Probabilistic Logic Graph Attention Networks for Reasoning

L Vivek Harsha Vardhan*  
School of Computing  
National University of Singapore  
Singapore  
harsha@comp.nus.edu.sg

Guo Jia*  
School of Computing  
National University of Singapore  
Singapore  
guojia@u.nus.edu

Stanley Kok  
School of Computing  
National University of Singapore  
Singapore  
skok@comp.nus.edu.sg

ABSTRACT

Knowledge base completion, which involves the prediction of missing relations between entities in a knowledge graph, has been an active area of research. Markov logic networks, which combine probabilistic graphical models and first order logic, have proven to be effective on knowledge graph tasks like link prediction and question answering. However, their intractable inference limits their scalability and wider applicability across various tasks. In recent times, graph attention neural networks, which capture features of neighbouring entities, have achieved superior results on highly complex graph problems like node classification and link prediction. Combining the best of both worlds, we propose Probabilistic Logic Graph Attention Network (pGAT) for reasoning. In the proposed model, the joint distribution of all possible triplets defined by a Markov logic network is optimized with a variational EM algorithm. This helps us to efficiently combine first-order logic and graph attention networks. With the goal of establishing strong baselines for future research on link prediction, we evaluate our model on various standard link prediction benchmarks, and obtain competitive results.

CCS CONCEPTS

• Computing methodologies → Probabilistic reasoning; Artificial intelligence; Knowledge representation and reasoning;

KEYWORDS

Graph attention networks, Markov logic networks, Link prediction, Knowledge graphs

ACM Reference Format:


1 INTRODUCTION

Knowledge graphs are used in recommendation systems [6], semantic search [1, 2] and question answering [28]. Knowledge graph reasoning, which is the task of inferring hidden relations between entities based on observed relations, is an active area of research. Markov logic networks [17] combine first-order logic and probabilistic graphical models. However, the computational intractability of Markov logic networks limit their performance and applicability to various tasks. In another line of work, graph convolutional networks [9] (e.g., R-GCN [19]) have shown their applicability to relational data. Specifically, attention-based knowledge graph embeddings [11] have shown significantly superior performance on tasks like link prediction. But they do not consider logic rules and therefore miss the knowledge that can be inferred from them. Hence, we leverage the best of both worlds by combining Markov logic networks and graph attention neural networks via a variational EM algorithm [12]. In the E step, we use graph-attention-neural-network embeddings for inferring the unobserved triplets. In the M step, the weights of the rules of a Markov logic network are updated based on the observed triplets and the inferred triplets obtained from the aforementioned embeddings. Our proposed Probabilistic Graph Attention Network (pGAT) yields competitive results over the present state-of-the-art baselines on the FB15K-237 [8] and WN18RR [23] datasets for link prediction. Our contributions lie in proposing pGAT, and establishing strong baselines for future research to compare against.

2 RELATED WORK

First order logic has been extensively used for reasoning in the past [21, 26]. Markov logic networks [18], which combines logic rules and probabilistic graphical models, are very effective at reasoning but their inference remains intractable for large datasets like those typically used for knowledge base completion. This is due to their complicated graph structures and their underlying combinatorially-huge computational graphs.

Knowledge graph embedding methods [5, 20] are recent advancements that use low dimensional embeddings of entities and relations to represent their semantics and reason about them. These methods use various scoring functions to model different rule patterns. In TransE [5], each relation is represented as a translational vector to model composition rules and inverse rules effectively. RotateE [20] models each relation as a rotation in a complex space. Convolution neural networks (CNN) based models, like ConvE [8] and ConvKB [13], use convolution filters on knowledge graphs for link prediction. Recent works like KBAT [11] show that graph attention networks lead to more effective knowledge graph embeddings. pLogicNet [16] uses Markov logic networks with neural networks for inference. However, pLogicNet’s vanilla neural network does not effectively model relational data because it considers each triplet independently, and thus neglects the interaction among triplets. Some recent works like Graph Markov Neural Networks...
3 APPROACH

We first introduce the knowledge graph embedding model employed in our framework. Then we elaborate on how the learned graph-attention-based embeddings and the parameter learning of Markov logic networks benefit each other in an iterative manner.

A knowledge graph $KG$ is represented as a set of relational triples, $KG = \{(e_h, r, e_t)\}$, where $e_h, e_t \in E$, $r \in R$, and $E$ and $R$ are a set of entities and a set of relations respectively. $T_O$ and $T_U$ respectively denote the set of observed triples and unobserved triples. Each triple $(e_h, r, e_t) \in T_O$ indicates that there exists a directed relation $r$ from head entity $e_h$ to tail entity $e_t$ ($e_h \neq e_t$).

