GCN-VAE for Knowledge Graph Completion

Karen Yang
kaiyuany@stanford.edu

Abstract

Knowledge graphs are powerful abstraction to represent relational facts among entities. Since most real world knowledge graphs are manually collected and largely incomplete, predicting missing links from given graph becomes crucial for knowledge graph to be more useful. Recently many studies have shown promising results by embedding entities and relation in a vector space. However, a common issue with most point estimates methods is that they could fail to capture more complex structures such as one-to-many, or many-to-many relations, nor can they adapt to scenarios where entities and relations have intrinsic uncertainties and multiple semantic meanings. To this end, we consider a probabilistic framework and propose GCN-VAE, a variantaional autoencoder to encode entities' neighborhood structure into a compact latent embedding distribution and reconstruct the links between entities by estimating edge likelihood via a decoder. We demonstrated the effectiveness of GCN-VAE on two popular benchmarks and got improved performance over state-of-the-art baselines on FB15k-237.

1 Introduction

Knowledge graphs (KG) is a powerful abstraction to represent relational data. It is useful in many applications like Q&A, data insights, search, medical ontology, where efficient relation extraction can be efficiently inferred from graph structured data. A knowledge graph consists of a collection of fact triplets (head entity, relation, tail entity), e.g. (Barak Obama, president_of, United States). It can also be transformed into a heterogeneous graph consisting of entities as nodes and relations as different types of edges.

Most of the real-world knowledge graphs are collected and created by humans, which is a long and manually process, with a lot of facts incomplete (missing entities or missing links between them). Thus many link prediction algorithms are interested in capturing the graph information in order to infer possible unknown facts from a given incomplete graph.

We found that most existing methods for link prediction task focuses on computing a point vector representation for each entity and relation within a knowledge graph. However, many of them fail to capturing symmetric, compositional, inverse relations. While some can cleverly design algorithm in complex space to tackle the above scenarios, such as RotateE [Sun et al., 2019] these methods could still fail in non-injective or many-to-many relations or settings where an entity or a relation has multiple semantic meanings that a fixed vector value would not be able to capture.

To this end, we devise a probabilistic framework for knowledge graph embedding method that aims to model the uncertainties of the entities and relations.

While recent successes of deep generative methods has achieved superior results on images, texts, and general graphs structured data, few has worked knowledge graphs domain. One of the prior studies that utilizes generative model is KG2E [He et al., 2015], which is the probabilistic version of TransE [Bordes et al., 2013] with a Gaussian model.

We would like to explore the representation power of a generative models on knowledge graphs further by designing a more complex generative model. In this paper, we introduce GCN-VAE, a variational autoencoder with relational graph convolutional network to encoder knowledge graph neighborhood into a non-Gaussian latent distribution, and a decoder to reconstruct the graph by predicting the likelihood for each possible triplets. Code is available on https://github.com/karenyang/GCN-VAE.

2 Related Work

TransE & variants[Bordes et al., 2013]. Introduced in lecture, TransE is one of the most commonly used benchmark for knowledge graph embedding. It represents head and tail entities h and t in a vector space, and treat relation r as a transnational distance, we can learn a model to fit $h + r \approx t$. Given its simplicity, TransE only assumes linear relation that can be embedded with one-dimensional vectors. This will be limiting in 1-N or N-1 multi-relation scenarios where entities embeddings could be close in one type of relation but should be far away in another relation type.

There are multiple Trans"X" variants that address above problem. One example is TransH [Wang et al., 2014] represent each r as normal vector w_r to re-project the entity vectors. $h_\perp = h - w_r^T h w_r, h_\perp = t - w_r^T t w_r$.

Different from translation based embedding, DistMult [Yang et al., 2014] uses a bi-linear modeling for the triple (h, r, t) by defining $f_r(h, t) = h^T diag(r)t$. However, this is symmetric embedding between head and tail, which is not expressive enough for the knowledge graph's intrinsic direct (asymmetric) nature.

