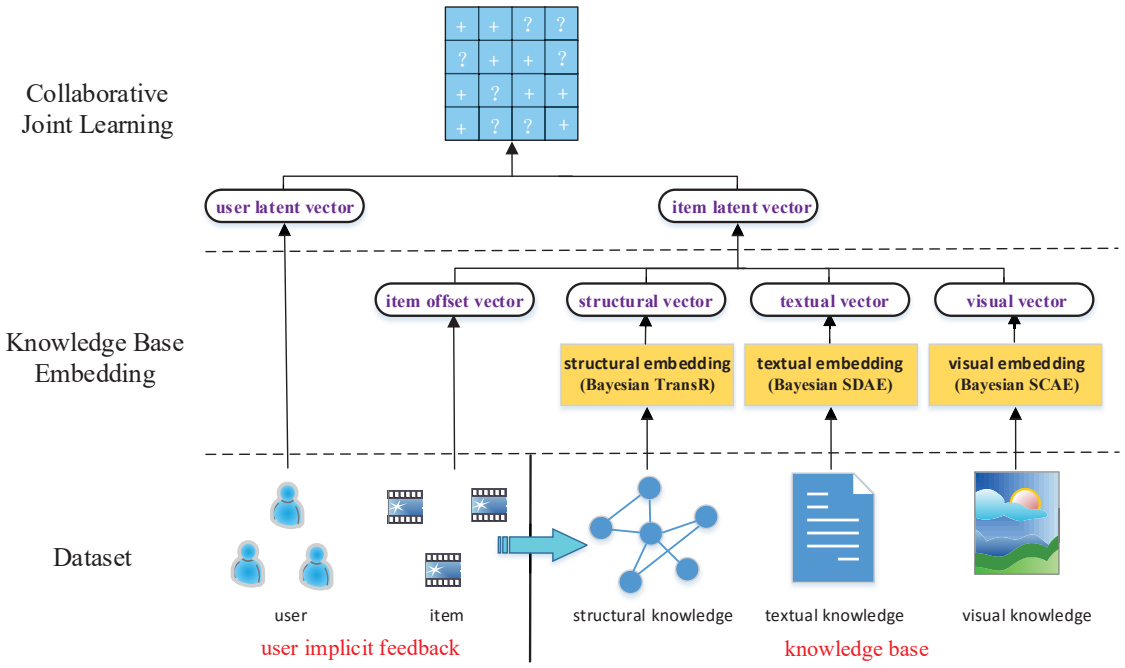


Figure 2: The flowchart of the proposed Collaborative Knowledge Base Embedding (CKE) framework for recommender systems

ior). The network structure implies some similarity between item entities, which is most probably useful for recommendation.

Definition 2: ***Textual Knowledge***. For an item entity such as book or movie in the knowledge base, we use the textual summary to represent the textual knowledge, which usually gives the main topics of this book or this movie.

Definition 3: ***Visual Knowledge***. For an item entity, except for previous textual description, there are usually some images in the knowledge base, we use a book's front cover image or a movie's poster image to represent its visual knowledge.

User implicit feedback interactions and structural knowledge serve as the structural features of an item, while textual knowledge and visual knowledge serve as the content features. A snippet of the knowledge base with three kinds of knowledge as well as user implicit feedback are presented in Figure 1.

2.3 Problem Formulation

We define our recommendation problem in this paper as follows:

Given a knowledge base with structural knowledge, textual knowledge and visual knowledge, as well as user implicit feedback, we aim to recommend each user with a ranked list of items he will be interested.

3. OVERVIEW

In this article, by fully exploiting the structural knowledge, textual knowledge and visual knowledge in the knowledge base, we propose a Collaborative Knowledge Base Embedding model (**CKE**) for supporting our recommendation task. Our model mainly consists of two steps: 1) knowledge base embedding and 2) collaborative joint learning.

In the knowledge base embedding step, we extract an item entity's three embedding vectors from structural knowledge, textual knowledge and visual knowledge, respectively. These embedding vectors indicate an item entity's latent representation in each domain. For structural embedding component, we apply a network embedding procedure (Bayesian TransR) to find the latent representation from the heterogeneous network in the structural knowledge. For textual embedding component, we apply an unsupervised deep learning model called Bayesian stacked denoising auto-encoder (Bayesian SDAE) [29] to find the latent representation from the textual knowledge. Similarly, we apply another unsupervised deep learning model called Bayesian stacked convolutional autoencoder (Bayesian SCAE) to find the latent representation from the visual knowledge.

In the collaborative joint learning step, an item's latent vector is finally represented as the integration of three embedding vectors from the knowledge base as well as a latent offset vector. The final item latent vector represents an item's knowledge from structural content, textual content, visual content as well as historical user-item interactions. Then we use collaborative filtering by optimizing the pair-wise ranking between items to learn both user latent vectors and item latent vectors. Final recommendation is generated from these user latent vectors and item latent vectors.

The flowchart of our framework is presented in Figure 2. Knowledge base embedding and collaborative joint learning will be detailed in Section 4 and Section 5, respectively.

4. KNOWLEDGE BASE EMBEDDING

In this section, by leveraging network embedding and deep learning embedding, we present the details of how we extract an item entity's representations from structural knowledge, textual knowledge and visual knowledge, respectively.

4.1 Structural Embedding

The heterogeneous network encodes structured information of entities and their rich relations. To capture this structured knowledge, a promising approach is to embed this heterogeneous network into a continuous vector space while preserving certain information of the network. In this subsection, we first briefly review a state-of-the-art network embedding method called TransR [15], and then give a Bayesian formulation of TransR for our task.

First, to represent the structural knowledge, we use an undirected

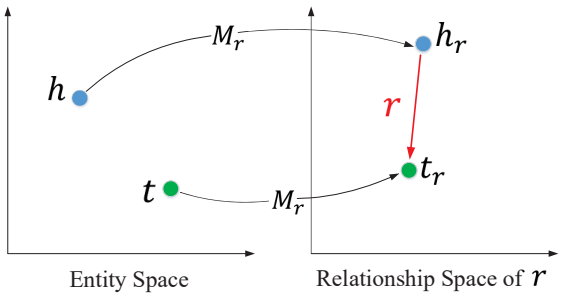


Figure 3: Illustration of TransR for structural embedding

graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_{|V|}\}$ is a set of vertices referring to different entities and $\mathcal{E}$ is a set of edges referring to different types of relation between these entities.