3.1 Graph Attention-Based Embeddings

A recent model KBAT [11] surpassed the performances of several state-of-the-art knowledge-embedding models. We choose KBAT as our knowledge embedding component because its embeddings provide rich semantic information about the relations between entities. This allows for more accurate inference for the subsequent parameter learning of logical rules.

The inputs to the attention layer in KBAT are a node embedding matrix $X \in \mathbb{R}^{N \times d_e}$, and a relation embedding matrix $Q \in \mathbb{R}^{M \times d_r}$, where $N$ and $M$ are the total numbers of entities and relations respectively, and $d_e$ and $d_r$ are the dimensions of the embedding vectors for each entity and each relation respectively.

The feature vector of each triple $(e_i, r, e_j)$ is represented as $t_{ijk} = W_t[x_i||x_j||q_k]$, where $W_t$ represents the weight matrix of a linear transformation over the concatenation of entity embeddings $x_i, x_j$, and relation embedding $q_k$.

The vector $t_{ijk}$ is used to produce a corresponding attention score $e_{ijk} = \text{LeakyReLU}(W_e t_{ijk})$. The attention score $e_{ijk}$ is normalized through a soft max function over all entities and relations in the neighborhood of entity $e_i$ to give the normalized value $a_{ijk}$. The new embedding vector for entity $i$ is computed as the weighted sum of all its connected triple representations weighted by the normalized attention values. A multi-head attention mechanism is used to compute the embedding vectors for entities. The final entity embedding vector is obtained by concatenating the entity embedding vectors from multiple attention mechanisms, and averaging over the vectors as follows:

$$\hat{x}_i = \sigma \left( \frac{1}{C} \sum_{c=1}^{C} \left( \sum_{j \in N_i} \sum_{k \in Q_{ij}} a_{ijk}^c t_{ijk} \right) \right),$$

where $N_i, Q_{ij}$ represent all connected nodes and relations of entity $i$ respectively, and $C$ is the number of head attentions. The final embedding matrices for entities $\tilde{X}$, and for relations $\tilde{Q}$ are computed as:

$$\tilde{X} = W^x X + \hat{X},$$
$$\tilde{Q} = W^q Q,$$

where $W^x, W^q$ are both parameters of linear transformations. This model is trained by minimizing the pairwise ranking loss:

$$\mathcal{L} = \sum_{s_{ijk} \in T_O} \sum_{s'_{ijk} \in T_O} \max\{\theta + d_{s_{ijk}} - d_{s'_{ijk}}, 0\},$$

where $\theta$ is a margin hyper-parameter, and the distance function of each triple is given by $d_{s_{ijk}} = ||\hat{x}_i + \tilde{q}_k - \hat{x}_j||_1$ [5]. $T'_O$ is the set of observed triples.
Table 1: Results of link prediction on test sets of FB15K-237 and WN18RR respectively. The best scores are in bold. The second best scores are underlined.

