Bayes EMbedding (BEM): Refining Representation by Integrating Knowledge Graphs and Behavior-specific Networks

Yuting Ye
yeyt@berkeley.edu
Division of Biostatistics, University of
California, Berkeley, U.S.

Xuwu Wang
18110240012@fudan.edu.cn
School of Computer Science, Fudan
University, China

Jiangchao Yao
Kunyang Jia
jiangchao.jyc@alibaba-inc.com
kunyang.jky@alibaba-inc.com
Alibaba Group, China

Jingren Zhou
jingren.zhou@alibaba-inc.com
Alibaba Group, China

Yanghua Xiao
shawyh@alibaba-inc.com
School of Computer Science, Fudan
University, China

Hongxia Yang
yang.yhx@alibaba-inc.com
Alibaba Group, China

ABSTRACT

Low-dimensional embeddings of knowledge graphs and behavior graphs have proved remarkably powerful in varieties of tasks, from predicting unobserved edges between entities to content recommendation. The two types of graphs can contain distinct and complementary information for the same entities/nodes. However, previous works focus either on knowledge graph embedding or behavior graph embedding while few works consider both in a unified way.

Here we present *BEM*, a Bayesian framework that incorporates the information from knowledge graphs and behavior graphs. To be more specific, *BEM* takes as prior the pre-trained embeddings from the knowledge graph, and integrates them with the pre-trained embeddings from the behavior graphs via a Bayesian generative model. *BEM* is able to mutually refine the embeddings from both sides while preserving their own topological structures. To show the superiority of our method, we conduct a range of experiments on three benchmark datasets: node classification, link prediction, triplet classification on two small datasets related to Freebase, and item recommendation on a large-scale e-commerce dataset.

CCS CONCEPTS

• **Computing methodologies** → **Bayesian network models**;
Learning in probabilistic graphical models; Probabilistic reasoning;

KEYWORDS

Knowledge Graph, Bayesian Model, Graph Embedding

ACM Reference Format:

Yuting Ye, Xuwu Wang, Jiangchao Yao, Kunyang Jia, Jingren Zhou, Yanghua Xiao, and Hongxia Yang. 2019. Bayes EMbedding (BEM): Refining Representation by Integrating Knowledge Graphs and Behavior-specific Networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](https://permissions.acm.org).

CIKM '19, November 3–7, 2019, Beijing, China

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6976-3/19/11...\$15.00

<https://doi.org/10.1145/3357384.3358014>

In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19)*, November 3–7, 2019, Beijing, China. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3357384.3358014>

1 INTRODUCTION

Graphs widely exist in the real world, including social networks [13, 21], physical systems [1, 33], protein-protein interaction networks [9], knowledge graphs [12] and many other areas [18]. There may be different views of the same set of nodes, and thus graphs of different architectures are built. For example, in the e-commerce industry, item-item networks can be constructed based on the user behaviors of clicks, purchases, add-to-preferences and add-to-carts respectively — two items are linked if they are clicked (or via other operations) by the same user. A corresponding knowledge graph can be crafted to represent a collection of interlinked descriptions of the items, e.g., color, materials, functions. Throughout this article, we refer to the graph with respect to a certain behavior/context as the behavior graph (BG)¹, in order to distinguish it from the knowledge graph (KG) that consists of structured symbolic knowledge (triplets). KG and BG both reflect the interactions between entities/nodes in reality, but they differ in two aspects: 1) the graph structures; 2) the contained information; see Section 2 for a detailed discussion. The connection and the distinction between KG and BG imply that they can be complementary to each other. It is of great interest to integrate these two types of graphs in a unified way.

Benefits of the integration of KG and BG. In the sequel, we give three perspectives with examples in the e-commerce industry to illustrate the benefits of incorporating KG and BG. First, KG-aided BG can achieve accurate recommendations. For instance, given a formal dress and high-heel shoes, methods based on BG alone may recommend arbitrary lipsticks. With information from the KG, it can make a better recommendation of formal lipsticks instead of sweet lipsticks, as KG has the knowledge that the dress and the shoes are associated with formal occasions. Second, KG-aided BG can do more than BG alone. Suppose a user buys a ticket to Alaska in January, the knowledge “enjoying aurora in Alaska in winter” is triggered in KG. So it can recommend down jacket, outdoor shoes

¹The concept of BG covers a wide range of conventional graphs and networks: pagelink networks (the link behavior), author-citation networks (the citation behavior), item-item interaction behavior (the co-click, co-purchase behaviors and etc.), to name a few.

and tripods for the aurora viewing in a freezing environment. But methods using BG-only embeddings can hardly connect the flight ticket to such outfits. Third, novel knowledge can be discovered from BG on top of the known. For example, recent clothing fashions can be inferred by the frequently co-clicked or co-purchased clothes. Then humans' common sense or other experts' knowledge can be used to identify the most likely choice of the fashion of this year.

Motivation. To deal with multiple graphs, a standard practice is to embed the nodes as vectors while simultaneously integrating the information from all the sources [22, 44, 45]. To the best of our knowledge, however, there is no existing method that jointly learns the BG embedding and the KG embedding. As an alternative solution, it is common to take the pre-trained KG/BG embeddings as the input to learn the representation of BG/KG [13, 40, 41]. Or one can simply learn the embeddings of KG and BG separately, then incorporate them via an aggregation method, e.g., concatenation, linear combination. For the first strategy, the interaction information contained in the KG/BG embedding can be distorted if it does not agree with that of BG/KG. For the second strategy, the topological structure from either side is either disguised (e.g., concatenating a short embedding with a long embedding) or destroyed (e.g., taking the average of two embeddings of the same length). In this article, we work with the pre-trained BG and KG embeddings as this strategy is widely applicable. Our goal is to integrate BG and KG without losing the topological information from both sides.

Contribution. Throughout this paper, we consider only one KG and one BG. We develop a Bayesian framework called *BEM* (Bayes EMbedding) that refines the KG and BG embeddings in an integrated fashion while preserving and revealing the topological information from the two sources. The key idea behind *BEM* is that the KG embedding, plus a behavior-specific bias correction term, acts as the prior information for the generation of the BG embedding; see Figure 1 (c). *BEM* aims to maximize the likelihood under this Bayesian generative model. Our contribution is twofold. From the perspective of modelling, *BEM* is proposed to bridge KG and BG seamlessly, with the consideration of their respective topological structures. As a framework, *BEM* is general and flexible in that it can take any pre-trained KG embeddings and any BG embeddings to mutually refine themselves.

The rest of the paper is organized as follows. In Section 2, we discuss the difference and connection between KG and BG. In Section 3, we review works that are related to our method. In Section 4, we present our method *BEM*. In the sequel, we demonstrate the utility of *BEM* in three application studies involving two small datasets related to Freebase and a large dataset in e-commerce (Section 5). We test the *BEM*-refined embeddings in varieties of downstream tasks. Finally, we conclude with a discussion of the *BEM* framework and highlight promising directions for future work in Section 6.

2 DISCUSSION OF KG AND BG

Here we discuss the difference and the connection between KG and BG to illustrate three points: 1) KG and BG are different and hard to jointly learn; 2) KG and BG contain complementary (distinct but related) information, and therefore it is promising to get better embeddings by integrating the two types of graphs; 3) KG and BG can be unified from two reasonable perspectives.