TransR [15] is a state-of-the-art embedding approach for heterogeneous network. Being different from other methods which assume embedding of entities and relations within the same space $\mathbb{R}^k$, TransR represents entities and relations in distinct semantic space bridged by relation-specific matrices. In TransR, for each triple (v_h, r, v_t) in the network (v_h and v_t are two linked entities, r is the type of edge between them), entities are embedded into vectors $\mathbf{v}_h, \mathbf{v}_t \in \mathbb{R}^k$ and relation is embedded into $\mathbf{r} \in \mathbb{R}^d$. For each relation r, we set a projection matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$, which projects entities from entity space to relation space. As shown in Figure 3, the projected vectors of entities are defined as

$$\mathbf{v}_h^r = \mathbf{v}_h \mathbf{M}_r, \qquad \mathbf{v}_t^r = \mathbf{v}_t \mathbf{M}_r. \tag{2}$$

The score function of this triple is correspondingly defined as

$$f_r(v_h, v_t) = ||\mathbf{v}_h^r + \mathbf{r} - \mathbf{v}_t^r||_2^2. \tag{3}$$

Similar to [22], we use a sigmoid function to calculate the pairwise triple ranking probability instead of margin-based objective function adopted in original TransR. Then we extend TransR to a Bayesian version and propose the generative process as follows:

1. For each entity v, draw $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I})$.
2. For each relation r, draw $\mathbf{r} \sim \mathcal{N}(\mathbf{0}, \lambda_r^{-1}\mathbf{I})$ and $\mathbf{M}_r \sim \mathcal{N}(\mathbf{0}, \lambda_M^{-1}\mathbf{I})$, respectively.
3. For each quadruple $(v_h, r, v_t, v_{t'}) \in \mathcal{S}$, draw from the probability $\sigma(f_r(v_h, v_t) - f_r(v_h, v_{t'}))$, where $\mathcal{S}$ is the set of quadruples satisfying that (v_h, r, v_t) is a correct triple and $(v_h, r, v_{t'})$ is an incorrect triple. $\sigma(x) := \frac{1}{1+e^{-x}}$ is the logistic sigmoid function.

It is routine to corrupt correct triple (v_h, r, v_t) by replacing one entity with another entity of the same type, and construct incorrect triple $(v_h, r, v_{t'})$. Note that step 3 implies the fact that when score function of a correct triple is larger than that of an incorrect triple, the quadruple is more likely to be sampled.

For each item entity j, we use embedding vector $\mathbf{v}_j$ from Bayesian TransR to denote its structural representation.

4.2 Textual Embedding

In this subsection, we investigate how to apply an unsupervised deep learning model called stacked denoising auto-encoders (SDAE) to get item entities' textual representations from the textual knowledge.

SDAE [27] is a feedback neural network for learning the representation of the corrupted input data by learning to predict the clean itself in the output. Before presenting the model detail, we give the notations used in SDAE. Assume the number of network layers is

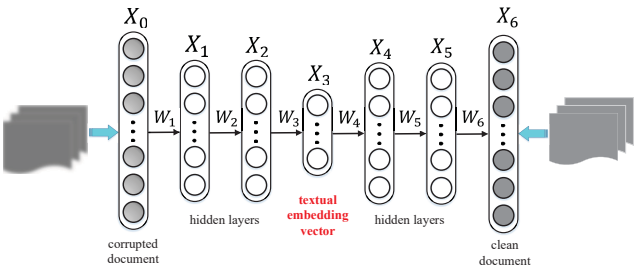


Figure 4: Illustration of a 6-layer SDAE for textual embedding

L_t, we use matrix $\mathbf{X}_l$ to represent the output of layer l in SDAE. Note that we use the last layer output $\mathbf{X}_{L_t}$ to represent the original clean textual knowledge of all item entities, where the j-th row is the bag-of-words vector $\mathbf{X}_{L_t, j*}$ for item entity j. Similarly, we use matrix $\mathbf{X}_0$ to represent the noise-corrupted matrix (randomly masking some entries of $\mathbf{X}_{L_t}$ by making them zero). $\mathbf{W}_l$ and $\mathbf{b}_l$ are weight parameter and bias parameter, respectively, for layer l.

Figure 4 gives the illustration of a 6-layer SDAE for our textual embedding component. As shown in this figure, the first $\frac{L_t}{2}$ layers of the network (from $\mathbf{X}_0$ to $\mathbf{X}_3$) usually acts as encoder part, which maps the corrupted input $\mathbf{X}_0$ to a latent compact representation $\mathbf{X}_3$, and the last $\frac{L_t}{2}$ layers (from $\mathbf{X}_3$ to $\mathbf{X}_6$) usually acts as the decoder part, which recovers the clean input $\mathbf{X}_6$ from the latent representation $\mathbf{X}_3$.

Similar to [29], given that both the clean input $\mathbf{X}_{L_t}$ and the corrupted input $\mathbf{X}_0$ are observed, we present the generative process of each layer l in Bayesian SDAE as follows:

1. For weight parameter $\mathbf{W}_l$, draw $\mathbf{W}_l \sim \mathcal{N}(\mathbf{0}, \lambda_W^{-1}\mathbf{I})$.
2. For bias parameter, draw $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_b^{-1}\mathbf{I})$.
3. For the output of the layer, draw $\mathbf{X}_l \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), \lambda_X^{-1}\mathbf{I})$

The embedding vector in the middle layer, i.e., $\mathbf{X}_{3,j*}$ in Figure 4, is used as the textual representation for item entity j.

4.3 Visual Embedding

In this subsection, similar to previous textual embedding part, we apply another unsupervised deep learning model, termed as stacked convolutional auto-encoders (SCAE), to extract item entities' semantic representations from the visual knowledge.

For visual objects, convolutional layers based deep learning architectures often beat the common fully connected architectures due to the fact that they can preserve the image's neighborhood relations and spatial locality in the latent higher-level feature representation [7]. Furthermore, convolutional layers restrict the number of free parameters by sharing weights so that they scale well to high-dimensional image content. Given above, by following the work in [16], we adopt the stacked convolutional auto-encoders (SCAE) by using convolutional hidden layers to replace fully-connected layers in previous SDAE.

Assume that there are L_v layers in SCAE, similar to the notation in SDAE, we use a 4-dimensional tensor $\mathbf{Z}_{L_v}$ to denote the collection of clean images, where the j-th row is a 3-dimensional tensor $\mathbf{Z}_{L_v, j*}$ of raw pixel representation in RGB color space for item entity j. Similarly, we use $\mathbf{Z}_0$ to denote the corrupted images (randomly masking some entries of $\mathbf{Z}_{L_v}$ by adding Gaussian noise). Next, for each layer l, we use $\mathbf{Z}_l$ to represent the output, $\mathbf{Q}_l$ to represent the weight parameter, and $\mathbf{c}_l$ to represent the bias parameter.