<table>
<thead>
<tr>
<th>Method</th>
<th>MR</th>
<th>MRR</th>
<th>H@1</th>
<th>H@3</th>
<th>H@10</th>
<th>MR</th>
<th>MRR</th>
<th>H@1</th>
<th>H@3</th>
<th>H@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE [5]</td>
<td>323</td>
<td>0.279</td>
<td>19.8</td>
<td>37.6</td>
<td>44.1</td>
<td>2300</td>
<td>0.243</td>
<td>4.27</td>
<td>44.1</td>
<td>53.2</td>
</tr>
<tr>
<td>DistMult [27]</td>
<td>512</td>
<td>0.281</td>
<td>19.9</td>
<td>30.1</td>
<td>44.6</td>
<td>7000</td>
<td>0.444</td>
<td>41.2</td>
<td>47</td>
<td>50.4</td>
</tr>
<tr>
<td>ComplEx [24]</td>
<td>546</td>
<td>0.278</td>
<td>19.4</td>
<td>29.7</td>
<td>45</td>
<td>7882</td>
<td>0.449</td>
<td>40.9</td>
<td>46.9</td>
<td>50.4</td>
</tr>
<tr>
<td>RotateE [20]</td>
<td>185</td>
<td>0.297</td>
<td>20.5</td>
<td>32.8</td>
<td>48.0</td>
<td>3277</td>
<td>0.470</td>
<td>42.2</td>
<td>56.5</td>
<td></td>
</tr>
<tr>
<td>ConvE [8]</td>
<td>245</td>
<td>0.312</td>
<td>22.5</td>
<td>34.1</td>
<td>49.7</td>
<td>4464</td>
<td>0.456</td>
<td>41.9</td>
<td>47</td>
<td>53.1</td>
</tr>
<tr>
<td>ConvKB [13]</td>
<td>216</td>
<td>0.289</td>
<td>19.8</td>
<td>32.4</td>
<td>47.1</td>
<td>1295</td>
<td>0.265</td>
<td>5.82</td>
<td>44.5</td>
<td>55.8</td>
</tr>
<tr>
<td>R-GCN [19]</td>
<td>600</td>
<td>0.164</td>
<td>10</td>
<td>18.1</td>
<td>30</td>
<td>6700</td>
<td>0.123</td>
<td>20.7</td>
<td>13.7</td>
<td>8</td>
</tr>
<tr>
<td>KBAT [11]</td>
<td>204</td>
<td>0.431</td>
<td>35.5</td>
<td>46.2</td>
<td>57.8</td>
<td>1970</td>
<td>0.431</td>
<td>35.2</td>
<td>47.3</td>
<td>57.4</td>
</tr>
<tr>
<td>BLP [7]</td>
<td>1985</td>
<td>0.092</td>
<td>6.2</td>
<td>9.8</td>
<td>15.0</td>
<td>12051</td>
<td>0.254</td>
<td>18.7</td>
<td>31.3</td>
<td>35.8</td>
</tr>
<tr>
<td>MLN [17]</td>
<td>1980</td>
<td>0.098</td>
<td>6.7</td>
<td>10.3</td>
<td>16.0</td>
<td>11549</td>
<td>0.259</td>
<td>19.1</td>
<td>32.2</td>
<td>36.1</td>
</tr>
<tr>
<td>pLogicNet [16]</td>
<td>173</td>
<td>0.330</td>
<td>23.1</td>
<td>36.9</td>
<td>52.8</td>
<td>3436</td>
<td>0.230</td>
<td>1.5</td>
<td>41.1</td>
<td>53.1</td>
</tr>
<tr>
<td>pLogicNet* [16]</td>
<td>173</td>
<td>0.332</td>
<td>23.7</td>
<td>36.7</td>
<td>52.4</td>
<td>3408</td>
<td>0.441</td>
<td>39.8</td>
<td>44.6</td>
<td>53.7</td>
</tr>
<tr>
<td>pGAT</td>
<td>181</td>
<td>0.457</td>
<td>37.7</td>
<td>49.4</td>
<td>60.9</td>
<td>1868</td>
<td>0.459</td>
<td>39.5</td>
<td>48.9</td>
<td>57.8</td>
</tr>
</tbody>
</table>

of negative triples created by randomly replacing the head or tail entity of triples in \( T_0 \) We employ ConvKB [13] as our decoder after updating entity embeddings and relation embeddings, and train it using soft-margin loss [25]. More details about the decoder are found in [13, 25].

### 3.2 Parameter learning of MLN

A Markov logic network (MLN) [17] consists of a set of weighted first-order logic formulas, which can be viewed as templates for constructing Markov networks. Given a knowledge graph \( KG \), the joint probability of all relational triples is given by

\[
\begin{align*}
    p(s) &= \frac{1}{Z} \exp \left( \sum_{l=1}^{L} \phi_l(n_l(s)) \right),
\end{align*}
\]

where \( s \) represents triple \( (e_i, r_i, e_j) \in KG \), \( L \) is the total number of logical formulas in the MLN, \( n_l(s) \) is the number of true groundings of the \( l \)th logical rule (that has \( s \) as is consequent) according to the truth value of \( KG \)‘s triples, and \( \phi_l \) is the weight parameter of the \( l \)th logical rule. We follow the training regime which uses a variational EM algorithm [12]. Due to the intractability of directly optimizing the joint probability distribution, we instead optimize the following evidence lower bound (ELBO) of its log-likelihood function.

\[
\begin{align*}
    \log p(s_o) &\geq \log p(s_o) - KL[q(s_u) || p(s_u | s_o)] \\
    &= \int q(s_u) \log p(s_u, s_o) ds_u - \int q(s_u) \log q(s_u) ds_u,
\end{align*}
\]

where \( KL \) denotes the KL divergence [10], and \( q(\cdot) \) represents the variational distribution of unseen triples \( s_u \). Equality in the above expression holds when \( q(s_u) = p(s_u | s_o) \). In the variational E-step, we fix \( p(\cdot) \) and optimize \( q(\cdot) \) by inferring the true posterior distribution with mean-field approximation [14], which is based on the learned embeddings of the knowledge graph embedding model (KBAT) we have trained, i.e.,

\[
    q(s_u) = \prod_{s_u \in T_U} \Pr(s_u | f_{score}(s_u)).
\]