Difference between KG and BG. There are mainly two differences between KG and BG. First, KG encodes entities and their relations in the form of the triplet as $\langle h, r, t \rangle$, where h , t and r are the head entity, the tail entity and their relation. It corresponds to a directed and highly heterogeneous network. In comparison, BG is constructed based on the interplay between the nodes under certain task/behavior-specific contexts. It corresponds to an undirected network with limited number of edge types (homogeneous or less heterogeneous than KG). Figure 1 (a) shows the difference between KG and BG in terms of the network structure. The distinction in structure makes it difficult to put the two graphs in a single framework for embedding learning. Second, the triplets in KG are extracted from authentic knowledge and experience. Thus, KG is a semantic network reflecting relatively objective facts that can stand the test of time. As for BG, it embodies a time-varying and behavior-biased link between nodes, which we illustrate with two examples: 1) People may buy sunglasses and swimwear at the same time in summer, but they will barely purchase these two items in winter; 2) Two sorts of sunglasses can be viewed (the click behavior) for comparison but they are rarely bought (the purchase behavior) together. The difference in information between KG and BG indicates that they can complement each other.

Connection between KG and BG. Despite the distinction, KG and BG are also closely related, resembling the connection between humans' knowledge and experience. KG can be regarded as an abstracted graph that reflects the shared properties among multiple BGs. This bottom-up idea (from BGs to KG) implies that it is possible to acquire novel knowledge from all kinds of BGs. On the contrary, we can heuristically interpret the connection from top down, as shown in Figure 1 (c). KG contains the general information of items, e.g., the item properties (color, materials etc.), the category of the item, the concepts/scenarios² of the category. Then, the node of BG can be thought of as being generated by adjusting the associative entity in KG with a behavior-specific correction term. For instance, the cell-phone is conceptually a portable electronics (KG). It exhibits varieties of properties under different scenarios (BG), e.g., a communication tool when connecting to others, an entertainment platform when playing games, a working/studying tool when looking up information online. The top-down idea indicates that we can use KG information to help the learning of BG.

3 RELATED WORK

In this section, we review related work to our method. As to the best of our knowledge, there is no existing method that learns the BG embeddings and the KG embeddings jointly. We first introduce multi-view learning that is closest to this goal. Then we review alternative methods, followed by classic representation learning methods for conventional graphs and knowledge graphs.

3.1 Multi-view Embedding Learning

In real life, entities may have different feature subsets which is called multi-view data. For instance, in e-commerce, an item may be associated with different behavior data in different scenarios, such as the data of purchases, clicks, add-to-preferences and add-to-carts.

²Scenarios are manually crafted to include items that appear together frequently under certain conditions. For example, the sunglasses and the swimwear both belong to the scenario "summer-beach".

Figure 1: (i) Examples of KG and BG; (ii) The workflow of BEM: 1) Embed KG/BG; 2) Train BEM with the parameters of the generative model in (iii); 3) Feed the original embeddings and trained parameters into BEM for refining; 4) Refined (corrected) KG/BG embeddings. (iii) A top-down generative model (Equation (1)) that connects one KG and three BGs with different behaviors a, b, c. First, for each behavior, there exists a behavior-specific correction term δ that accounts for the associative bias. Then the refined KG embedding is projected into the BG space via a non-linear transformation function f_{ϕ} . Finally, the BG embeddings are sampled from a distribution p_{θ} given the projected KG embedding. The model is trained to find the optimal $(\delta^*, f_{\phi^*}, \theta^*)$ that maximizes the likelihood of observing the BG embeddings z 's given the KG embeddings w 's.

These multi-view data can be learned to get a uniform representation for one item. For this purpose, varieties of approaches have been proposed, including co-training, multiple kernel learning, and subspace learning [22, 44, 45]. In particular, many efforts have been made in multi-view network representation learning. Qu et al. [32] combines the embeddings of different network views linearly. Shi et al. [34] proposes two characteristics (preservation and collaboration), and gets node vectors by simultaneously modeling them. It is closely related to our work in the sense that it emphasizes the integration of different sources while preserving their own specialties. However, it only deals with homogeneous networks as other multi-view embedding learning methods. In contrast, our method is designed to combine BG with KG, which differ in the data structures and the contained information.

3.2 Alternative Ways to Integrate KG and BG

There are alternative approaches to integrate KG and BG. First, the standard practice is to embed one graph into vectors, then take the embeddings as the input of the learning for the other graph. For example, Wu et al. [40] embeds sequential texts, then takes them as node/entity attributes for knowledge graph learning. Xie et al. [41] learns knowledge graph embedding by using the embeddings of entity descriptions. Hamilton et al. [13] can take as input the pre-trained KG embeddings to learn BG embeddings as well. However, this line of works tends to focus on the targeted graph (the graph that uses the pre-trained embedding for learning), but the topological structures from the other graph (the graph that generates the pre-trained embedding) may be missing. Even though interaction information between nodes is contained in the pre-trained embeddings, it can be weakened or ignored if not agreeing with the topology from the targeted graph. Second, there is a even simpler strategy to integrate KG and BG, i.e., learning the embeddings of KG and BG separately, then incorporating them

via an aggregation method, e.g. concatenation, linear combination [13]. Nonetheless, the topological structures from both sides are disguised or destroyed by these aggregation methods. Our work falls in the second category, and is designed to solve the above issue: it preserves and reveals the topological information when integrating BG and KG.

3.3 Representation Learning for BG and KG

Here we review methods used to pre-train BG and KG embeddings. A line of works perform graph embedding based on graph spectrum [2, 36]. Some works use matrix factorization to get node embeddings [6, 42, 43]. Additionally, simple neural networks are used to generate embeddings by making the distribution of the node embeddings close to that obtained by the topological structure [31, 37]. Recently, some graph neural network based techniques are also proposed and widely applied [13, 21, 28, 38].

Since KG differs from BG due to the semantic links between entities, the above embedding methods are not applicable to KG. Many efforts have been made to embed the nodes in KG. As a seminal work, TransE [4] learns a low dimensional vector for every entity and relation in KGs. Later extensions include TransH [39], TransR [24] and STransE [27] for more flexibilities.

4 THE PROPOSED METHOD

4.1 Notation

We denote w and z as the KG embedding and the BG embedding with dimension d_w and d_z respectively. For a vector x , let d_x be the dimension of x , and let x_k be its k -th entry. We use $\odot$ for element-wise multiplication, i.e., for two vectors x and y with length d , $x \odot y = (x_1 y_1, \dots, x_d y_d)$. Denote $D_{KL}(p||q)$ as the Kullback-Leibler (KL) divergence between distributions p and q . Other detailed notations used throughout this section are summarized in Table 1.

Table 1: Notation Table.

Notation	Meaning
e	Entity.
$\mathbf{w}$	KG embedding.
$\mathbf{z}$	BG embedding.
$d_{\mathbf{w}}/d_{\mathbf{z}}$	Dimension of $\mathbf{w}/\mathbf{z}$.
$\boldsymbol{\delta}$	The behavior-specific correction term.
$\mathbf{f}_{\phi}(\cdot)$	The nonlinear transformation that projects the refined (corrected) KG embedding into the BG space.
$\mathbf{v}$	Projection of the KG embedding onto the behavior space by $\mathbf{f}_{\phi}$.
$\mathbf{g}(\cdot, \cdot)$	The edge function that characterizes the interaction between entities in the behavior space.
p_{η}	The distribution of $\boldsymbol{\delta}$.
p_{θ}	The distribution of $\mathbf{g}(\mathbf{z}_i, \mathbf{z}_j)$.
$\mathbf{h}_{\Psi}(\cdot, \cdot)$	The inference network.
$\boldsymbol{\tau}_i/\boldsymbol{\tau}_{i,j}$	all the latent variables for $e_i/(e_i, e_j)$.