In SCAE, we set layer $\frac{L_v}{2}$ and layer $\frac{L_v}{2} + 1$ as fully connected layers, while other layers as convolutional layers. Figure 5 gives

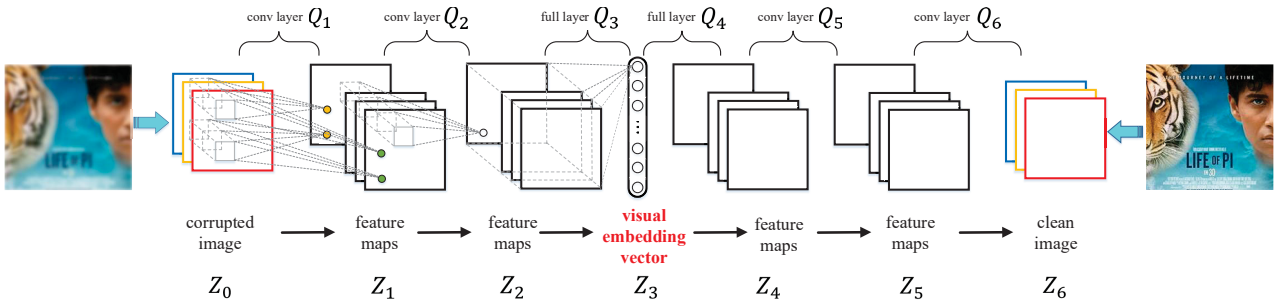


Figure 5: Illustration of a 6-layer SCAE for visual embedding

the illustration of a 6-layer SCAE, which also consists of encoder part and decoder part. As shown in the figure, encoder part consists of two convolutional layers (from $\mathbf{Z}_0$ to $\mathbf{Z}_2$) and a fully connected layers ($\mathbf{Z}_2$ to $\mathbf{Z}_3$). Similarly, decoder part consists of a fully connected layer ($\mathbf{Z}_3$ to $\mathbf{Z}_4$) and two following deconvolutional layers (from $\mathbf{Z}_4$ to $\mathbf{Z}_6$). Note that the output of the middle hidden layer $\mathbf{Z}_3$ is a matrix, which denotes the collection of all item entities' visual embedding vectors, while the output of other hidden layers are usually termed as feature maps [7], which are 4-dimensional tensors generated from convolutional layers. The mapping for a convolutional layer is given as

$$\mathbf{Z}_l = \sigma(\mathbf{Q} * \mathbf{Z}_{l-1} + \mathbf{c}_l) \tag{4}$$

where $*$ denotes the convolutional operator, which can preserve the local connectivity of previous output. More details about convolutional operator can be referred to [20].

Similar to textual embedding component, given both clean image input $\mathbf{Z}_{L_v}$ and corrupted input $\mathbf{Z}_0$, we present the generative process of each layer l in Bayesian SCAE as follows:

1. For weight parameter, draw $\mathbf{Q}_l \sim \mathcal{N}(\mathbf{0}, \lambda_Q^{-1}\mathbf{I})$.
2. For bias parameter, draw $\mathbf{c}_l \sim \mathcal{N}(\mathbf{0}, \lambda_c^{-1}\mathbf{I})$.
3. For the output of the layer,
   (a) If layer l is a fully connected layer: draw $\mathbf{Z}_l \sim \mathcal{N}(\sigma(\mathbf{Z}_{l-1}\mathbf{Q}_l + \mathbf{c}_l), \lambda_Z^{-1}\mathbf{I})$,
   (b) Else: draw $\mathbf{Z}_l \sim \mathcal{N}(\sigma(\mathbf{Z}_{l-1} * \mathbf{Q}_l + \mathbf{c}_l), \lambda_Z^{-1}\mathbf{I})$.

The embedding vector in the middle layer, i.e., $\mathbf{Z}_{3,j*}$ in Figure 5, is used as the visual representation for item entity j.

5. COLLABORATIVE JOINT LEARNING

In this section, in order to integrate collaborative filtering with items' embedding representations from the knowledge base, we propose the collaborative joint learning procedure in our **CKE** framework.

Given user implicit feedback $\mathbf{R}$, motivated by [22], we consider the pair-wise ranking between items for the learning approach. To be more specific, when $R_{ij} = 1$ and $R_{ij'} = 0$, we say that user i prefers item j over j', and then use $p(j > j'; i|\theta)$ to denote the pair-wise preference probability, where θ represents the model parameters. In collaborative filtering, we use a latent vector $\mathbf{u}_i$ as the representation for user i, and a latent vector $\boldsymbol{\eta}_j$ as the representation for item j. To simultaneously capture an item's latent representation in collaborative filtering and representations in the knowledge base, the item latent vector can be re-expressed as

$$\mathbf{e}_j = \boldsymbol{\eta}_j + \mathbf{v}_j + \mathbf{X}_{\frac{L_t}{2},j*} + \mathbf{Z}_{\frac{L_v}{2},j*} \tag{5}$$

Then the pair-wise preference probability can be given as

$$p(j > j'; i|\theta) = \sigma(\mathbf{u}_i^T\mathbf{e}_j - \mathbf{u}_i^T\mathbf{e}_{j'}) \tag{6}$$

Using Bayesian TransR, Bayesian SDAE, and Bayesian SCAE in knowledge base embedding step as components, the generative process of our framework **CKE** by using collaborative join learning is given as follows:

1. Considering the structural knowledge,
   (a) For each entity v, draw $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \lambda_v^{-1}\mathbf{I})$.
   (b) For each relation r, draw $\mathbf{r} \sim \mathcal{N}(\mathbf{0}, \lambda_r^{-1}\mathbf{I})$ and $\mathbf{M}_r \sim \mathcal{N}(\mathbf{0}, \lambda_M^{-1}\mathbf{I})$, respectively.
   (c) For each quadruple $(v_h, r, v_t, v_{t'}) \in \mathcal{S}$, draw from the probability $\sigma(f_r(v_h, v_t) - f_r(v_h, v_{t'}))$.
2. Considering the textual knowledge, for each layer l in SDAE,
   (a) For weight parameter $\mathbf{W}_l$, draw $\mathbf{W}_l \sim \mathcal{N}(\mathbf{0}, \lambda_W^{-1}\mathbf{I})$.
   (b) For bias parameter, draw $\mathbf{b}_l \sim \mathcal{N}(\mathbf{0}, \lambda_b^{-1}\mathbf{I})$.
   (c) For the output of the layer, draw $\mathbf{X}_l \sim \mathcal{N}(\sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l), \lambda_X^{-1}\mathbf{I})$
3. Considering the visual knowledge, for each layer l in SCAE,
   (a) For weight parameter, draw $\mathbf{Q}_l \sim \mathcal{N}(\mathbf{0}, \lambda_Q^{-1}\mathbf{I})$.
   (b) For bias parameter, draw $\mathbf{c}_l \sim \mathcal{N}(\mathbf{0}, \lambda_c^{-1}\mathbf{I})$.
   (c) For the output of the layer,
      i. If layer l is a fully connected layer: draw $\mathbf{Z}_l \sim \mathcal{N}(\sigma(\mathbf{Z}_{l-1}\mathbf{Q}_l + \mathbf{c}_l), \lambda_Z^{-1}\mathbf{I})$,
      ii. Else: draw $\mathbf{Z}_l \sim \mathcal{N}(\sigma(\mathbf{Z}_{l-1} * \mathbf{Q}_l + \mathbf{c}_l), \lambda_Z^{-1}\mathbf{I})$.
4. For each item j, draw a latent item offset vector $\boldsymbol{\eta}_j \sim \mathcal{N}(\mathbf{0}, \lambda_I^{-1}\mathbf{I})$, and then set the item latent vector as: $\mathbf{e}_j = \boldsymbol{\eta}_j + \mathbf{v}_j + \mathbf{X}_{\frac{L_t}{2},j*} + \mathbf{Z}_{\frac{L_v}{2},j*}$.
5. For each user i, draw a user latent vector as $\mathbf{u}_i \sim \mathcal{N}(\mathbf{0}, \lambda_U^{-1}\mathbf{I})$.
6. For each triple $(i, j, j') \in \mathcal{D}$, draw from the probability $\sigma(\mathbf{u}_i^T\mathbf{e}_j - \mathbf{u}_i^T\mathbf{e}_{j'})$.

Here, $\mathcal{D}$ is a collection of triples, where each triple (i, j, j') satisfies that $R_{ij} = 1$ and $R_{ij'} = 0$ (j' is randomly sampled from user i's uninterested items). Note that $\mathbf{v}_j$, $\mathbf{X}_{\frac{L_t}{2},j*}$ and $\mathbf{Z}_{\frac{L_v}{2},j*}$ serve as the bridges between implicit feedback preference and structural knowledge, textual knowledge as well as visual knowledge, respectively.

Learning the parameters. Computing the full posterior of the parameters is intractable. Like [28], maximizing the posterior probability of $\mathbf{u}$, $\mathbf{e}$, $\mathbf{r}$, $\mathbf{M}$, $\mathbf{W}$, $\mathbf{b}$, $\mathbf{Q}$ and $\mathbf{c}$ is equivalent to maximizing the log-likelihood as follows:

$$\begin{aligned}\mathcal{L} = & \sum_{(i,j,j')\in\mathcal{D}} \ln \sigma(\mathbf{u}_i^T\mathbf{e}_j - \mathbf{u}_i^T\mathbf{e}_{j'}) - \frac{\lambda_X}{2}\sum_l ||\sigma(\mathbf{X}_{l-1}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_l||_2^2 \\ & + \sum_{(v_h,r,v_t,v_{t'})\in\mathcal{S}} \ln \sigma(||\mathbf{v}_h\mathbf{M}_r + \mathbf{r} - \mathbf{v}_t\mathbf{M}_r||_2^2 - ||\mathbf{v}_h\mathbf{M}_r + \mathbf{r} - \mathbf{v}_{t'}\mathbf{M}_r||_2^2) \\ & - \frac{\lambda_Z}{2}\sum_{l\notin\{\frac{L_v}{2},\frac{L_v}{2}+1\}} ||\sigma(\mathbf{Z}_{l-1} * \mathbf{Q}_l + \mathbf{c}_l) - \mathbf{Z}_l||_2^2 - \frac{\lambda_U}{2}\sum_i ||\mathbf{u}_i||_2^2 \\ & - \frac{\lambda_Z}{2}\sum_{l\in\{\frac{L_v}{2},\frac{L_v}{2}+1\}} ||\sigma(\mathbf{Z}_{l-1}\mathbf{Q}_l + \mathbf{c}_l) - \mathbf{Z}_l||_2^2 - \frac{\lambda_v}{2}\sum_v ||\mathbf{v}||_2^2 \\ & - \frac{1}{2}\sum_l(\lambda_W||\mathbf{W}_l||_2^2 + \lambda_b||\mathbf{b}_l||_2^2) - \frac{1}{2}\sum_l(\lambda_Q||\mathbf{Q}_l||_2^2 + \lambda_c||\mathbf{c}_l||_2^2) \\ & - \frac{\lambda_I}{2}\sum_j ||\mathbf{e}_j - \mathbf{v}_j - \mathbf{X}_{\frac{L_t}{2},j*} - \mathbf{Z}_{\frac{L_v}{2},j*}||_2^2 \\ & - \frac{\lambda_r}{2}\sum_r ||\mathbf{r}||_2^2 - \frac{\lambda_M}{2}\sum_r ||\mathbf{M}_r||_2^2 \end{aligned} \tag{7}$$

To maximize the objective in Eq. (7), we employ a stochastic gradient descent (SGD) algorithm similar to [22]. In each iteration, for a randomly sampled triple $(i, j, j') \in \mathcal{D}$, we find the subset $\mathcal{S}_{j,j'} \in \mathcal{S}$ satisfying that each quadruple in $\mathcal{S}_{j,j'}$ contains item j or item j'. Then we perform a SGD update for each parameter using the gradient of the corresponding objective function [10].

Prediction The final item recommendation for a user i is given according to the following ranking criterion:

$$i : j_1 > j_2 > \ldots > j_n \rightarrow \mathbf{u}_i^T\mathbf{e}_{j_1} > \mathbf{u}_i^T\mathbf{e}_{j_2} > \ldots > \mathbf{u}_i^T\mathbf{e}_{j_n} \tag{8}$$

6. EXPERIMENTS

In this section, we evaluate our proposed framework on two real-world datasets for both movie and book recommendation scenarios. The experimental results demonstrate evidence of significant improvement over many competitive baselines.