Deep learning based embedding Recently, Neural network based KG embeddings achieved even better results on the Link predictions and node classification tasks. ConvE[Dettmers et al., 2018] uses a multi-layer convolutional network model, where the entities and relations are reshaped into 2D matrices and treat them as "images" to operate with multiple CNN layers. ConvE learns an expressive and parameter-efficient model that reaches SOTA on Link Prediction tasks at its time of publishing.

Motivated by the neighborhood aggregation concept in Graph Convolution Network (GCN), Schlichtkrull et al. proposed R-GCN[Schlichtkrull et al., 2018] to adapt GCN for KG's relational modeling. Instead of learning a adjacency matrix, R-GCN defines a head entity's neightborhood for each relation type r_i by collecting tail entities that fulills (h, r_i, t) with the h, as well as a self edge (include head entity's oen feature), and aggregate neighborhoods of each relation together by summing them together and apply non-linearity. Their performance on node classification is a good entity embedding for node classification tasks.

GraphVAE [Simonovsky and Komodakis, 2018] has made progress towards generating novel molecules and chemical compounds using variational autoencoders. They also extend a conditional version to condition on input label (in their case, a histogram of heavy atoms). They achieved good performace and rubustness on small molecules generation, however the model struggles to model and generate larger molecule's complex chemical interactions.

Graphite [Grover et al., 2018] applies unsupervised learning of node representation using variational autoencoders. They introduce a latent variable Z and model $p(Z|X)$ as a Gaussian prior over every entry of X. They applys GCN as forward message passing mechanism, after acquiring latent presentation Z, passing by constructing a as an inner product of latent matrix with itself $Z^T Z$, goes through symmetric levels of GCN layers to model the reverse message passing procedure and iteratively refine the generated graph.

GraphRNN [You et al., 2018] learns to generate sequences of graphs with a recurrent autoregressive model. The RNN model learns the distribution of distribution of observed graphs (represented by unweighted or weighted adjacency matrices), and is optimized for the maximum likelihood loss over the newly generated graph conditioned on previous ones. The RNN model allows it to generate arbitrarily large graphs , and the autoregressive features allow it to generate sequences of diverse graphs that learns the structural characteristics from previous samples.

3 Methods

3.1 Problem Formulation

3.1.1 Input

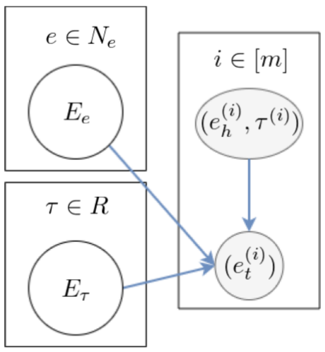

Figure 1: Graphical model for link prediction problem.

Suppose we are given a knowledge graph dataset in the form of a set of triplets $D = \{(e_h, \tau, e_t)\}$. e_h, e_t are the head entities and tail entities, and n represents the total number of entities in the dataset. τ is the edge or relation between them, and R represents the set of all relation types in the dataset.

We transform it in to a graph structure $G = (E, A)$ defined by

Entity attribute matrix: $E \in \mathbb{R}^{n \times d_e}$

Relation-specific Adjacency List: $A = \{A_\tau, \forall \tau \in R\}$

3.1.2 KG completion Task

The model is trained to learn a entity embedding and relation embedding given input graph $G = (E, A)$, and predict the likelihood of any given triplet to perform in Link Prediction tasks, as shown in Figure 1.

We only provide 50% of the edges from the input graph we sampled from the dataset to the model as input, since we want the model to learn from a subset of the graph structure and infer unseen edges.

The joint probability of a reconstructed graph can be expressed as:

$$p_\theta(G, E, A) = \prod_{h \in N_e, \tau \in R} p_\theta((h, \tau)) \prod_{(h,\tau,t) \in D} p_\theta((h, \tau, t)|G) \tag{1}$$

3.1.3 Training

Since there is an enormous amount of possible triplet ($O(|N_e| \times |R| \times |N_e|)$), directly doing inference on all of them is too computationally expensive. It also leads to an unbalanced training set with much more negative triplets than positive ones.