4.2 The Generative Model

Section 2 sheds light on the bottom-up and top-down relations between KG and BG. KG is thought of as the abstract representation of an entity, and BG is its realization under certain context. We can view BG as a mix of KG and a context-specific factor (an adjustment term), but usually it only reflects some aspect of KG (i.e., a projection of the mix). Such insights motivate us to connect KG and BG in a generative model as follows.

Throughout this paper, we focus on the case where each entity has one KG embedding and one BG embedding. Mathematically, suppose there are N entities and each entity e_i has a KG embedding $\mathbf{w}_i$ and a BG embedding $\mathbf{z}_i$. As depicted in Figure 1, $\mathbf{w}_i$ and $\mathbf{z}_i$ act as priors and observations respectively. We use $\boldsymbol{\delta}_i$ to model the adjustment effects between $\mathbf{w}_i$ and $\mathbf{z}_i$. In other words, $\boldsymbol{\delta}_i$ acts as an residual to $\mathbf{z}_i$ so that $\boldsymbol{\delta}_i + \mathbf{w}_i$ is sufficient to determine the marginal distribution of $\mathbf{z}_i$ via a projection function $\mathbf{f}_{\phi}$. The projection not only reflects the fact that BG characterizes KG partially, but is also technically required to map $\boldsymbol{\delta}_i + \mathbf{w}_i$ into the BG space. To be more specific, we assume the joint distribution of $(\mathbf{z}_1, \dots, \mathbf{z}_N)$ hinges on the following three components:

- "Refined" KG embeddings $(\mathbf{w}_1 + \boldsymbol{\delta}_1, \dots, \mathbf{w}_N + \boldsymbol{\delta}_N)$, where $(\boldsymbol{\delta}_1, \dots, \boldsymbol{\delta}_N)$ are sampled from the behavior-specific distribution p_{η} ;
- The nonlinear transformation $\mathbf{f}_{\phi}$ that projects the refined KG embedding into the BG space;
- The distribution of BG embedding p_{θ} .

Then, write the generative model as

$$\begin{aligned} (\boldsymbol{\delta}_1, \dots, \boldsymbol{\delta}_N) &\sim p_{\eta}(\cdot) \\ \mathbf{v}_i &= \mathbf{f}_{\phi}(\mathbf{w}_i + \boldsymbol{\delta}_i) \\ (\mathbf{z}_1, \dots, \mathbf{z}_N) &\sim p_{\theta}(\cdot | \mathbf{v}_1, \dots, \mathbf{v}_N). \end{aligned} \quad (1)$$

Table 2: Abbreviations of models and embeddings

Abbreviation	Meaning
<i>BEM-P</i>	BEM with node interactions; Model (3).
<i>BEM-I</i>	BEM with full independence; Model (4).
<i>BEM-O</i>	Without applying BEM.
$\mathcal{G}$ -P, $\mathcal{G}$ -I ($\mathcal{G} \in \{\text{BG, KG}\}$)	The $\mathcal{G}$ embedding by <i>BEM-P</i> , <i>BEM-I</i> .
$\mathcal{G}$ -O ($\mathcal{G} \in \{\text{BG, KG}\}$)	The original $\mathcal{G}$ embedding (by <i>BEM-O</i>).
concat-X ($X \in \{\text{P, I, O}\}$)	The concatenation of KG-X and BG-X.

Our target is to optimize the following objective function:

$$\begin{aligned} &\max_{\eta, \mathbf{f}_{\phi}, \theta} \log \mathbb{P}(\mathbf{z}_1, \dots, \mathbf{z}_N | \mathbf{w}_1, \dots, \mathbf{w}_N) \\ &= \max_{\eta, \mathbf{f}_{\phi}, \theta} \log \int p_{\theta}(\mathbf{z}_1, \dots, \mathbf{z}_N | \mathbf{f}_{\phi}(\mathbf{w}_1 + \boldsymbol{\delta}_1), \dots, \mathbf{f}_{\phi}(\mathbf{w}_N + \boldsymbol{\delta}_N)) \cdot \\ &\quad p_{\eta}(\boldsymbol{\delta}_1, \dots, \boldsymbol{\delta}_N) d\boldsymbol{\delta}_1 \dots d\boldsymbol{\delta}_N. \end{aligned} \quad (2)$$

However, the objective function Equation (2) under Model (1) is generally intractable. For the sake of computational feasibility, assumptions are needed to simplify the model:

- To reduce the model complexity, we assume $\boldsymbol{\delta}_i$'s are identically independently distributed, i.e., $\boldsymbol{\delta}_i \stackrel{i.i.d.}{\sim} p_{\eta}(\cdot)$, where η is shared by all the entities.
- To retain the interaction information between entities, we come up with an edge function $\mathbf{g}(\cdot, \cdot)$ that characterizes the interplay between e_i and e_j . For example, $\mathbf{g}(\mathbf{z}_i, \mathbf{z}_j)$ can be the similarity or the vector difference between $\mathbf{z}_i$ and $\mathbf{z}_j$. Then, p_{θ} is assumed to be a generative distribution for $\mathbf{g}(\mathbf{z}_i, \mathbf{z}_j)$.
- To further reduce the model complexity, we assume $\mathbf{g}(\mathbf{z}_i, \mathbf{z}_j)$'s are i.i.d sampled from $p_{\theta}(\cdot | \mathbf{g}(\mathbf{v}_i, \mathbf{v}_j))$, where θ is shared for all pairs of (e_i, e_j) .

Then, Model (1) is reduced to

$$\begin{aligned} \boldsymbol{\delta}_i &\sim p_{\eta}(\cdot), \boldsymbol{\delta}_j \sim p_{\eta}(\cdot) \\ \mathbf{v}_i &= \mathbf{f}_{\phi}(\mathbf{w}_i + \boldsymbol{\delta}_i), \mathbf{v}_j = \mathbf{f}_{\phi}(\mathbf{w}_j + \boldsymbol{\delta}_j) \\ \mathbf{g}(\mathbf{z}_i, \mathbf{z}_j) &\sim p_{\theta}(\cdot | \mathbf{g}(\mathbf{v}_i, \mathbf{v}_j)), \end{aligned} \quad (3)$$

which is visualized as Figure 2 (a). Compared to Model (1), the reduced model has a much smaller model complexity while retaining the interaction information between entities, i.e., preserving the topological structure, which is crucial for all the BG and KG embedding methods [5, 7]. We call this model **BEM-P** ("P" denotes pairwise interactions). In comparison, we can ignore the interactions for further complexity reduction:

$$\begin{aligned} \boldsymbol{\delta}_i &\sim p_{\eta}(\cdot) \\ \mathbf{v}_i &= \mathbf{f}_{\phi}(\mathbf{w}_i + \boldsymbol{\delta}_i) \\ \mathbf{z}_i &\sim p_{\theta}(\cdot | \mathbf{v}_i). \end{aligned} \quad (4)$$

In fact, Model (4) is a special case of Model (3) by letting $\mathbf{g}(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$ and assuming $p_{\theta}(\cdot | \mathbf{x}, \mathbf{y}) = p_{\theta}(\cdot | \mathbf{x})p_{\theta}(\cdot | \mathbf{y})$. Then it becomes a model with full independence. We call this model **BEM-I** ("I" denotes vertex independence). Finally, for the sake of simplicity, we denote **BEM-O** ("O" denotes NULL) as using the original embeddings directly without applying *BEM*. All these models are summarized in Table 2. In the sequel, we will omit the subscript η, ϕ and θ for simplicity if it does not bring about ambiguity.