6.1 Datasets Description

To demonstrate the effectiveness of the proposed collaborative knowledge base embedding for recommendation framework, we use two datasets from different domains (movie and book) for empirical studies. The first dataset, MovieLens-1M[8], consists of $1M$ ratings with 6,040 users and 3,706 movies. Similar to [31], in order to be consistent with the implicit feedback setting, we extract only positive ratings (rating 5) for training and testing. After removing users with less than 3 positive ratings, we have 5,883 users, 3,230 movies, and 226,101 ratings in the final dataset. The second dataset, termed as IntentBooks, is collected from Microsoft's Bing search engine and Microsoft's Satori knowledge base [1]. In this dataset, users' interests for books are extracted from click/query actions, e.g., if a user has conducted queries with "Harry Potter" or clicked documents containing that name, this user might be interested for the related book entity. To reduce the name conflict issue (e.g., instead of the book, "Harry Potter" might actually mean the related movie), we extracted user's book interests by combining unsupervised similarity computation with supervised classification by following [13]. Moreover, to validate the effectiveness of book interest extraction, we randomly selected 200 extracted book interest instances and manually labeled each one as true or false. The result shows that the precision is 91.5%, which we believe is accurate enough for subsequent experiments. We sample the user implicit feedback data from Bing's search log from Sep. 2014 to Jun. 2015. After removing users with less than 5 book interests, we finally have 92,564 users, 18,475 books, and 897,871 user-book interests.

[8]http://grouplens.org/datasets/movielens/1m/

Table 1: Detailed statistics of the two datasets

	MovieLens-1M	IntentBooks
#user	5,883	92,564
#item	3,230	18,475
#interactions	226,101	897,871
#sk nodes	84,011	26,337
#sk edges	169,368	57,408
#sk edge types	10	6
#tk items	2,752	17,331
#vk items	2,958	16,719

We also use Satori knowledge base to extract structural knowledge, textual knowledge and visual knowledge for these two datasets. First, we applied a two staged method described in [24] (including both title match and attributes match) to map each movie from MovieLens-1M dataset to an entity in the knowledge base (note that a book in the IntentBooks dataset is already an entity in the knowledge base, therefore matching step is ignored). We explicitly observed 200 paired results, where 92% of the pairs are correctly matched (the match precision is good enough for later process). In addition, we find that only 134 movies can not be mapped to any movie entities from the knowledge base. Next, to build the structural knowledge, we extract a subgraph from the knowledge base which contains item entities, entities which are 1-step away from item entities, and the corresponding relationships. For movie entities, the 1-step entities include genre, director, writer, actors, language, country, production date, rating, nominated awards, and received awards; for book entities, the 1-step entities include genre, author, publish date, belonged series, language, and rating. Then for text knowledge, we follow the word hashing procedure as that in [12] to preprocess the text information extracted from the plots of the movies and the descriptions of the books. Finally, for visual knowledge, we use the poster image of a movie entity and the front cover image of a book entity, where the finally used visual input are images that are reshaped to the $3 \times 64 \times 64$ tensor format in the RGB space. Some detailed statistics of the two datasets are summarized in Table 1. For example, "#sk nodes" indicates the total number of nodes in the extracted structural knowledge, "#tk items" indicates the number of items having textual knowledge, and "#vk items" indicates the number of items having visual knowledge.

6.2 Evaluation Schema

As discussed in [19], precision is not a suitable performance measure for implicit feedback recommendation. Therefore, in our experiments, we use MAP@K (mean average precision) [26] and Recall@K to evaluate the performance of the top K recommendation. For each dataset, similar to [14], we randomly select 70% items associated with each user to constitute the training set and use all the remaining as the test set. For each evaluation scenario, we repeat the evaluation five times with different randomly selected training sets and the average performance is reported in the following parts. For each dataset, we also use a validation set from the training set to find the optimal hyperparameters for our methods as well as baselines introduced in later parts. In the following reported results, hyperparameter settings of our methods are given in Table 2 since best performance is achieved. In Table 2, dim denotes the latent dimension, ϵ denotes the noise masking level, σ denotes the

Table 2: Hyperparameter settings of our framework for the two datasets. cf, sk, tk and vk indicate the parameters in the component of collaborative filtering, structural knowledge embedding, textual knowledge embedding and visual knowledge embedding, respectively.

	MovieLens-1M	IntentBooks
cf	dim=150,λ_U=λ_I=0.0025	dim=100,λ_U=λ_I=0.005
sk	λ_v=λ_r=0.001,λ_M=0.01	λ_v=λ_r=0.001,λ_M=0.1
tk	λ_W=λ_b=0.01, λ_X=0.0001, ϵ=0.2, L_t=4, N_l=300	λ_W=λ_b=0.01, λ_X=0.001, ϵ=0.1, L_t=6, N_l=200
vk	λ_Q=λ_c=0.01, λ_Z=0.0001, σ=3, L_v = 6, N_f=20, S_f=(5,5)	λ_Q=λ_c=0.01, λ_Z=0.001, σ=2, L_v = 8, N_f=20, S_f=(5,5)

standard deviation for image's Gaussian filter noise, L_t and L_v denote the number of layers, N_l denotes the number of hidden units when layer l is not a middle layer or an output layer in textual embedding's SDAE step, N_f and S_f denote the number and the size of filter maps in each convolutional layer of visual embedding's SCAE step, respectively. Note that for the purpose of keeping the collaborative filtering part the same in different factorization methods, the latent dimension of compared baselines in the following subsections are set the same as that in Table 2, other hyperparameters of baselines are determined by grid search.

In the following subsections, we will evaluate the proposed framework according to four aspects. First, we evaluate the recommendation performance with regard to structural knowledge usage, textual knowledge usage, and visual knowledge usage, respectively. Then, we compare the joint model **CKE** with state-of-the-art baselines to demonstrate the effectiveness of our system.

