Bayesian Embedding — a generic framework for incorporating prior knowledge into graph embedding

Yuting Ye Xuwu Wang

ABSTRACT

This paper provides a Bayesian framework, called BEM (Bayes EMbedding) that bridges the knowledge graph and entity-specific information. This framework is highly flexible in that it can take as input either the raw data or pre-trained embeddings. On the one hand, BEM can mutually correct the embeddings from two sides, one from the knowledge graph and the other one from the entity information. The corrected embeddings are enriched with the information from the other side. On the other hand, BEM holds the promise of making an end-to-end embedding network that uses the knowledge graph to guide the downstream embedding learning. Extensive experiments have been done to display the efficacy of BEM.

1. INTRODUCTION

2. METHODS

2.1 Notation

2.2 Models of BEM

The motivation of BEM originates from the cognitive science. Human beings's understanding process works in two systems: (System I) unconsciously retrieves pieces of knowledge associated with the given task; (System II) subjectively use analytic reasoning to analyze or ponder the received information. In spirit, the triplets in the knowledge graph function and distribute similarly to the knowledge used in System I: they provide association rather than causlity; they are scattered around rather than organized. Therefore, we are motivated to design a model that mimics System II. The triplets in the knowledge graph are regarded as the base knowledge that will be further processed to solve a specific task. In other words, the embedding from the knowledge graph should be corrected in terms of the task to tackle.

We first start with the simplest case where each entity has two embeddings, one from the knowledge graph and the other given by a specific task. For an entity e, we suppose that the task-specific embedding $\mathbf{z}_e$ is generated from a distribution determined by the knowledge graph embedding $\mathbf{h}_e$ and a correction term with δ_e respect to the task, that is,

$$\mathbf{z}_e \approx \mathbf{f}(\mathbf{h}_e + \delta_e), \tag{1}$$

where $\mathbf{f}$ is a non-linear function that projects $\mathbf{h}_e+\delta_e$ as $\mathbf{z}_e$. However, both $\mathbf{z}_e$ and $\mathbf{h}_e$ are distributed on the sphere. The set of all $\mathbf{z}'_e s$ (or $\mathbf{h}'_e s$) is the same for the downstream goal such as classification or prediction, up to a variety of operations including rotation and reflection. A sufficiently complicated non-linear function $\mathbf{f}$ is required to express such operation. However, the number of parameters are obviously larger than the sample size, since each entity needs a correction term. Learning δ_e's and $\mathbf{f}$ seems infeasible. On the other hand, we note that the learnings of both the knowledge graph embeddings and the task-specific embeddings are highly dependent on the interaction between entities. Decoupling the second-order information between entities as (1) might lead to an intolerable loss regarding the goal for a good embedding. Considering the interaction not only retains the loss of key information, but also increases the sample sizes (there are N_e^2 pairs of entities). With these considerations, we formulate a generation model from $\mathbf{h}_e$ to $\mathbf{z}_e$:

1. For each entity e:
   - sample its correction variable $\delta_e \sim N(\mathbf{0}, \lambda_1 \cdot s_h^2)$, where λ_1 is a tuning parameter and s_h^2 is estimated by $\mathbf{h}_e$.
   - sample its assoiated uncertainty variable $\sigma_{e_j}^2 \sim \log \text{Normal}(\ell_{\mu_e}, \lambda_2 \cdot \ell_{\sigma_e^2})$, where $j = 1, \ldots, d_2$. Here, ℓ_{μ_e}, $\ell_{\sigma_e^2}$ can be estimated by $\mathbf{z}_e$'s, and λ_2 is a tuning parameter.
2. For each pair of entities, e_1, e_2, sample
$$\mathbf{z}_{e_1} - \mathbf{z}_{e_2} \sim N\left(\mathbf{f}_\phi(\mathbf{h}_{e_1} + \delta_{e_1}) - \mathbf{f}_\phi(\mathbf{h}_{e_2} + \delta_{e_2}), \text{diag}(\sigma_{e_1}^2 + \sigma_{e_2}^2)\right), \tag{2}$$
where $\mathbf{f}_\phi$ is a neural network with parameter ϕ.

The model is depicted in Figure 1 as well. Our goal is to seek the best parameter ϕ and the posteriors of δ_e's,

σ_{e_j}'s in (2), given all the $\mathbf{h}_e$'s and $\mathbf{z}_e$'s. Suppose the fitted network paramter is $\hat{\phi}$ and the posterior mean of δ_e is $\hat{\mu}_e$, we use $\mathbf{h}_e + \hat{\mu}_e$ as the corrected embedding for the knowledge graph, and $\mathbf{f}_{\hat{\phi}}(\mathbf{h}_e + \hat{\mu}_e)$ as the corrected embedding for the specific task.