4.3 The Inference Model

Given Equation (3), the objective function (2) can be rewritten as

$$\max \sum_{(i,j): i \neq j} \log \mathbb{P}(g(z_i, z_j) | (w_i, w_j)). \quad (5)$$

There are varieties of off-the-shelf methods to optimize Equation (5), such as the EM [26] or MCMC [10] algorithm. But these methods usually fail due to intractability of scalability. To this end, we resort to variational inference [3], which is very popular for large-scale scenarios or distributions with intractable integrals. Let τ_i be a set of all the latent variables for node i , and $\tau_{ij} = \tau_i \cup \tau_j$. For example, in the generative model Equation (4), $\tau_i = \{\delta_i\}$, $\tau_j = \{\delta_j\}$ and $\tau_{ij} = \{\delta_i, \delta_j\}$. It is easy to derive that

$$\begin{aligned} & \log \mathbb{P}(g(z_i, z_j) | (w_i, w_j)) \\ & \geq \mathbb{E}_{q(\tau_{i,j} | z_i, z_j, w_i, w_j)} \log \mathbb{P}(g(z_i, z_j) | \tau_{i,j}, w_i, w_j) \\ & \quad - D_{KL}(q(\tau_{i,j} | z_i, z_j, w_i, w_j) || p(\tau_{i,j} | w_i, w_j)), \end{aligned} \quad (6)$$

where $q(\tau_{i,j} | z_i, z_j, w_i, w_j)$ is called the *inference model* [20], i.e., an approximated density function to the posterior density of $\tau_{i,j}$ given (z_i, z_j) . $p(\tau_{i,j} | w_i, w_j)$ is the associated prior density. Formula (6) is also called the variational lower bound or evidence lower bound (ELBO) [15] for $\log \mathbb{P}(g(z_i, z_j) | w_i, w_j)$. The first term in the ELBO is termed as the reconstruction term that measures the goodness of the fit, while the second one is a penalty term that measures the distance between the approximated density to the prior density. Then, our goal of maximizing $\log \mathbb{P}(g(z_i, z_j) | w_i, w_j)$ can be relaxed to maximizing the ELBO. It is well-known that the naive Monte-Carlo gradient estimator exhibits very high variance and is impractical when N is large [30]. Thus we will utilize particular distributions and introduce additional assumptions to further simplify the ELBO.

We assume $p_\eta(\cdot)$ to be a multivariate normal density. Assume $p_\theta(\cdot | g(v_i, v_j)) = p_{\theta_{ij}}(\cdot | g(v_i, v_j))$ to be a multivariate normal density with mean $g(v_i, v_j)$ and variance matrix $\text{diag}(\theta_{ij})$, where $\theta_{ij} = s_i + s_j$ is the sample-specific variance (see Figure 2 (a)). Here, s_i and s_j are assumed to be sampled from a multivariate log-normal distribution. We introduce the latent variable s_i and s_j to account for the nuisance variation induced by sampling (see Section 4.4). Here we choose the multivariate normal/log-normal distribution because it enjoys appealing statistical and computational properties: 1) normal/log-normal random variables are easy to sample; 2) normal/log-normal distributions can be easily reparametrized with only two parameters [20]; 3) There is a closed-form expression for the KL divergence between two normal/log-normal distributions.

By introducing the latent variable s_i , the set of latent variables for node j becomes $\tau_i = \{\delta_i, s_i\}$ and $\tau_{ij} = \{\delta_i, \delta_j, s_i, s_j\}$. We then impose two common conditions in the mean-field variational inference [20]:

- Both $q(\tau_{ij} | z_i, z_j, w_i, w_j)$ and $p(\tau_{ij} | w_i, w_j)$ are from mean-field family. That is

$$q(\tau_{ij} | z_i, z_j, w_i, w_j) = q(\delta_i | w_i, z_i) q(\delta_j | w_j, z_j) q(s_i | w_i, z_i) q(s_j | w_j, z_j) \\ p(\tau_{ij} | w_i, w_j) = p(\delta_i | w_i) p(\delta_j | w_j) p(s_i | w_i) p(s_j | w_j).$$

- $q(\delta_i | z_i, w_i)$ and $q(s_i | z_i, w_i)$ are normal and log-normal densities with a diagonal covariance matrix, respectively.

Thus, the approximated posterior means and variances of each element in τ_i can be represented by a function of z_i and w_i , denoted

Figure 2: The BEM-P method with the normal/log-normal distributions with the sample-specific variance $\theta_{ij} = s_i + s_j$ for $g(z_i, z_j)$. (a) The generative model (3). The shaded circles represent observed/estimated variables. The empty circles represent latent variables. Edges signify conditional dependency (including deterministic mapping). The solid rectangles (“plates”) indicate independent replication while the dashed rectangles indicate replication only. (b) The inference network indexed by ψ . It takes in KG-O and BG-O and outputs the posterior means/variances of the latent variables in (a). (c) The computational pipeline that concatenates the inference model (b) and the generative model (a) to produce refined (corrected) KG/BG embeddings. (d) The illustration graph that explains the translation edge function is equivalent to the similarity function using inner product or cosine similarity on the sphere.

as h_Ψ , which is called the *inference network*. In detail,

$$h_\Psi(z_i, w_i) = (\hat{\mu}_{\delta_i}, \hat{\sigma}_{\delta_i}, \hat{\mu}_{s_i}, \hat{\sigma}_{s_i}), \quad (7)$$

where $\hat{\mu}_{\delta_i}, \hat{\sigma}_{\delta_i}, \hat{\mu}_{s_i}, \hat{\sigma}_{s_i}$ are the approximated posterior means and variances (a vector consisting of the diagonal elements of the covariance matrix) of δ_i and s_i respectively. With the reparametrization trick, we can express $x = \hat{\mu}_x + \hat{\sigma}_x \odot \epsilon$, $x \in \tau_i$ and $\epsilon \sim N(0, I_{d_x})$. Correspondingly, we express their prior means and variances as $\mu_{\delta_i}, \lambda_{\delta_i} \cdot \sigma_{\delta_i}^2, \mu_{s_i}, \lambda_{s_i} \cdot \sigma_{s_i}^2$, where λ_{δ_i} and λ_{s_i} are two tuning parameters. Then the ELBO in Equation (6) can be explicitly expressed. The reconstruction term is