6.3 Study of Structural Knowledge Usage

In this subsection, to study the performance of our structural knowledge usage, we only incorporate structural knowledge embedding component into collaborative filtering in the joint learning process. We compare our method, termed as "CKE(S)", against the following baselines:

- **BPRMF**: We use Bayesian Personalized Ranking based Matrix Factorization [22], which is based on users' pair-wise preference, as the single collaborative filtering method. Compared with our method CKE(S), this baseline completely ignores the usage of structural knowledge.
- **BPRMF+TransE**: This method combines BPRMF and a network embedding method called TransE [3] for joint learning. This baseline actually uses the same settings as our method CKE(S), except that it uses TransE, which ignores the heterogeneity of entities and relations, instead of TransR to embed structural knowledge.
- **PRP**: PageRank with Priors [17] integrates user-item relation and structural knowledge into a unified homogeneous graph and then perform pagerank for each user with a personalized initial probability distribution.
- **PER**: Personalized Entity Recommendation [30] treats the structural knowledge as a heterogeneous information network and extract meta-path based latent features to represent the connectivity between users and items along different types of relation paths. In this paper, we use all the item-attribute-item formatted meta-path features (e.g., "movie-genre-movie").
- **LIBFM(S)**: LIBFM [21] is a state-of-the-art feature based factorization model. In this paper, LIBFM(S) implies that we use an item's attributes in the structural knowledge as raw features to feed into LIBFM.

The results of different methods are given in Figure 6(a), Figure 7(a), Figure 8(a), and Figure 9(a). The results precipitate several observations for both two datasets, which we summarize as: 1) BPRMF performs the worst among all approaches. Since BPRMF is the only one which totally ignores structural knowledge, the results imply that the additional usage of structural knowledge can significantly improve the recommendation performance. 2) PRP performs worse than other approaches leveraging structural knowledge. This is because among these, PRP is the only one which does not leverage factorization, which can capture the latent low-rank approximation of user-item interactions under the sparseness of the datasets. 3) BPRMF+TransE outperforms LIBFM(S) and PER, which demonstrates that instead of directly using structural knowledge in a feature-engineering way, network embedding can capture the semantic representation in a more reasonable way and thus improve the recommendation quality. 4) Our method CKE(S) beats BPRMF+TransE, which implies that by using TransR, there is still room for improvement when considering the heterogeneity of network embedding.

6.4 Study of Textual Knowledge Usage

In this subsection, we investigate the performance of our textual knowledge usage. Specifically, we use the textual knowledge embedding component and collaborative filtering in the joint learning process. We compare our method, which is termed as "CKE(T)", against BPRMF as well as the following baselines:

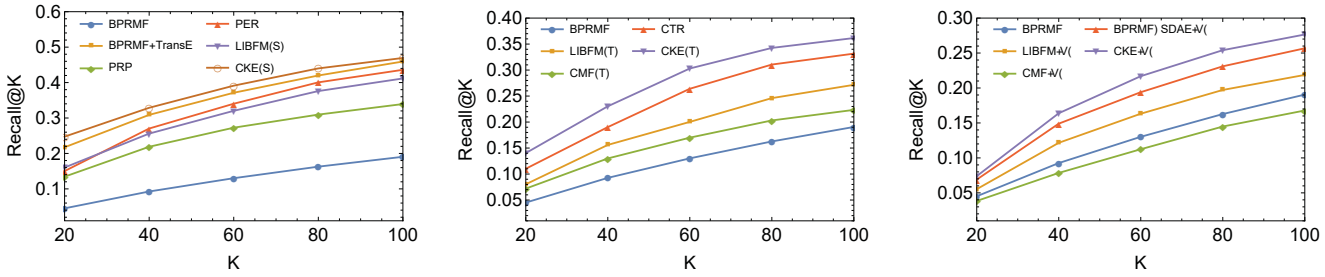
- **LIBFM(T)**: This method is the same as LIBFM(S) introduced in Section 6.3, except that bag of words in the textual knowledge are now used as raw features to feed into LIBFM.
- **CMF(T)**: Collective Matrix Factorization [21] combines different kinds of data sources by simultaneously factorizing multiple matrix. In this paper, CMF(T) implies that the two matrices we use are user-item matrix and item-word matrix.
- **CTR**: Collaborative Topic Regression [28], which is state-of-the-art method leveraging textual information for recommendation, integrates collaborative filtering and topic modeling simultaneously.

As shown in Figure 6(b), Figure 7(b), Figure 8(b), and Figure 9(b), the comparison of the results presents the following observations: 1) CKE(S) outperforms CKE(T) and LIBFM(S) outperforms LIBFM(T), which implies that compared to structural knowledge, textual knowledge has weaker improvement on recommendation performance. 2) LIBFM(T), CTR and CKE(T) usually give better performance than CMF, which reveals that the direct factorization of item-word matrix can not make the full use of textual information. 3) CTR is a strong baseline and sometimes even achieve the best performance in the IntentBooks dataset. However, our method CKE(T) can beat CTR most of the time, which demonstrates that compared to topic modeling, deep learning embedding are really good at extracting text's semantic representation by delving into deep structure.

6.5 Study of Visual Knowledge Usage

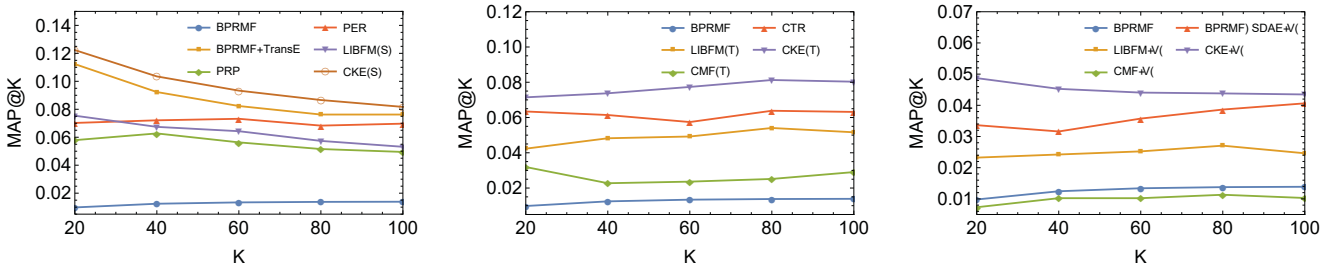
In this subsection, we focus on studying the performance of our visual knowledge usage. As before, we use the visual knowledge embedding component and collaborative filtering in the joint learning. We compare our method, which is named as "CKE(V)" against BPRMF as well as the following baselines:

- **LIBFM(V)**: This method is the same as LIBFM(S) in Section 6.3, except that flattened raw pixel representation in RGB color space are used as raw features.