\( \Pr(\cdot) \) measures the true probability of triples with the score function \( f_{score} \), which is computed on entity and relation embeddings. The score function measures the plausibility of triples. Through minimizing the KL divergence between \( q(s_u) \) and the true posterior distribution \( p(s_u | s_o) \), the optimal \( q(s_u) \) is computed as

\[
    \log q(s_u) = \mathbb{E}_{q(s_u, MB)} [\log p(s_u | s_u, MB)] + \text{const},
\]

where \( s_u, MB \) is the Markov blanket of \( s_u \). If there exists any unseen triple in \( s_u, MB \), we replace it with a sample from the potential distribution inferred by the knowledge graph embedding model.

To further optimize the parameter, we enhance the knowledge embedding model by updating its training dataset with added unseen triples, which are predicted by the MLN. In the M-step, we fix \( q(\cdot) \) and update \( p(\cdot) \) by maximizing the pseudo-likelihood [3] as follows:

\[
    \mathcal{L}_{pseudo} = \mathbb{E}_{q(s_u)} \left[ \sum_{s_u \in T_U, s_o \in T_O} \log p(s_u, s_o) \right] = \mathbb{E}_{q(s_u)} \left[ \sum_{s \in T_O} \log p(s | s_u, MB) \right].
\]

Similar to the variational E-step, we fill those unseen triples in the Markov blanket with samples obtained from our knowledge embedding model. Therefore, by alternating between the variational E-step and an M-step, this framework allows knowledge sharing.
between the knowledge graph embedding model and the MLN as in shown in Figure 1.

4 EXPERIMENTS

4.1 Datasets
We evaluated our model on the task of link prediction knowledge graphs. We report the results of our work on two benchmark datasets FB15K-237 from Freebase [4] and WN18RR from Wordnet [5], which are created to resolve the reversible relation problem in their respective source datasets.

4.2 Evaluation Metrics
We remove a head entity or tail entity and predict the resulting triple using our proposed model. The average scores from replacing head and replacing tail are reported. Other baselines results are taken from their official code implementations and KBAT [11]. We evaluated our models in a filtered setting following previous works [11]. We report mean reciprocal rank (MRR), mean rank (MR) and the proportion of correct entities in the top N ranks (Hits@N) for N = 1, 3, and 10.

Table 4: Hyperparameters for Markov Logic Network.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Rule-threshold</th>
<th>Triplet-threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB15K-237</td>
<td>0.6</td>
<td>0.7</td>
</tr>
<tr>
<td>WN18RR</td>
<td>0.1</td>
<td>0.4</td>
</tr>
</tbody>
</table>

4.3 Implementation Details
We used TransE [5] to initialise the knowledge graph embedding before training. The knowledge graph embedding is used for inference at test time. The candidate rules for Markov logic network are generated using a brute force method where we search for all possible composition rules, inverse rules, symmetric rules and sub-relation rules in the observed triplets. We use an accuracy threshold above which we select a rule as a candidate rule in our Markov logic network. We also use a probability threshold above which we add a triplet to the training set for knowledge graph embeddings. We use the Adam optimizer for training knowledge graph embeddings. The hyper-parameters of our model are in Table 2, Table 3 and Table 4.

4.4 Baselines
We compare our proposed pGAT model against knowledge graph embedding methods (TransE, DistMult, ComplEx, RotateE [5, 20, 24, 27]), convolution-based knowledge graph embedding methods (ConvE [8] and ConvKB [13]), and graph-neural-networks-based knowledge graph embedding methods (R-GCN [19] and KBAT [11]). We also compare against Markov logic networks (MLN) [17], Bayesian logic programs (BLP) [7], pLogicNet, and pLogicNet* [16].

5 RESULTS AND DISCUSSION
The results on two datasets are presented in Table 1. Our proposed pGAT model outperforms all the baseline methods on the FB15K-237 dataset, achieving 0.457 for MRR and 37.7 for Hits@1. We are one of the top two performers on the WN18RR dataset for most of the metrics. The good empirical results of our PGAT model bears out the efficacy of incorporating domain knowledge in the form of Markov logic rules and using graph attention networks to leverage neighbourhood information at various distances.

6 CONCLUSION
We propose pGAT, which combines Markov logic networks and graph attention networks, for the task of link prediction in knowledge graphs. Our work establishes strong baselines for future work on knowledge base completion. As future work, we plan to extend our model to inductive settings [22], and explore efficient ways to find rules for the Markov logic network. This research is partly funded by MOE AcRF Tier 1 grant (R-253-000-146-133) to Stanley Kok.

REFERENCES