$$\begin{aligned} & \mathbb{E}_{q(\tau_{i,j} | z_i, z_j, w_i, w_j)} \log \mathbb{P}(g(z_i, z_j) | \tau_{i,j}, w_i, w_j) \\ & = -\mathbb{E}_{\epsilon_{\delta_i}, \epsilon_{\delta_j}, \epsilon_{s_i}, \epsilon_{s_j}} \sum_{k=1}^{d_z} \left\{ \frac{1}{2} \log(s_{i,k} + s_{j,k}) \right. \\ & \quad \left. + \frac{[g(z_i, z_i)_k - g(f_\phi(w_i + \delta_i), f_\phi(w_j + \delta_j))_k]^2}{2(s_{i,k} + s_{j,k})} \right\} + C_0, \end{aligned} \quad (8)$$

where C_0 is a constant and

$$\begin{aligned}\delta_i &= \hat{\mu}_{\delta_i} + \hat{\sigma}_{\delta_i} \odot \epsilon_{\delta_i}, \delta_j = \hat{\mu}_{\delta_j} + \hat{\sigma}_{\delta_j} \odot \epsilon_{\delta_j}, \\ s_i &= \hat{\mu}_{s_i} + \hat{\sigma}_{s_i} \odot \epsilon_{s_i}, s_j = \hat{\mu}_{s_j} + \hat{\sigma}_{s_j} \odot \epsilon_{s_j}.\end{aligned}\quad (9)$$

The penalty term is

$$\begin{aligned}D_{KL}(q(\tau_{i,j}|\mathbf{z}_i, \mathbf{z}_j, \mathbf{w}_i, \mathbf{w}_j) || p(\tau_{i,j}|\mathbf{w}_i, \mathbf{w}_j)) \\ = \sum_{\mathbf{x} \in \tau_{ij}} \sum_{k=1}^{d_x} \left\{ -\log \frac{\hat{\sigma}_{\mathbf{x},k}^2}{\lambda_{\mathbf{x}} \cdot \sigma_{\mathbf{x},k}^2} + \frac{\hat{\sigma}_{\mathbf{x},k}^2}{\lambda_{\mathbf{x}} \cdot \sigma_{\mathbf{x},k}^2} + \frac{(\hat{\mu}_{\mathbf{x},k} - \mu_{\mathbf{x},k})^2}{\lambda_{\mathbf{x}} \cdot \sigma_{\mathbf{x},k}^2} \right\} + C_1,\end{aligned}\quad (10)$$

where C_1 is a constant.

We can draw several implications from the closed-form expression of the ELBO. Maximizing the ELBO in Equation(6) is equivalent to minimizing the sum of Equation(8) and Equation(10), which are balanced by λ_{δ_i} and λ_{s_i} . Minimizing the reconstruction term forces the corrected KG/BG embeddings to behave similarly to the observed BG embeddings as per the selected edge function $\mathbf{g}$. It suggests that the reconstruction term preserves the topological structure of BG. Accordingly, minimizing Equation (10) enforces the approximated posterior mean/variance to be close to the prior mean/variance. If the prior mean of δ_i is set to be $\mathbf{0}$, such minimization forces the corrected KG/BG embeddings to be close to the observed KG embeddings. It indicates the penalty term preserves the topological structure of KG. Thus, the refined KG/BG embeddings can be regarded as a mixture of information. The two parameters λ_{δ_i} and λ_{s_i} act as controllers of such mixing. For example, a small λ_{δ_i} indicates the corrected embeddings squint towards the observed KG embeddings other than the observed BG embeddings, vice versa.

4.4 Algorithm

Given all the components discussed above, we can write down the detailed algorithm of *BEM*. First, we sample two batches of samples of batch size n_B , denoted as batch B_a and B_b ; then pair them up randomly, denoted as $B_{pair} = \{(a_m, b_m)\}_{m=1}^{n_B}$. For each batch, we impose the same prior information for all the samples in this batch, and estimate

$$\begin{aligned}\mu_{\delta}^{(l)} &= \mathbf{0}, l = a, b \\ \sigma_{\delta,k}^{(l)} &= \frac{1}{n_B-1} \sum_{m=1}^{n_B} (\mathbf{w}_{l,m,k} - \sum_{m'=1}^{n_B} \frac{\mathbf{w}_{l,m',k}}{n_B})^2, k = 1, \dots, d_w, l = a, b \\ \mu_{s,k}^{(a,b)} &= \frac{1}{n_B} \sum_{m=1}^{n_B} (\mathbf{g}(\mathbf{z}_{a_m}, \mathbf{z}_{b_m})_k - \bar{\mathbf{g}}_k^{(a,b)})^2, k = 1, \dots, d_g \\ \sigma_{s,k}^{(a,b)} &= \frac{1}{R} \sum_{r=1}^R (\mu_{s,k}^{(a,b,r)} - \bar{\mu}_{s,k}^{(a,b)})^2, k = 1, \dots, d_g \text{ (bootstrap)}\end{aligned}\quad (11)$$

where $\bar{\mathbf{g}}^{(a,b)} = \frac{1}{n_B} \sum_{m'=1}^{n_B} \mathbf{g}(\mathbf{z}_{a_{m'}}, \mathbf{z}_{b_{m'}})$, R is the number of bootstrap replicates, $\mu_s^{(a,b,r)}$ is the r -th bootstrap estimator of $\mu_s^{(a,b)}$ from B_{pair} ($r = 1, \dots, R$), and $\bar{\mu}_s^{(a,b)} = \frac{1}{R} \sum_{r=1}^R \mu_s^{(a,b,r)}$. Then, for each pair of sample (a_m, b_m) , use the inference network Equation (7) to get the approximated posterior information $\hat{\mu}_{\delta_{l_m}}, \hat{\sigma}_{\delta_{l_m}}, \hat{\mu}_{s_{l_m}}, \hat{\sigma}_{s_{l_m}}, l = a, b, m = 1, \dots, n_B$, as shown in Figure 2 (b). Next, we sample $2n_B \cdot (d_w + d_z)$ standard normal variables to get δ_{l_m} and s_{l_m} by Equation (9), where we set $\lambda_{\delta_i} \equiv \lambda_1$ and $\lambda_{s_i} \equiv \lambda_2, i = 1, \dots, N$. We obtain the ELBO in Equation (6) via Equations (8)-(10), as shown in Figure 2 (c). Finally, We can use any optimization method, such as Adam [19], to update ϕ and ψ when maximizing the ELBO. We

run the above steps for T times, and we can get the refined KG/BG embedding for e_i by

$$\hat{\mathbf{w}}_i = \mathbf{w}_i + \hat{\mu}_{\delta_i}, \hat{\mathbf{z}}_i = \mathbf{f}_{\phi}(\hat{\mathbf{w}}_i) \quad (12)$$

Algorithm 1 The *BEM* method.

Input: Pre-Trained KG/BG embeddings $(\mathbf{w}_i, \mathbf{z}_i), i = 1, \dots, N$; tuning parameters λ_1, λ_2 ; batch size n_B , number of iterations T .