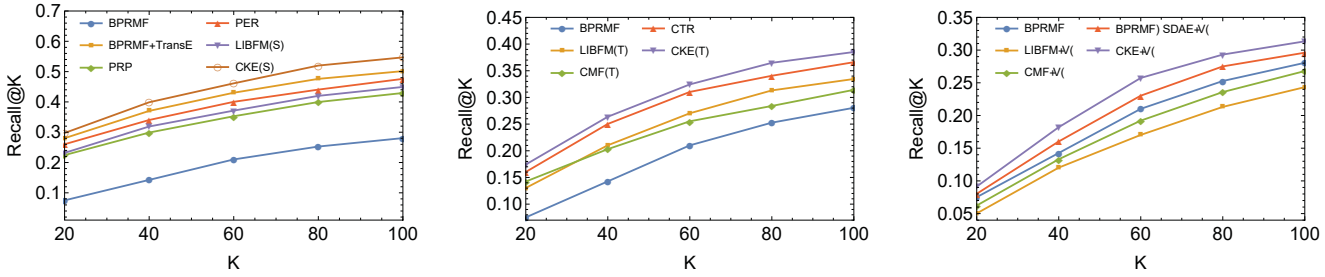
(a) Usage of Structural Knowledge (b) Usage of Textual Knowledge (c) Usage of Visual Knowledge

Figure 6: Recall@K results comparison between our methods using each component in knowledge base embedding and related baselines for dataset MovieLens-1M.



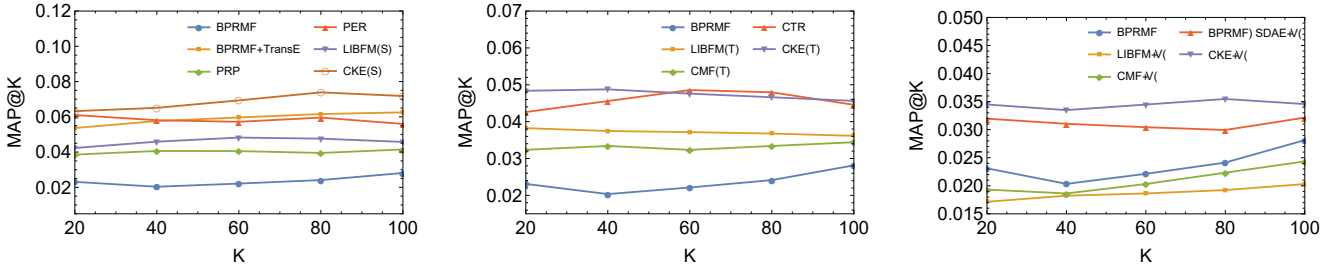
(a) Usage of Structural Knowledge (b) Usage of Textual Knowledge (c) Usage of Visual Knowledge

Figure 7: MAP@K results comparison between our methods using each component in knowledge base embedding and related baselines for dataset MovieLens-1M.



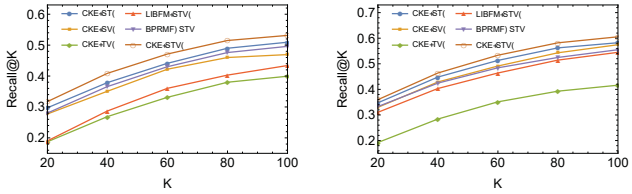
(a) Usage of Structural Knowledge (b) Usage of Textual Knowledge (c) Usage of Visual Knowledge

Figure 8: Recall@K results comparison between our methods using each component in knowledge base embedding and related baselines for dataset IntentBooks.



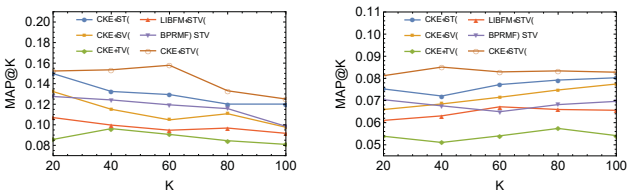
(a) Usage of Structural Knowledge (b) Usage of Textual Knowledge (c) Usage of Visual Knowledge

Figure 9: MAP@K results comparison between our methods using each component in knowledge base embedding and related baselines for dataset IntentBooks.



(a) MovieLens-1M (b) IntentBooks

Figure 10: Recall@K results comparison between our framework and related baselines for both datasets.



(a) MovieLens-1M (b) IntentBooks

Figure 11: MAP@K results comparison between our framework and related baselines for both datasets.

- **CMF(V)**: This method uses the same setting as CMF(T) described in Section 6.4, except that we use user-item matrix and item-pixel matrix for simultaneous factorization.
- **BPRMF+SDAE(V)**: This method uses the same settings as our method CKE(V), except that it uses stacked denoising auto-encoder to embed visual knowledge. This is for evaluating whether the convolutional layers are effective for visual knowledge embedding.

The results of different approaches are shown in Figure 6(c), Figure 7(c), Figure 8(c), and Figure 9(c), which provide us the following observations: 1) Compared to structural knowledge and textual knowledge, the performance improvement with the usage of visual knowledge is limited but still significant. 2) CKE(V) and BPRMF+SDAE(V) outperform other approaches, which demonstrates the superiority of deep network for visual knowledge embedding. 3) The performance gap between CKE(V) and BPRMF+SDAE(V) is still significant, which reveals that convolutional layers are more suitable for extracting visual representation.

6.6 Study of The Whole Framework

Finally, we evaluate the performance of our whole framework. We compare our ultimate model using three components in knowledge base embedding, denoted as "CKE(STV)" against the following baselines:

- **CKE(ST)**, **CKE(SV)**, **CKE(TV)**: CKE(ST) uses the same setting as CKE(STV), except that it only incorporates structural knowledge and textual knowledge, this is for evaluating whether the additional usage of visual knowledge is effective. The definition is similar to CKE(SV) and CKE(TV).
- **LIBFM(STV)**: This method uses the same setting as LIBFM(S) mentioned in Section 6.4, except that structural knowledge, textual knowledge and visual knowledge are all used as features.
- **BPRMF+STV**: This method uses the same setting as CKE(STV), except that it learns collaborative filtering and three knowledge base embedding components separately. This is for evaluating whether joint learning is effective.

The results are shown in Figure 10 and Figure 11, which give the following observations: 1) CKE(STV) outperforms CKE(ST), CKE(SV) and CKE(TV). This implies that the additional usage of each kind of knowledge can improve the recommendation performance, which demonstrate the insight of our framework combining different kinds of knowledge together for enhancing recommendation. 2) CKE(STV) outperforms LIBFM(STV), which demonstrates that compared to direct usage of the knowledge base in a feature-engineering way, the embedding components can capture the knowledge base's semantic representations more effectively and thus improve the recommendation performance. 3) CKE(STV) still archives better performance than BPRMF+STV, which suggests that the joint learning of collaborative filtering and knowledge base embedding can target the recommendation optimization task in a more direct way and thus improve the recommendation quality.