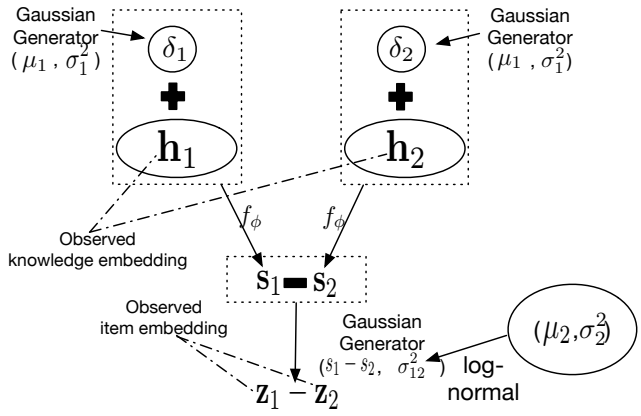

Figure 1: The BEM model.

2.3 Algorithm of BEM

3. EXPERIMENTS

3.1 Data description

We mainly use two datasets for evaluation. One dataset is a subset of E-commerce data from Alibaba on November 28th for the mother&baby scenario, termed as *Alibaba-MB*. In this dataset, we have 377, 183 items, each of which has 22 continuous attributes and 10 discrete attributes (details of the attributes are deferred to the appendix). All these attributes appear as the head entity in 5, 565, 558 triplets. There are 1, 931 relations (2, 000 properties for "HasProperty", "IsInstanceOf", "BelongToScenario") and tail entities (430 categories, 1, 511, 000 values, 111, 000 concepts). Another dataset is the public dataset *FBK15*. It has 14, 904 entities appearing as both the heads and tails in 579, 655 triplets. For each entity, we find an associated short paragraph in the database *freebase* that describes the entity.

3.2 Embedding correction

We first investigate the performance of BEM for embedding correction. For the dataset *Ali-MB*, the item attributes, along with item graph obtained by user behaviour, are used to get the item embeddings by *GraphSAGE*. For the dataset *FBK15*, the entity-specific description paragraphs are fed into *doc2vec* for embeddings. As with the knowledge graph, *TransE* (we utilize the implementation from the OpenKE library) is used to obtain the embeddings using the triplets. We call the embeddings from item attribute as "embedding(item)", the embeddings from the description paragraphs as "embedding(desc)", and those from knowledge graph as "embedding(kg)" respectively. Multiple tasks are studied to compare the embeddings before and after the correction by BEM.

3.2.1 Item Prediction for *Alibaba-MB*

There are two classification tasks, that is, to predict item categories of level 1 (L1) and level 2 (L2). We have 83 L1 categories and 580 L2 categories. The evluation measurement for this task is classification accuracy := #{correctly classified items}/#{total items}. In addition, we execute two regression tasks, that is, to predict item prices and item popularities. We use the squared ℓ_2 norm for evaluation. We note that the category information is used in the training of item embedding and knowledge graph embedding, while the item price and popularity are only used in the training of item embedding. We use a one-hidden-layer MLP with 32 hidden nodes for classification or regression.

Table 1 displays the results of the four tasks for item embeddings and kg embeddings before and after correction by BEM respectively. The extremely high accuracy of the two classification tasks results from the leaky information — the features partially include the predictors. Nevertheless, the knowledge graph and item attributes see different degrees of leak. It seems that the original embedding(item) is better informed of the categories than the original embedding(kg). After correction, the embedding(kg) always benefits from getting information from the other side in terms of classification accuracy, while the result of the embedding(item) gets worse for the level-2 category. It verifies the fundamental motivation of BEM that it merges two sides of embeddings. As with the two regression tasks, we observe similar results as the classification tasks. The item attributes include the price level but the knowledge graph does not, thus the original embedding(item) has a lower loss than the corrected one. The corrected embedding(kg) is obviously benefitted from learning the embedding(item). As for the task of predicting the item popularity, the predictors concentrate on low values but there are quite a few outliers. Therefore, the results display a high variance that masks the same trend as the task of predicting the price.

Table 1: Classification/Regression results for BEM on the *Alibaba-MB* dataset. Predictors category(L1/L2) correspond to using level1/level2 category for classification, while predictors price level/popularity correspond to using the item price or the item popularity for regression.

Predictor	embedding (item)		embedding (kg)	
	original	corrected	original	corrected
category(L1)	99.24%	99.31%	95.87%	98.99%
category(L2)	96.93%	94.06%	90.89%	96.86%
price level	0.5392	0.6935	0.9309	0.6644
popoularity	1852	1962	2008	1977

3.2.2 Link Prediction & Triplet Classification for *FBK15*

Two tasks are done to study the performance of BEM on the *FBK15* dataset, i.e., link prediction (LP) and triplet classification (TC). Both the two tasks are executed using the default APIs from the OpenKE library, given the embeddings from the knowledge graph.