- 1: **for** $t = 1, 2, \dots, T$ **do**
- 2: Sample two batches B_a, B_b of batch size n_B , and pair them up as $B_{pair} = \{(a_1, b_1), \dots, (a_{n_B}, b_{n_B})\}$;
- 3: Estimate the prior information by $(\mu_{\delta}^{(l)}, \sigma_{\delta}^{(l)}, \mu_s^{(a,b)}, \sigma_s^{(a,b)})$ by Equation(11);
- 4: **for** $l = a, b; m = 1, \dots, n_B$ **do**
- 5: Get the posterior information $(\hat{\mu}_{\delta_{l_m}}, \hat{\sigma}_{\delta_{l_m}}, \hat{\mu}_{s_{l_m}}, \hat{\sigma}_{s_{l_m}})$ by Equation(7);
- 6: Sample a standard normal variable from $N(\mathbf{0}, \mathbf{I}_{d_w})$ and $N(\mathbf{0}, \mathbf{I}_{d_z})$ respectively. Get δ_{l_m} and s_{l_m} via Equation(9).
- 7: Obtain the ELBO in Equation(6) via Equations (8)-(10);
- 8: Update ψ and ϕ by maximizing the ELBO.
- 9: **for** $i = 1, \dots, N$ **do**
- 10: Get the refined KG/BG embeddings $\hat{\mathbf{w}}_i$ and $\hat{\mathbf{z}}_i$ by Equation(12).
- 11: Denote the ϕ and ψ in the last round as $\hat{\phi}$ and $\hat{\psi}$.

Output: $\hat{\phi}, \hat{\psi}, (\hat{\mathbf{w}}_i, \hat{\mathbf{z}}_i), i = 1, \dots, N$.

To analyze the complexity of Algorithm 1, we simply use two-layer MLPs (multi-layer perceptron) for $\mathbf{f}_{\phi}$ and $\mathbf{h}_{\psi}$. Let n_h be the number of hidden nodes of these neural networks. Then it is easy to see that the computational complexity is $O((n_z + n_w)n_h \cdot (n_{iter} + R) \cdot n_B \cdot T)$, where n_{iter} is the number of iterations for the maximization step (Line 8) in Algorithm 1. If we set $T \propto N/n_B$, the computational complexity is $O((n_z + n_w)n_h \cdot (n_{iter} + R) \cdot N)$. Furthermore, the storage complexity is just $O(n_z n_h + n_w n_h)$, since it merely needs to keep track of two sets of parameters in $\mathbf{f}_{\phi}$ and $\mathbf{h}_{\psi}$. Therefore, the algorithm is efficient in both time and storage in the sense that the size of the dataset only affects the computational time linearly. However, when the dataset is too large to be entirely loaded into the CPU, the algorithm might suffer from a non-negligible overhead caused by partitioning and loading the data during the iteration.

4.5 Edge Function

The edge function $\mathbf{g}$ in Equation (3) characterizes the interplay between nodes. The choice of this function determines what kind of KG information is incorporated into the BG embeddings. We give four examples as below:

- A natural choice is the translation function i.e., $\mathbf{g}(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \mathbf{y}$, where $d_g = d_w$ or d_z . TransE and its variants are based on the translation operation, and aim to minimize the L_2/L_1 loss between the translated embedding of the head entity and the corresponding embedding of the tail entity [4, 24, 39].
- An arbitrary similarity function can be used that measures the similarity between $\mathbf{z}_i$ and $\mathbf{z}_j$, where $d_g = 1$. Such choice coincides with the objective functions of the majority of BG/KG embedding methods [5, 7]. For instance, GraphSAGE

[13], GCN [21], node2vec [11] etc., maximize the inner product between positive samples while minimizing this metric between negative samples.

- If the edge function only relies on the index i and j , such as the edge attribute between node i and node j , *BEM* becomes a supervised model.
- If the edge function is an identity function $g(\mathbf{x}, \mathbf{y}) = (\mathbf{x}, \mathbf{y})$, then Model (3) is reduced to Model (4). Here, $\mathbf{g}$ simply concatenates vectors $\mathbf{x}$ and $\mathbf{y}$, thus $d_g = 2d_w$ or $2d_z$.

In this article, we use the translation function $g(\mathbf{x}, \mathbf{y}) = \mathbf{x} - \mathbf{y}$. In fact, the translation function is equivalent to the similarity function using inner product or cosine similarity if the embeddings are normalized onto the unit sphere, such as embeddings generated by GraphSAGE, TransE and its variants. As shown in Figure 2 (d), the module ℓ of the difference between two points on the sphere is bijectively mapped to the angle α between the rays from the origin to the two points.

5 EXPERIMENTS

We empirically study and evaluate *BEM* on two small datasets and one large-scale dataset for a variety of tasks. Each dataset consists of one KG and one BG with pre-trained node embeddings. The goal of these experiments is to show that embeddings refined by *BEM* can outperform the original pre-trained embeddings on some tasks, while remaining the efficacy for most of the others:

- The **node classification** task (on two small datasets) studies if *BEM* can help refine the KG/BG embedding using the BG/KG embedding. It also investigates whether *BEM* can reveals useful information in KG and BG for the classification purpose (Section 5.1.2).
- The **link prediction** [4] and the **triplet classification** [35] (on two small datasets) investigate whether *BEM* can extract useful information from BG to refine the KG embedding.
- The **item recommendation** task (on the large dataset) studies whether the information in KG can enhance the performance of the BG embedding.

For the node classification task, we study the KG/BG and the concatenated embeddings that are refined by *BEM*. In contrast, we only consider the KG embedding for the link prediction task and the triplet classification task since the two tasks are designed for the KG embedding. We only consider the BG embedding for the item recommendation for the same reasoning.

We implement³ *BEM* as per Algorithm 1 based on *tensorflow*⁴. Throughout this section, we use the following default parameter setting:

- Functions $\mathbf{f}_\phi$ and $\mathbf{h}_\psi$ are implemented as two-layer MLPs with 500 hidden nodes and the ReLU [25] activation.
- The batch size n_B is 500, the optimization algorithm is Adam [19], the learning rate is 0.001, the number of training steps $T = N/n_B \cdot 20$.
- $\lambda_1 = 1.0$, $\lambda_2 = 1.0$.

A discussion on the selection of the above parameters is deferred to Section C of the supplementary materials³.

³The code and the supplementary materials can be found at https://github.com/Elric2718/Bayes_Embedding.

⁴<https://www.tensorflow.org/>

Table 3: The node classification accuracy (%) using the refined BG/KG embeddings by *BEM*. Here KG, BG and concat refer to the KG embedding, BG embedding and the concatenation of the KG and BG embeddings, respectively.

		FB15K237 + <i>pagelink</i>					
		node2vec			LINE		
	<i>BEM</i>	KG	BG	concat	KG	BG	concat
TransE	O	85.59	75.12	89.39	85.59	77.57	89.44
	I	85.51	82.56	85.97	86.35	85.44	87.05
	P	88.89	86.32	90.29	88.21	86.27	90.01
TransD	O	86.06	75.12	89.18	86.06	77.57	89.00
	I	83.73	78.86	84.16	86.58	85.10	86.69
	P	88.60	85.39	89.90	88.70	85.30	89.73
		FB15K237 + <i>desc</i>					
		doc2vec			sentence2vec		
	<i>BEM</i>	KG	BG	concat	KG	BG	concat
TransE	O	85.32	75.62	87.92	85.32	83.42	88.43
	I	86.19	81.50	86.41	87.61	85.18	88.07
	P	87.68	81.52	87.86	88.05	85.82	88.57
TransD	O	85.83	75.62	88.07	85.83	83.42	88.52
	I	86.75	81.44	86.85	87.96	84.97	88.07
	P	87.34	82.24	88.15	88.36	86.12	88.86

5.1 Two Small Datasets

The two small datasets have the same KG but differ in the BGs. The shared KG is FB15K237, which is reduced from FB15K to remove the reversal relations [8]. There are 14,541 entities, 237 relations, and 272,115 training triplets, 20,466 validation triplets, 17,535 testing triplets. The first dataset uses a *pagelink* network (denoted as *pagelink*) that records the linkages between the wikipedia pages of entities in FB15K237. It includes 14,071 nodes (a subset of the entities in FB15K237) and 1,065,412 links. The second dataset comes with a short paragraph description (denoted as *desc*) for each entity in FB15K237. Strictly speaking, the descriptions do not form a BG due to the lack of connection between descriptions. We regard them as an isolated graph to evaluate *BEM* under extreme conditions where BG does not contain any interplay information between nodes. See Section A of the supplementary materials for more details on the two datasets.