7. RELATED WORK

7.1 Knowledge Base for Recommendation

The usage of the knowledge base within the context of recommender systems are attracting increasing attention in recent years. For example, Cheekula et al. [5] explores the hierarchical category knowledge derived from the pruned DBpedia knowledge base and applies the spreading activation algorithm to identify personalized entities as recommendation. Passant [18] computes the semantic distance in the graphical structure of a knowledge base and used this distance measure to build music recommendation systems. Compared to previous works which investigated this problem mainly by leveraging the structure of a knowledge base, we present a framework to integrate the heterogeneous knowledge from structure, text as well as visual content for recommendation.

7.2 Structural Knowledge Embedding

Previous works of structural knowledge embedding mainly aim to embed entities and relations into a continuous vector space and model the semantics of structure in that space. For example, TransE [3] represents a relation by a translation vector so that the pair of embedded entities in a triple can be connected with low error. However, TransE has flaws in dealing with many-to-many relations for the heterogeneity of structural information in a knowledge base. To solve the issue of TransE, TransR [15] steps to propose that entities and relations are completely different objects and models entities and relations in distinct spaces and performs translation in relation space by using a translation matrix. In this paper, we integrate TransR with our recommendation task to fully leverage the structural knowledge in the knowledge base.

7.3 Deep Learning for Recommendation

With the success of deep learning in recent years, there is a marked switch from hand-crafted features to those that are learned from raw data in recommendation research. For example, Elkahky et al. [8] proposes a multi-view deep learning model to learn user features and item features from different domains, and uses the enriched feature representation to improve the recommendation quality across all the domains. Wang [29] significantly improve the recommendation performance by jointly performing deep representation learning for the textual content information and collaborative filtering for the ratings matrix. Our framework distinguishes itself from the above-mentioned works in the following aspects: 1) We specifically apply stacked convolutional auto-encoder to extract the semantic representation of visual content, which has not been exploited in previous works. 2) We design a joint model by inte-

grating collaborative filtering and the heterogeneity of structural, textual and visual knowledge to boost the recommendation quality.

8. CONCLUSION

This paper proposes a hybrid recommender system termed as **CKE**, which integrates collaborative filtering and knowledge base for recommendation. Following this framework, we first design three components, which leverage heterogeneous network embedding and deep learning embedding approaches, to automatically extract semantic representations from structural knowledge, textual knowledge and visual knowledge in the knowledge base, respectively. Next, we combine collaborative filtering and knowledge base embedding components into a unified framework and learn different representations jointly. The extensive experiments we have conducted validated the effectiveness of our **CKE** framework. Besides, this research sheds new light on the usage of heterogeneous information in the knowledge base, which can be consumed in more application scenarios.

9. REFERENCES

[1] ahmet uyar, f. m. a. Evaluating search features of google knowledge graph and bing satori. *Online Information Review* (2015).

[2] Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems 46* (2013), 109–132.

[3] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems* (2013), 2787–2795.

[4] Cambria, E., Schuller, B., Liu, B., Wang, H., and Havasi, C. Knowledge-based approaches to concept-level sentiment analysis. *IEEE Intelligent Systems*, 2 (2013), 12–14.

[5] Cheekula, S. K., Kapanipathi, P., Doran, D., Jain, P., and Sheth, A. Entity recommendations using hierarchical knowledge bases.

[6] Chui, M., Manyika, J., and Kuiken, S. What executives should know about open data. *McKinsey Quarterly, January 2014* (2014).

[7] Ciresan, D. C., Meier, U., Masci, J., Maria Gambardella, L., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22 (2011), 1237.

[8] Elkahky, A. M., Song, Y., and He, X. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee (2015), 278–288.

[9] Freitas, A., Oliveira, J. G., OâĂŹRiain, S., Curry, E., and Da Silva, J. C. P. Querying linked data using semantic relatedness: a vocabulary independent approach. In *Natural Language Processing and Information Systems*. Springer, 2011, 40–51.

[10] Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, ACM (2011), 305–308.

[11] Grad-Gyenge, L., Filzmoser, P., and Werthner, H. Recommendations on a knowledge graph.

[12] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, ACM (2013), 2333–2338.

[13] Ji, H., and Grishman, R. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics (2011), 1148–1158.

[14] Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., and Rui, Y. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM (2014), 831–840.

[15] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI* (2015).

[16] Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*. Springer, 2011, 52–59.

[17] Nguyen, P., Tomeo, P., Di Noia, T., and Di Sciascio, E. An evaluation of simrank and personalized pagerank to build a recommender system for the web of data. In *Proceedings of the 24th International Conference on World Wide Web Companion*, International World Wide Web Conferences Steering Committee (2015), 1477–1482.

[18] Passant, A. dbrecâĂŤmusic recommendations using dbpedia. In *The Semantic Web–ISWC 2010*. Springer, 2010, 209–224.

[19] Purushotham, S., Liu, Y., and Kuo, C.-C. J. Collaborative topic regression with social matrix factorization for recommendation systems. *arXiv preprint arXiv:1206.4684* (2012).

[20] Ren, J. S., and Xu, L. On vectorization of deep convolutional neural networks for vision tasks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[21] Rendle, S. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST) 3*, 3 (2012), 57.

[22] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, AUAI Press (2009), 452–461.

[23] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. Collaborative filtering recommender systems. In *The adaptive web*. Springer, 2007, 291–324.

[24] Scheffler, T., Schirru, R., and Lehmann, P. Matching points of interest from different social networking sites. In *KI 2012: Advances in Artificial Intelligence*. Springer, 2012, 245–248.

[25] Tiddi, I., d'Aquin, M., and Motta, E. Using linked data traversal to label academic communities. In *Proceedings of the 24th International Conference on World Wide Web Companion*, International World Wide Web Conferences Steering Committee (2015), 1029–1034.

[26] Van den Oord, A., Dieleman, S., and Schrauwen, B. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems* (2013), 2643–2651.

[27] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research 11* (2010), 3371–3408.

[28] Wang, C., and Blei, D. M. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM (2011), 448–456.

[29] Wang, H., Wang, N., and Yeung, D.-Y. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, ACM (New York, NY, USA, 2015), 1235–1244.

[30] Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., and Han, J. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, ACM (2014), 283–292.

[31] Zhou, K., and Zha, H. Learning binary codes for collaborative filtering. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM (2012), 498–506.