The results for the two task with different parameters are presented in Table 2. The *default* setting only uses the *TransE* method without the BEM correction and with parameters #epochs = 500, #batches = 100, learning rate = 0.001, margin = 1.0, embedding dimension = 100, and etc.. The *noisy* setting only trains the relation embeddings while leaving the entity embeddings as initialized. The two *Tune-TransE* settings have exactly the same parameters as the *default* setting except for the ones they specify. As for the six settings in *Tune-BEM*, they utilize the BEM method to incorporate the information from the description paragraphs into the embedding(kg). The setting with learning rate = 0.001 is exactly the same parameters for *TransE* as the *default* one. Its parameters for BEM are $\lambda_1 = 1.0$, $\lambda_2 = 1.0$, #epochs = 20, #batches = 500, learning rate = 0.001. Other settings differs from the setting with learning rate = 0.001 only in the parameter they specify.

The results of corrected embedding(kg) are inferior to those of the original embeddings in the task of link prediction and triplet classification. This results from the fact that the description paragraphs do not contain sufficient information about knowledge graph triplets to benefit the other side. A direct implication is that the results of the corrected embedding(kg) get worse as more information from the description paragraphs is incorporated into it. On the other hand, we still get valuable insight by comparing the results between distinct parameters. λ_1 and λ_2 play the role of coefficients that adjust the trade-off between the reconstruction term and the penality term (KL divergence). Small λ_1 forces the corrected embedding(kg) to get close to the original one while small λ_2 forces any pair of the corrected embeddings(desc) to distribute as far as the original pair.

3.2.3 *Entity Classification for FBK15*

We investigate the performance the corrected embeddings(desc) and embeddings(kg) obtained from the *FBK15* dataset on another task. We note that the relations in this dataset not only contains the true relation between the head entity and the tail entity, but also include the domain and category words potentially associated with the entities. We retrieve the category labels for each entity from all the relations it is connected with in existing triplets. One entity can have multiple labels and we use the one of the highest occurence across all entities. We are aware that such label is not perfectly reliable due to the potentially leaky information during the training of the embedding(kg) and the naive selection of labels. Once the embeddings are obtained, the classification task is done via a one-hidden-layer MLP with 32 hidden nodes.

Tabel 3 displays the classification accuracy of the corrected embeddings in contrast to the orignal ones. We study the embedding(desc), the embedding(kg) and the embedding obtained by concatenating both of them. The original embedding(kg) can better distinguish the entity category, which implies that the training process sees partial information about the labels. We note that the embeddings by concatnating the original two types of embeddings perform better than single embeddings regardless of whether corrected or not. It is easily understood since the concatenation operation does not lose information from both sides in distinction to BEM. However, BEM obviously benefits the embeddings in that all the corrected ones are superior to the corresponding orignal ones. And the corrected embedding(kg) attains pretty close accuracy as the orinal concatenated embedding. It is not fully clear whether the gap is caused by the more expressive classifier for the concatenated embedding (its input size is double that of the single embedding). In fact, the single embedding and the concatnated embedding are not directly comparable. Still, we see that the BEM implicitly increases the singal-to-ratio even for the concatenated embeddings with respect to the task of classifying entity labels.

3.3 End-to-end learning by BEM

4. DISCUSSION

Table 2: Link prediction (LP)/Triplet classification (TC) results for BEM on the *FBK15* dataset. The result with one tuning parameter (except for tuning BEM) only differs in this parameter from the default. The result with one tuning parameter (for tuning BEM) only differs in this parameter from the setting of learning rate = 0.001.

task	metrics	default	noisy	Tune-TransE		Tune-BEM					
				(batch, epoch)	dim	learning rate		λ_1		λ_2	
				(800, 1000)	50	0.001	0.005	0.1	2	0.1	2
LP	MR	63	5693	69	66	82	69	65	93	79	73
	hit@10	71.83%	8.39%	76.11%	62.97%	60.26%	63.55%	67.24%	55.25%	59.00%	62.34%
	hit@5	54.41%	7.43%	59.08%	45.19%	41.45%	47.59%	48.84%	37.25%	40.61%	43.62%
	hit@1	32.60%	6.58%	35.56%	26.88%	21.49%	25.86%	26.65%	19.76%	21.59%	23.61%
TR	accuracy	85.99%	52.67%	87.48%	85.28%	82.65%	84.01%	85.50%	80.73%	82.27%	83.38%

Table 3: Classification results for BEM on the *FBK15* dataset. The abbreviation *desc* refers to embeddings from entity-specific description paragraphs, kg refers to the embeddings from the knowledge graph, concat refers to concatenting the desc embeddings nd the kg embeddings.

	embedding		
	desc	kg	concat
original	62.10	68.19	72.33
corrected	66.01	72.00	73.83