Therefore, we train our model via negative sampling, where we sample k negative samples per positive samples by corrupting either the head or tail entity of a correct triplet (but not both). The reconstruction loss can then be seen as the binary cross entropy loss on positive sample set D^+ and negative sample set D^-.

The model's objective is to minimize the approximated negative ELBO (Evidence Lower Bound), defined as:

$$\begin{aligned} ELBO(\phi, \theta; G) = &\sum_{h \in N_e, \tau \in R, t \in N_e} \mathbb{E}_{e_h, e_t \sim q_\phi(z|G)}[\log p_\theta((e_h, \tau, e_t)|G)] - \mathcal{D}_{KL}[q_\phi(z|G)||p(z)] \\ \simeq &\sum_{(h,\tau,t) \in D^+} \mathbb{E}_{e_h, e_t \sim q_\phi(z|G)}[\log p_\theta((e_h, \tau, e_t)|G)] + \\ &\sum_{(h,\tau,t) \in D^-} \mathbb{E}_{e_h, e_t \sim q_\phi(z|G)}[\log p_\theta((e_h, \tau, e_t)|G)] - \mathcal{D}_{KL}[q_\phi(z|G)||p(z)] \end{aligned} \tag{2}$$

The first and second term is the reconstruction loss of the sampled input graph . The third term is the KL divergence between the variational conditional distribution and the true conditional distribution. To avoid overfitting, we also add regularization terms.

3.2 Preprocessing

Here we explain the processing steps to transform triplets into a local graph neighborhood of a given entity as input representation G defined in 3.1.

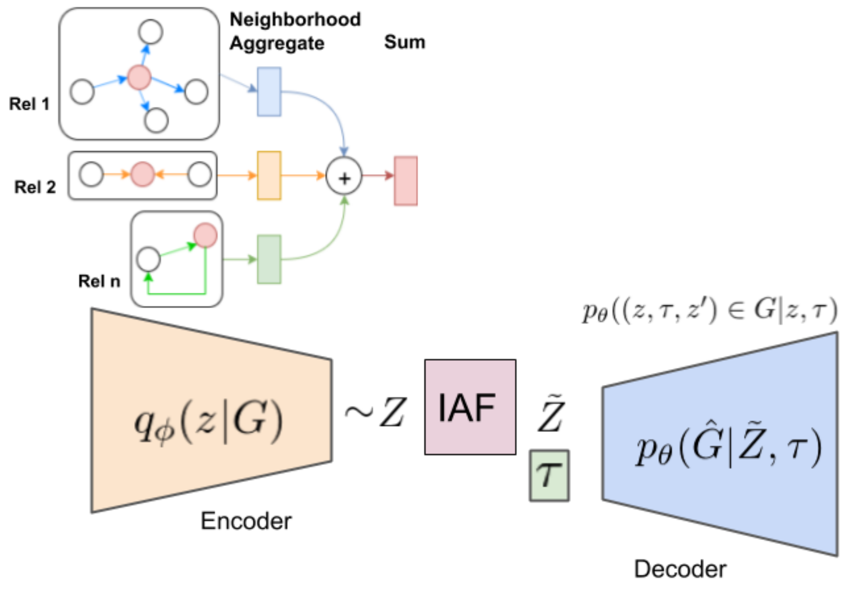

Figure 2: Model Architecture for GCN-VAE. Given an input graph defined on local neighborhood $G = (E, A)$, the encoder uses graph convolutional neural network to aggregate an entity's neighborhood, and outputs the mean and variance of latent embedding distribution. It then samples a latent code z from the distribution. In one of our proposed models that utilizes IAF (inverse autoregressive flow), the latent codes Z first goes through 3 layers of IAF transform to form a more complex non-Gaussian posterior. Lastly, conditioned on relations, the decoder takes in sampled latent codes and relation embedding, and outputs likelihood scores for possible triplets.

We sampled local neighborhoods as subgraphs in order to handle large-size datasets such as FB15k-237. We collected a local neighborhood of size n to be in one batch. Starting with a root entity, we used BFS for all relation edge type to collect local neighborhoods until we reached to an end (and jump to another root entity) or reach n nodes. Thus, most sampled subgraph contains one or several connected components.

