Report on: Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks

Sai Shreyashi Penugonda (UIN: 731008208)

1 Problem Statement and Motivation of Research

In the real world, it is important to analyze what is important and focus on certain aspects. The way how our brain decides in a lot of cases, for example, which work to prioritize first or which course do we like the most is dependent on multiple factors like in the latter example, it could be due to the field of our interest from our prior readings, the company we worked for, course instructor. The importance for each of these related fields could be different and influence us in different ways. In this research, the authors have tried to harness these types of qualities to estimate node importance in knowledge graphs using Graph Neural Networks. Knowledge graphs are important in multiple use-cases because they help represent and analyze heterogeneous information as well as relationships. Relation to thinking in real-life is understood which is reflected in the current real-time needs and applications studied using knowledge graphs like financial analytics (startup-investors relations), item recommendations (e-commerce user-product relations), resource allocation (manufacturing resources-orders relations). Existing research and implementations for node importance prediction, have two major drawbacks: inability to harness more information from KGs as well as low flexibility in depicting and analyzing complex entity-importance relationships. The reason why it is important to study node importance efficiently in KGs has two primary reasons: these graphs are huge with millions or more entities so update operations are tricky and information validation is heavy-duty in terms of resources as well as time consumed. Thus, if node importance is known it would be easy to prioritize and delegate limited resources if that's the required use case. This signifies node importance is one major significance measure in KGs. These issues and shortcomings in existing solutions are addressed using GNN-based model, GENI proposed in this research, whose implementation details and results are discussed further in this report.

2 Technical Contribution - Steps and Implementation Details

Preliminary Information: to keep in mind before proceeding to discuss the solution suggested is **In-Domain** and **Out-Domain Estimation** of importance scores to both be computed close to ground-truth estimations. In-Domain estimations are scores that need to be found for nodes of the same type τ as the input importance scores of a subset of nodes V_S while out-domain estimations are scores that need to be found of type other than τ.

GNN: aids the framework with certain needed factors for modeling NI which are the provision of flexibility of adaptation to different types of input scores (understanding the underlying meaning to those scores), utilization of input importance scores (for a subset of nodes) rather than just the graph structural information and in general are neighborhood aware considering they harness the neighborhood aggregation property. The interesting factor is that rather than utilizing node embeddings, in this implementation they have utilized aggregation of neighboring importance scores while keeping in mind the importance of central nodes by taking into account the node centrality for adjustments to calculated scores. This ensured taking into consideration both structural and non-structural information.

Predicates: Apart from neighborhood scores aggregation and centrality measures taken, edge-types (predicates) are modeled on, to determine the effects of edge-types on node importance. It is good to see the representation power when different edge-types, different types of importance scores and

different node-types are used. The number of possible combinations to explore different information for each node and overall KG increases.

Score Aggregation: In standard GNNs, embedding aggregation is followed by non-linear transformation of node embeddings. Then, the node representations and aggregated neighborhood representations are combined. Unlike standard GNN, in GENI for aggregation to directly be dependent on NI scores, instead of intermediate feature vector(hidden embedding) usage to get the current layer embedding, intermediate scores of all the neighbors of the node under consideration (node i) are used for score estimation for node i in current layer(l^{th} layer). Node embeddings are heavier in terms of model parameters, so naturally, their propagation will be computationally more expensive compared to the propagation of NI scores, thus validating to some extent the performance gains seen in GENI. Only for the initial/seed importance scores to be computed a **Scoring Network** is used which in GENI's case is a fully-connected neural network. For score estimation in further layers, a learnable weight(α_{ij}^l) between node i and j, signifying attention of node i on node j computed by SA head in a particular layer 'l'. This computation of α_{ij} is done using Predicate-Aware attention mechanism.