We use TransE [4] and TransD [16] from OpenKE⁵ to pre-train KG's embeddings. Both of them are trained for 500 epochs with dimension $d_w = 50$ and other parameters are taken as default. For the BGs, we use doc2vec [23] and sentence2vec [29] to pre-train *desc* BG embeddings, and node2vec [11] and LINE [37] to pre-train *pagelink* BG embeddings respectively. The dimension of the BG embedding is set to be $d_z = 100$. More details on the experiment and hyper-parameter setups are included in Section B of the supplementary materials.

5.1.1 Node classification. In the node classification task, there are 46 class labels. The embeddings are fed into a multi-label logistic regression model for training and prediction. Table 3 shows the results of *BEM*, from which we can draw three implications. First,

⁵<https://github.com/thunlp/OpenKE> [14]

we observe consistent improvements of *BEM-P* over *BEM-O* (the original embedding) through almost all settings (accuracies boosted by 2%-10% for KG and BG). It indicates that we can benefit from integrating information of the two sources. Second, if the classifier is sufficiently expressive, concat-O is expected to perform the best since there is no loss of information from the input. However, concat-P turns out to perform slightly better than concat-O in most cases. It suggests that *BEM-P* not only preserves the information for node classification, but also reveals signals. Third, as we expected, *BEM-P* outperforms *BEM-I* since the former accounts for the pairwise interactions that are crucial for the embedding learning of KG/BG. Finally, we point out that the concatenated embedding and the KG/BG embedding are not comparable. The concatenated embedding is longer than the *BEM*-refined embedding, so the classifier for the former has more parameters, thus more expressive. For a fair comparison, we study the projection of the concatenated embedding onto the BG/KG space, and the associative results are deferred to Section D of the supplementary materials.

5.1.2 Empirical analysis. To understand the property of the embeddings refined by *BEM-P*, we perform two empirical data analyses on the FB15K237-*pagelink* dataset. First, we compute the absolute cosine similarity for each pair of nodes using KG-O, KG-P, BG-O, BG-P respectively. From Figure 3, we observe that the KG-P and BG-P are distributed more extremely than KG-O and BG-O — there are more highly correlated and more uncorrelated node pairs for the former. It indicates that *BEM-P* enforces some nodes to group tightly while some others are distracted from each other. This result can also be concluded by the visualization of the embeddings using t-SNE (Figure 4). Second, we use the class labels for the node classification task to compute

$$r = \frac{\max_C \{\text{within-cluster-distance}(C)\}}{\min_{C, C'} \{\text{between-cluster-distance}(C, C')\}},$$

where C, C' are two classes, and

$$\begin{aligned} \text{within-cluster-distance}(C) &= \frac{1}{|C|} \sum_{\mathbf{x} \in C} \|\mathbf{x} - \bar{\mathbf{x}}\|_2, \quad \bar{\mathbf{x}} = \frac{1}{|C|} \sum_{\mathbf{x} \in C} \mathbf{x}, \\ \text{between-cluster-distance}(C, C') &= \min_{\mathbf{x} \in C, \mathbf{y}' \in C'} \|\mathbf{x} - \mathbf{y}'\|_2. \end{aligned}$$

This metric reflects the degree to which the topological structure of the embeddings aligns with the labels. We have $r(X_{KG-O}) = 0.3042$, $r(X_{KG-P}) = 0.2695$, $r(X_{BG-O}) = 0.3890$ and $r(X_{BG-P}) = 0.3764$, indicating that *BEM-P* enforces nodes in the same classes to get closer to each other while nodes across classes are pulled away. This result suggests that *BEM-P* is able to preserve and further reveal the topological structure for both KG and BG.

5.1.3 Link Prediction and Triplet Classification on the KG side. We evaluate *BEM* on the link prediction and the triplet classification tasks. Since *BEM* can only refine the entity embeddings, we retrain the relation embedding for another 500 epochs using *BEM*-refined KG embeddings and the original relation embeddings as the initial values. In Table 4, notice that the KG embeddings can also benefit from incorporating the BG information via the *BEM* refining. In contrast, the concat-O embeddings are much inferior. It validates that the concatenation does not fully expose the topological structure of KG while *BEM* can make good use of this information. Moreover, we observe the improvement mainly occurs for the *pagelink* dataset. For the *desc* dataset, the TransD embeddings get improved slightly

Figure 3: Distribution of the similarities between nodes. Here, 1000,000 node pairs are sampled randomly.

Figure 4: Visualization of the embeddings. Blue: *BEM-O*; Red: *BEM-P*.

while the TransE embeddings get worse after applying the *BEM* refining. Such observation can be explained as the *desc* dataset does not provide supplementary interaction information to the KG graph.

5.2 A Large-Scale Dataset

In this section, we apply *BEM* to the KG/BG embeddings generated from an Alibaba Taobao's large-scale dataset⁶, whose statistics are summarized in Table 5. Considering the computational efficiency, TransE is used to get the KG embeddings on a knowledge database established by Alibaba Taobao. As with the BG embeddings, we run GraphSAGE on a graph constructed in terms of users' behaviors, e.g., two items are connected if a certain number of customers bought them simultaneously over the past months. GraphSAGE is a representative work for graph neural network (GNN) and has achieved good performances for large datasets. The dimension of KG embedding and the dimension of the BG embedding are $d_w = 64$, $d_z = 128$ as the online setting of Alibaba Taobao. We take the recommendation task for evaluation. Specifically, each customer has a set of trigger items from his/her historical behaviors including clicks, purchases, add-to-preferences and add-to-carts. These trigger items are then used to retrieve (by FAISS [17]) more items based on the BG embeddings. We evaluate our method by counting the

⁶The details of the Alibaba Taobao's dataset are deferred to Section A of the supplementary materials.

Table 4: Results of Link prediction (LP) and Triplet classification (TC).

Metrics	Embedding	FB15K237 + <i>pagelink</i>				FB15K237 + <i>desc</i>			
		TransE		TransD		TransE		TransD	
		node2vec	LINE	node2vec	LINE	doc2vec	sentence2vec	doc2vec	sentence2vec
Hit@10 (%) in LP-Filtered	KG-O	43.14	43.14	43.86	43.86	43.14	43.14	43.86	43.86
	KG-I	42.25	43.00	44.31	44.56	41.86	42.05	42.31	44.58
	KG-P	43.66	43.52	44.72	44.67	41.99	42.21	44.26	44.47
	concat-O	36.99	37.47	38.32	38.45	40.17	40.07	40.79	37.83
Accuracy (%) in TC	KG-O	76.56	76.56	78.29	78.29	76.56	76.56	78.29	78.29
	KG-I	76.70	76.86	78.54	78.80	76.06	76.42	78.63	78.61
	KG-P	77.13	77.09	78.96	79.11	76.17	76.21	78.70	78.60
	concat-O	71.97	73.23	71.82	70.75	71.41	71.32	72.15	69.91

Table 5: Specifications for the large-scale dataset.