After sampling the n entities in a subgraph, the model constructs the embedding for entities and relations. We considered the simplest form of representations for entities – one-hot id for each node within this batch, which would be passed into an Embedding layer of the model that initializes the entity embedding matrix $E \in \mathbb{R}^{n \times d_e}$. For each relation, we considered the reverse relation as another independent instance of relation type. The model constructs the one-hot id for each relation and inverse-relation, as well a relation-specific adjacency list $A = \{A_\tau, \forall \tau \in R\}$ over the subgraph.

3.3 Encoder

3.3.1 multi-relation GCN

The Encoder is modeled as a multi-relation extension of graph convolution network called relation-GCN [Schlichtkrull et al., 2018] to extract the neighborhood information for each entity.

At each level, the graph propagation rule is modeled as:

$$H^{(l)} = \sigma(B_l + \sum_{\tau \in R} \tilde{A}_\tau H^{(l-1)} W_l) \tag{3}$$

where B_l is the bias term, $\tilde{A}_\tau = D_\tau^{-1/2} A_\tau D_\tau^{-1/2}$ is the symmetric diagonalization of adjacency matrix with respect to a relation τ, given the diagonal degree matrix D_τ. We treated self-edge as a special relation type and added the identity matrix into the relational adjacency matrix A to preserve nodes' own information.

We designed the model with 2 layers of R-GCN blocks, which have the capacity to capture 2-hop neighborhood information.

The output of encoder is the distribution parameters for the latent representation z. We modeled it as a Gaussian posterior to generate mean and variance of z: $\mu, \sigma = q_\phi(Z|R, E)$.

3.3.2 Non-Gaussian posterior with Inverse Autoregressive Flow

In order to model the posterior $q_\phi(z|G)$ as a more flexible, non-Gaussian distribution rather than a simple multi-variate Gaussian, we also studied the concept of Inverse Autoregressive Flow (IAF)[**?**] which gained success in image generation tasks to help improve the density estimation of the node embedding distribution.

After acquiring mean and variance of latent representation, the model first samples a latent embedding $z^{(0)}$ and passes it through 3 layers of masked transform with each variable conditioned on its predecessors with an random arbitrary ordering.

$$\mathbf{z_i^{(t+1)}} \sim p(z_i^{(t)}|\mathbf{z^{(t)}}_{1:i-1}) = z_i^{(t)} \odot \sigma_i(\mathbf{z^{(t)}}_{1:i-1}) + \mu_i(\mathbf{z^{(t)}}_{1:i-1}), \text{ where } \mathbf{z^{(0)}} \sim q_\phi(z|G) \tag{4}$$

Where $\odot$ is element-wise product, μ_i and σ_i are the parameters in the flow transform layers. After such transform, we need to subtract the log-determinant of the Jacobian of the transformation from the density estimation that was part of the expectation term in model loss defined in Equation 2 :

$$\begin{aligned} \mathbf{z} = \mathbf{z}_K &= f_D \circ f_{D-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\ \log q(\mathbf{z}|G) &= \log q_\phi(\mathbf{z}_0|G) - \sum_{t=1}^{D} \log \left| \det \frac{df_i}{d\mathbf{z}^{(t-1)}} \right| \\ &= \log q_\phi(\mathbf{z}_0|G) - \sum_{t=1}^{D} - \log \sigma_i(\mathbf{z}^{(t-1)}) \end{aligned} \tag{5}$$

3.3.3 Theoratical Analysis

We show that GCN-VAE's encoder can injectively encode computations on any given knowledge graph.

Theorem 1 Let $T : G \to \tilde{G}$ be the GCN-VAE encoder defined above. T will map non-isomorphic heterogeneous graphs $G1$, $G2$ determined by Weisfeiler-Lehman (WL) test to different embeddings.

