Introduction

Knowledge graphs collect and organize relations and attributes about entities, which are important in many applications, including question answering and information retrieval.

Markov Logic Networks (MLNs) is a probabilistic graphical model that combines hard logic rules. It can be applied to various tasks on knowledge graphs. The logic rules enable MLNs to perform tasks with small amount of labeled data. But the inference in MLN is computationally intensive.

GNN is popular for solving graph related problems. But GNN-based methods require sufficient labeled instances on specific end tasks to achieve good performance. But knowledge graph's long-tail nature (i.e., a large portion of the relations are only in a few triples) makes data scarce for those long-tail relations.

The authors propose a method to combine MLN and GNN, aiming for a method which is data-driven yet can still exploit the prior knowledge encoded in logic rules. Specifically, they designed a GNN called ExpressGNN, and use it as the inference module to enable fast inference in MLN.

Related work

Probabilistic logic reasoning on graph: Relational Markov Networks (RMNs; Taskar et al. (2007)) and Markov Logic Networks (MLNs; Richardson & Domingos (2006)). They introduced probabilistic graphical models in logic reasoning.

More about MLN: it has been widely studied due to the principled probabilistic model and effectiveness in a variety of reasoning tasks. However, the inference and learning in MLNs is computationally expensive. Though lots of work tried to improve the original MLNs in both accuracy and efficiency, Scalability is still one main issue of MLNs, which hinders MLNs to be applied to industry-scale applications.

GNN: Graph neural networks (GNNs; Dai et al. (2016); Kipf & Welling (2017)) can learn effective representations of nodes by encoding local graph structures and node attributes. Recently, Qu et al. (2019) proposed Graph Markov Neural Networks (GMNNs), which employs GNNs together with conditional random fields to learn object representations.

Knowledge Graph embedding: there are many knowledge graph embedding methods, such as TransE (Bordes et al., 2013), NTN (Socher et al., 2013), DistMult (Kadlec et al., 2017), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019). They are effective in learning embeddings of both entities and relations, but are not able to leverage logic rules, which can be crucial in some relational learning tasks, and have no consistent probabilistic model. Qu & Tang

(2019) has proposed a probabilistic Logic Neural Network (pLogicNet), which integrates knowledge graph embedding methods with MLNs with EM framework.

Background

Knowledge graph: can be represented as a tuple K = (C, R, O) consisting of a set C = {c1,...,cM} of M entities, a set R = {r1,...,rN} of N relations, and a collection O = {o1,...,oL} of L observed facts.

First-order logic:
Constants (entities): an object
Predicates (relations): a function over constants r(·) : C × ... × C → {0,1}. E.g. Friends(c,c')
Ground predicates: a binary variable obtained by assigning constants to predicates. e.g. Friends(Andy, Billy)
logic formula: f (·) : C × . . . × C □→ {0, 1} is a binary function defined via the composition of a few predicates. E.g. Smoke(c) ∧ Friend(c, c′) ⇒ Smoke(c′) <=> ¬Smoke(c) ∨ ¬Friend(c, c′) ∨ Smoke(c′)
Ground formula: a binary variable obtained by assigning constants to formula
Observed fact: a truth value {0,1} assigned to a ground predicate
Unobserved fact: latent variables

Knowledge base represented as knowledge graph:
A bipartite graph GK = (C,O,E), where nodes on one side of the graph correspond to constants C and nodes on the other side correspond to observed facts O. The set of T edges, E = {e1, . . . , eT }, connect constants and the observed facts. An edge e = (c, o, i) between node c and o exists, if the ground predicate associated with o uses c as an argument in its i-th argument position.

Markov Logic Network:
Defined by a joint distribution: Pw (O, H) := 1 exp □ □ wf □ φf (af)□

Solution

The authors train the model by optimizing the variational evidence lower bound (ELBO) of the data log-likelihood. In order to optimize the ELBO, they use the variational EM algorithm.

For E-step inference, they use variational inference with mean-field distribution to approximate the true posterior distribution. And the mean-field distribution (the approximated posterior) is modeled as a GNN (the ExpressGNN they proposed). They also use mini-batch sampling of formula in each iteration, in order to reduce the computational cost. Besides, they add supervised objective on top of ELBO if labeled data is sufficient. This objective can capture the predicates that are not well covered by logic rules but have sufficient observed facts.

For M-step, they used pseudo-log-likelihood (Richardson & Domingos, 2006) as an alternative objective to optimize P(O,H), since directly optimizing it is intractable, as it involves exponential terms in Z(w). Then they optimize the objective by gradient descent.

The ExpressGNN used in E-step has three parts: a vanilla GNN that computes the embedding of entity nodes, the second part extends the embedding with tunable embeddings, and the third part uses the new embeddings to define the variational posterior.

Experimental results

The paper evaluates the effectiveness of ExpressGNN on four benchmarks: UW-CSE, Cora, Synthetic Kinship dataset, and FB15K-237 which is constructed from Freebase. All the experiments are conducted with a GPU-enabled machine and the models are trained by using Pytorch with Adam Optimizer. For Kinship, Cora, and UW-CSE, the training is performed under a fixed number of iterations and they use a small subset from the original split for parameter tuning. While for the Freebase dataset, the model is not trained with a fixed number of iterations and they use the original split for hyperparameter tuning. Also, for each dataset, the paper searched the appropriate configuration of ExpressGNN, which includes the embedding size, split points of tunable embedding and GNN embedding.

The kinship, UW-CSE and Cora are used to evaluate the accuracy and efficiency of the model. For the metrics, they used AUC-PR for the accuracy and clock time for efficiency. The experiment showed promising results on accuracy for all datasets. However, it would lose some accuracy if used on large dataset. The efficiency of ExpressGNN is outstanding on UW-CSE as the dataset grows linearly, the efficiency time could remain constant.

THe Freebase dataset is used to test the knowledge base completion test. The results show that in regards to MMR and Hits, ExpressGNN outperforms all the baselines significantly. In addition, the results show that ExpressGNN performs better on efficiency if trained on a small dataset than other baselines. However, if trained on large datasets, the baselines would catch up.

major conclusions and pros

This paper studies the probabilistic logic reasoning problem, and proposes ExpressGNN to combine the advantages of Markov Logic Networks in logic reasoning and graph neural networks in graph representation learning. ExpressGNN addresses the scalability issue of Markov Logic Networks with efficient stochastic training in the variational EM framework.

Cons

One of the cons is that the model with full EM would potentially place some unknown restrictions on datasets. As our group noticed, developers found some incompatibility between the ExpressGNN source code with other datasets, which would always return some errors.