#ent.	#scenario	#category	#rel.	#train
17.37M	182K	8.96K	5.18K	60.65M
#item	#value	#user	#edge_click	#edge_purchase
9.14M	8.04M	482M	7,952M	144M

Table 6: Hit recall rates (%) for item recommendation based on customer-specific trigger items. The recommended items are retrieved by finding the closest items to the trigger items using KG embeddings by *BEM-P* and *BEM-O*.

Granularity	Hit @	click		buy	
		<i>BEM-O</i>	<i>BEM-P</i>	<i>BEM-O</i>	<i>BEM-P</i>
brand	10	15.97	16.14	24.87	25.10
	30	16.65	17.12	25.70	26.57
	50	17.26	17.90	26.39	27.33
category	10	27.46	27.40	27.85	27.91
	30	28.43	29.99	28.50	29.45
	50	29.58	32.88	29.26	31.47

number of retrieved items that will be actually bought/clicked by the user in the following days. Table 6 exhibits the hit recall rates of the BG-P and BG-O on the recommendation task.

We check whether the retrieved items are of the same brand/category as those actually bought/clicked items in the following days. Combining these two granularities, we observe that the hit recall rates for BG-P are boosted by 1%-3% compared to BG-O, which is quite significant considering there are over 9 million items. It validates that *BEM-P* is able to incorporate useful KG information into the BG embedding for the item recommendation purpose.

Finally, for each concept/scenario, we use TransE to predict its top 50 item categories based on KG-O and KG-P (see the detailed procedure as Section 5.1.3). The result shows that KG-P can find more related items for the given concepts, as shown in Table 7. It indicates that by incorporating the BG information via *BEM*, we can acquire novel knowledge that does not exist in the original KG.

6 CONCLUSION

In this paper, we introduce *BEM*, a Bayesian framework that can refine graph embeddings by integrating the information from the KG and BG sources. *BEM* has been evaluated on a variety of experiments. It is shown to be able to improve the embeddings on multiple tasks by leveraging the information from the other side. *BEM* can achieve superior or comparable performance with higher efficiency to the concatenation method (the baseline) for the node classification task, and can help in other tasks where the simple aggregation methods (e.g., concatenation) are not applicable. It is designed by bridging KG and BG via a Bayesian generative model, where the former is regarded as the prior while the latter is the observation.

Currently, only one BG is considered at a time in this work. In fact, *BEM* can be easily extended to deal with multiple BGs. The integration of more than one BGs may further refine the KG, as their behavior-specific biases can be mutually canceled out. Besides, for the time being, *BEM* works only for pre-trained KG/BG embeddings. It can be potentially extended so that the networks for the KG/BG embeddings are connected and jointly trained via this framework. In other words, *BEM* can act as an interface that connects any KG embedding method with any BG embedding method for the end-to-end training. This makes the learning of the BG embedding supervised by the KG information. In turn, the learning of the KG embedding can be supplemented with instantiated samples in BG.

REFERENCES

- [1] P. Battaglia, R. Pascanu, M. Lai, and D. J. Rezende. Interaction networks for learning about objects, relations and physics. In *Neural Information Processing Systems*, pages 4502–4510, 2016.
- [2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.
- [3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
- [5] Hongyun Cai, Vincent W Zheng, and Kevin Chang. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2018.
- [6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

Table 7: Examples in which *BEM* acquire novel knowledge that does not exist in the KG.

concept	predicted categories using original KG embedding	predicted categories using the KG embedding refined by <i>BEM</i> -P
neuter clothing	jacket, homewear	Quick-drying T-shirt, sports down jacket, toning pants, aerobics clothes, warm pants
sports training	None	Quick-drying T-shirt, sports down jacket, sports bottle, Yoga T-shirt, training shoes, aerobics clothes
household items	succulents, detergent, tissue box, health tea, kitchen knife, man's facial cleanser, scented candle, washing cup, yoga mat towel	washing machine cover, spray, fish tank cleaning equipment, pen container, digital piano, pillow interior, wood sofa, table, bath bucket, composite bed, tape, mosquito patch, storage rack, needle, maker, storage box, leather sofa, indoors shoes, cotton swab, laundry ball, coffee cup, desiccant, trash bag

- [7] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 2018.
- [8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [9] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur. Protein interface prediction using graph convolutional networks. In *Neural Information Processing Systems*, pages 6530–6539, 2017.
- [10] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.
- [11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.
- [12] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1802–1808, 2017.
- [13] William L. Hamilton, Zitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1025–1035, 2017.
- [14] Xu Han, Shulin Cao, Lv Xin, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. Openke: An open toolkit for knowledge embedding. In *Proceedings of EMNLP*, 2018.
- [15] Matthew D Hoffman and Matthew J Johnson. Elbo surgery: yet another way to carve up the variational evidence lower bound. In *Workshop in Advances in Approximate Bayesian Inference, NIPS*, 2016.
- [16] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 687–696, 2015.
- [17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.
- [18] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems 30*, pages 6348–6358, 2017.
- [19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [20] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [22] Marius Kloft and Gilles Blanchard. The local rademacher complexity of lp-norm multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 2438–2446, 2011.
- [23] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.
- [24] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15*, pages 2181–2187, 2015.
- [25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.
- [26] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.
- [27] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranx: a novel embedding model of entities and relationships in knowledge bases. In *HLT-NAACL*, pages 460–466. The Association for Computational Linguistics, 2016.
- [28] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *KDD*, 2016.
- [29] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.
- [30] John Paisley, David Blei, and Michael Jordan. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.
- [31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.
- [32] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. An attention-based collaboration framework for multi-view network representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1767–1776. ACM, 2017.
- [33] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *arXiv preprint*, page 1806.01242, 2018.
- [34] Yu Shi, Fangqiu Han, Xinwei He, Xinran He, Carl Yang, Jie Luo, and Jiawei Han. mvn2vec: Preservation and collaboration in multi-view network embedding. *arXiv preprint arXiv:1801.06597*, 2018.
- [35] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.
- [36] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826. ACM, 2009.
- [37] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.
- [38] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 1(2), 2017.
- [39] Z Wang, J Zhang, J Feng, and Z Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 985–991, 2014.
- [40] Jiawei Wu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Knowledge representation via joint learning of sequential text and knowledge graphs. *arXiv preprint arXiv:1609.07075*, 2016.
- [41] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *AAAI*, pages 2659–2665, 2016.
- [42] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.
- [43] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. Fast network embedding enhancement via high order proximity approximation. In *IJCAI*, pages 3894–3900, 2017.
- [44] Shipeng Yu, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao. Bayesian co-training. *Journal of Machine Learning Research*, 12(Sep):2649–2680, 2011.
- [45] Deming Zhai, Hong Chang, Shiguang Shan, Xilin Chen, and Wen Gao. Multiview metric learning with global consistency and local smoothness. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):53, 2012.