Proof:
Given equation 1, we can write node update function at iteration k as [Xu et al., 2018]:

$$h_v^k = \sigma(h_v^{k-1}, f(h_u^{k-1} : u \in N_\tau(v), \tau \in R))$$

Here $\sigma = sigmoid(\cdot)$ is injective. At the message passing steps in encoder, f is designed to be sum aggregation of each neighborhood propagation, which is essentially the sum of masked MLP (summation after matrix multiplication). This entire operation of f is injective.

For model that incorporates IAF layers for the posterior transform, since Flow-based layers are bijective by definition, thus this operation is also injective.

The Weisfeiler-Lehman test specifies an injective hash function g:

$$l_v^k = g(h_v^{k-1}, l_u^{k-1} : u \in N_\tau(v), \tau \in R)$$

Since the composition of injective functions is an injective function, it suffices to prove by showing there always exists some function mapping Φ such that $h_v^k = \Phi(l_v^k)$.[Xu et al., 2018]

Prove by induction. At $k = 0$, the two representation will simply be the node entity matrix, Φ is the identity function. Suppose this holds for iteration $k - 1$, at k^{th} iteration, we have:

$$h_v^k = \sigma(\Phi(l_v^{k-1}), f(\Phi(l_u^{k-1}) : u \in N_\tau(v), \tau \in R))$$

Let φ represents the resulting injective function, we know it is injective because σ, Φ, f are all injective and thus their composition is also injective:

$$h_v^k = \varphi(l_v^{k-1}, (l_u^{k-1}) : u \in N_\tau(v), \tau \in R) = \varphi \circ g^{-1}\bigg(g(l_v^{k-1}, (l_u^{k-1}) : u \in N_\tau(v), \tau \in R)\bigg)$$

Assuming g is invertible, $\Phi = \varphi \circ g^{-1}$ is injective. For any iterations, the model T is injective and will reconstruct/embed non-isomorphic knowledge graphs into different embeddings.

3.4 Decoder

3.4.1 Sampling

For each query triplets consisting of either head or tail entity $e \in N_e$, and a relation $r \in R$, the decoder draws an embedding vector $z \sim q_\phi(z|G)$ and relation embedding tensor $E_\tau \sim p_\theta(E_\tau|G)$. Here the relation embedding is also retreated as a separate latent variables from the entity embedding, parametrized by θ.

We introduced mixture of Gaussian as our prior distribution for embedding spaces in order to model the ambiguity of entity and relation's semantic meaning. We hypothesized that a mixture of Gaussian prior will better model the distribution if there exist multiple clusters in embedding space for some entities and relations. We designed the number of mixtures to be $k = 10$.

In generation task, we first sample from one of the 10 mixtures, and then sample from the Gaussian distribution. $p(z) = \sum_{i=1}^{k} \frac{1}{k}\mathcal{N}(z|\mu_i, diag(\sigma_i^2))$

3.4.2 Triplet Likelihood Estimation

After sampling step, we used a bilinear DistMult operation [Yang et al., 2014] (dot-product of entities and relation embedding) with a sigmoid layer to score the edge likelihood for each triplet:

$$p_\theta((h, \tau, t)|G) = \sigma(z_h^T E_\tau z_t) \tag{6}$$

At training time, the decoder conditions on the relation types, and outputs the likelihood for all positive and negative triplets as described in 3.1.3.

At evaluation time, the decoder enumerates all possible tail entities given a head and a relation (or enumerates all head entities given a tail and a relation), it computes the likelihood for all possible triplets efficiently using batched matrix multiplication operation.

3.4.3 Maximum Mean Discrepancy (MMD)

We consider an alternative way to analyze and compare distributions than KL divergence, because we realize that during training the KL divergence could become too restrictive and dominant in the loss term that the model struggles to improve accuracy. We chose Maximum Mean Discrepency (MMD), a kernel-based method aimed at measuring the distance between two probability distributions based on moment matching.[Gretton et al., 2012] Another reason that MMD is potentially beneficial is that it encourages to maximizes mutual information between z and decoder output [Zhao et al., 2017]. We define the MMD distance between the posterior distribution and prior distribution as follows:

$$\begin{aligned} MMD^2(q_\phi(z)), p(z)) &= \|_{X\sim q_\phi(z)}\varphi(X) -_{Y\sim p(z)} \varphi(Y)\|_2 \\ &= \langle_{X\sim q_\phi(z)}\varphi(X),_{X'\sim q_\phi(z)} \varphi(X')\rangle + \langle_{Y\sim p(z)}\varphi(Y),_{Y'\sim p(z)} \varphi(Y')\rangle - 2\langle_{X\sim q_\phi(z)}\varphi(X),_{Y\sim p(z)} \varphi(Y)\rangle \\ &=_{X,X'\sim q_\phi(z)} k(X, X') +_{Y,Y'\sim p(z)} k(Y, Y') - 2_{X\sim q_\phi(z), Y\sim p(z)} k(X, Y) \end{aligned} \tag{7}$$

With X being the samples drawn from posterior distribution (encoder output) and Y being the samples drawn from the prior distribution (mixture of Gaussian samples).

In practice, we found lower the KL divergence weights (to $1e - 3$) and replace with MMD loss term in the loss function is very helpful for model to continuous performance improvement.

Datasets	#entities	#relations	#triplets	#train triplets	#test triplets
FB15K237	14541	237	272115	251649	20466
WN18RR	40943	11	40943	37809	3134

Table 1: Link prediction dataset statistics

	FB15k-237				WN18RR			
Model	MRR	Hits @			MRR	Hits @		
	Raw	1	3	10	Raw	1	3	10
TransE	.144	.147	.263	.398	.226	.284	.422	.501
DistMult*	.241	.155	.263	.419	.43	.39	.44	.49
RotateE*	.338	.241	**.375**	.533	**.476**	**.428**	.492	**.571**
R-GCN	.248	.153	.258	.417	.363	.283	.414	.488
GCN-VAE(k=1)	.279	.189	.300	.465	.386	.294	.422	.499
GCN-GMVAE(k=10)	.281	.191	.300	.471	.409	.345	.448	.511
GCN-GMVAE-IAF	**.343**	**.243**	**.375**	**.548**	.436	.367	**.509**	.547

Table 2: Link prediction task performance comparison on Mean reciprocal rank (MRR) and Hits@m. We proposed 3 variants, GCN-VAE with single Gaussian prior (k=1), GCN-GMVAE with mixture of Gaussians prior (k=10), and a GCN-GMVAE-IAF with mixture of Gaussians prior (k=10) and 3-layer IAF transform. * means the results are directly taken from corresponding papers.

4 Experiments

4.1 Datasets

We evaluated on two most commonly used benchmarks FB15k-237 (Freebase) [Toutanova and Chen, 2015] and WN18RR (WorldNet)[Dettmers et al., 2018]. The stats are listed in Table 1.

4.2 Link prediction

We predicted whether a relation exists between a pair of entities. We used 1:10 as the ratio for positive and negative samples for training, the overall negative sampling training procedure was described in 3.1.3.

The common evaluation metrics for link prediction tasks are Mean Reciprocal Rank (MRR) and the proportion of correct entities in the top N ranks (Hits). The function $rank(\cdot)$ compares the score of the correct triplet among all corrupted ones by replacing tail entity with all other entities in the N_e.

$$MRR := \frac{1}{|D_{test}|} \sum_{(h,r,t)\in D_{test}} \frac{1}{rank((h,\tau,t))}$$

$$Hits@m := \frac{1}{|D_{test}|} \sum_{(h,r,t)\in_{test}} \mathbb{1}(rank((h,\tau,t) \leq m)$$

We implemented TransE and R-GCN (strong baseline) as baselines. The original R-GCN uses the entire dataset as input graph, which was not scalable to the two large datasets, thus we sampled subgraph as input as described in 3.2 for R-GCN's trainning and evaluation. We also took metrics from papers on DistMult[Yang et al., 2014], RotateE[Sun et al., 2019] as comparison. See Table 2

4.3 Knowledge graph generation

We could also use the GCN-VAE to suggest potential new insights between entities. The generative process is as follow:

1. Input the entire graph into the encoder (it would need to use CPU memory since entire entity embedding tensor is large)

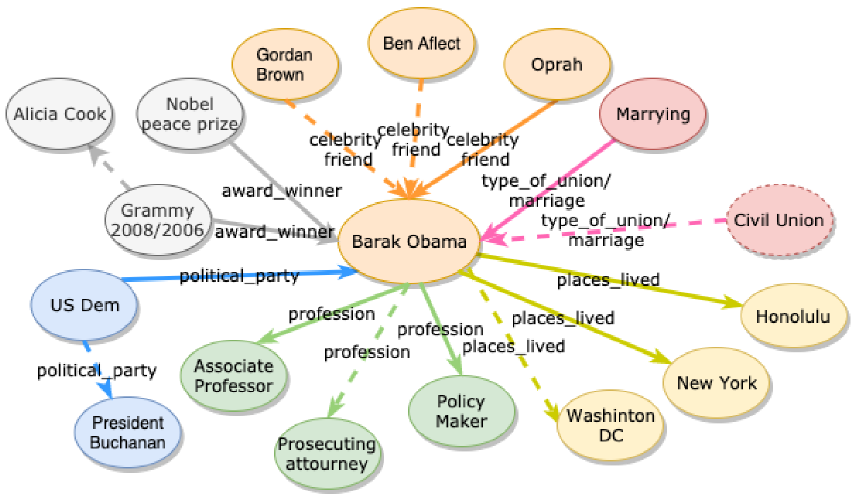

Figure 3: Sampled new links for Obama's neighborhood. The dotted lines are suggested by GCN-VAE as highest valued triplets that are not in original dataset.

2. After acquiring the node embedding distribution parameters for all entities in the graph, we sample the the latent embedding from their latent distribution.
3. Sample from all the relations τ that query entity is involved with in dataset, and then sample the relation embedding for their latent distribution.
4. For each sampled relation τ, we enumerate through all tail entities that form a triplet with the query entity and τ, pass them through the decoder and pick the highest scored k tail entities that are not in the original dataset. We could also extends to the neighbors that query entity connects with and form a 2-hop neighborhood.

For illustration, we generated a local subgraph around a query entity *"Barak Obama"* in FB15k-237 dataset, as in Figure 3 Most of the suggested links are sensible, some are incorrect such as ("Barak Obama", "profession", "Prosecuting attourney") and ("Alicia", "award_winner", "Grammy 2008/2006") were some are correct, such as ("Barak Obama", "places_lived", "Washington DC"), ("US Dem", "political party", "President Buchanan").

5 Conclusion and Future Work

In this paper we propose GCN-VAE, a new probabilistic framework for learning representations of entities and relations in large-scale knowledge graphs. GCN-VAE consists of a relational graph convolutional network encoder to derive the distribution of embeddings given input graph, and a decoder to reconstruct the graph by predicting the likelihood for each possible triplets. We learned a powerful representation of knowledge graph's relational structure and achieve supeior or close to SOTA results on link prediction benchmarks.

We found that modeling the uncertainty of entities and relations in a knowledge graph helps with the accuracy and generalization of link prediction tasks. This can be seen as the boost in performance between our GCN-GMVAE(k=10) model from the R-GCN model and the single Gaussian prior variant of our model. Since the encoder of our model is built upon R-GCN, the performance gain can be explained by their major different: GCN-VAE models a probabilistic distribution of embedding space. We believe that allowing for triplets with more complex contexts have more uncertainty is a desirable feature of knowledge graph embedding. We also found that modeling a more flexible non-Gaussian posterior with flow-based layers provide additional gains by improving the variational inference process.

In the future, we would extend this study by experimenting with different decoder structure such as RotateE, ConvE, or an end-to-end decoder structure. We would also like to visualize the latent embedding spaces to better understand the uncertainty for different types of entities and relations. Lastly, we could improve the model training process by providing more adversarial negative samples.

References

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.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. *arXiv preprint arXiv:1803.10459*, 2018.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 623–632. ACM, 2015.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*, pages 412–422. Springer, 2018.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space, 2019.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. *arXiv preprint arXiv:1802.08773*, 2018.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*, 2017.