Predicate-aware Attention Mechanism: breaking down the terms gives us 'Predicate' and 'Aware'. 'Predicate' because it takes into consideration the predicate for computing NI score and this is using predicate embeddings which are basically predicate feature vectors. 'Awareness' is achieved in two ways: one, these predicate embeddings are not node specific or layer-specific, they are shared across all layers and second, the predicate embeddings are learned hence are not static in terms of flexibility to edge-types. This improves predictions because the ability to be adaptive by learning constantly in a model is always desired, the more a model learns from the information provided, the better it would tend to give appropriate results. Since predicate embeddings are shared across layers and attentional layer is used to apprehend the relationship between node i, predicate(each since there can be many between two nodes) between them and node j, the 'awareness' factor is achieved. Thus, the sequence of operations to get attention of node i on node j in layer l is:

1. Intermediate score computation for node i and node j.
2. Mapping from predicate to its embedding(learned).
3. Concatenation of intermediate scores for both nodes and mapped predicate-embedding.
4. This concatenated vector is fed into the attentional layer with parameter $\vec{a}$ weight vector.
5. Summation over all predicates between node i and j to compute attention scores for each predicate.
6. Non-linearity over the attentional layer output applied.
7. Softmax normalization applied over output from non-linearity function.

The entire predicate-aware attention mechanism using SA is to get the attention of node i on node j in layer l (α_{ij}^l) which is utilized for score estimation as mentioned before. While this helps us get the effects of predicates on NI scores, the effects of structural information are also incorporated using centrality adjustments.

Centrality Adjustments: is considered in terms of in-degree of each node and what is interesting is that, in order to incorporate the possible difference of significance for a node gauged by score estimation(without centrality adjustment) and due to in-degree centrality, two learnable parameters (γ & β) are used to enable centrality to be scaled and shifted accordingly. This centrality score for each node($c^*(i)$) is applied to score estimation from last layer(L: $s^L(i)$) followed by non-linear function application to get the score estimation for node i($s^*(i)$).

Model Architecture: in the proposed model has a few interesting aspects which are noteworthy. Firstly, instead of using just one SA layer which would aggregate scores from immediate neighbors of each node, they're using multiple SA layers to aggregate scores(farther information) for each node from farther neighbors. Secondly, instead of using single SA head, they're using multiple attention heads working independently of one another. So, the score aggregations, attention computations and centrality score adjustments are all performed specific to a single attention head with independent set

of learnable parameters ($\vec{a}_{h,l}, \gamma_h, \beta_h$). Once score estimations for each head in final layer L are computed, they are averaged and non-linearity function is applied to get NI score for a node i. Equation breakdown for $\alpha_{ij}^{h,l}$ for better clarity is in the appendix.

3 Experimental Results and Findings

The datasets used for the experiments are relevant since they are real-world data and both OOD as well as in-domain evaluation were performed(except FB15K for which OOD evaluation wasn't performed). The diversity of the datasets in terms of number of nodes, edges, number of edge-types and number of input scores ensured studying effects of these factors on performance in real-world data. OOD evaluation is important to analyze how good or bad the model is performing since that determines how well the model has learned the relationships for other node-types (non-τ) as well.

For performance evaluation, NDCG (measure of the quality of ranking top 'k' entities) and SPEARMAN (measure of the correlation of ranking between ground-truth and predicted score rankings) measures were considered.
Results comparison from the real-world datasets between GENI and other models show that in almost all cases GENI significantly performed better. This is because it bridges the gaps existent in trainable models(GAT) and non-trainable models(PR, PPR, HAR). The gaps are, either considering only structural information, not accounting for edge-types information, not accounting for node-type information, incorrect predicate weight adjustment and fixed model structure meaning no learnable parameters.

For a thorough analysis of GENI, the authors of this paper also analyzed effects of predicate consideration, centrality adjustments and parameters' modifications (SA, predicate & hidden layer parameters), which were the major contributions in their proposed model.

They experimented with fixed predicate embedding for each predicate being shared across layers versus learned(distinct) embeddings for each predicate, where, as expected, the distinct predicate embeddings model saw an increase in performance by around 4% in NDCG@100 and 13% in SPEARMAN measures.

Next, they also experimented with fixed CA versus shifted and scaled CA. In this experiment, when NI scores were in agreement with node centrality, the results were comparable while when they weren't in agreement flexible CA performed 8% better by NDCG@100 ranking and 27% better by SPEARMAN measures. These are significantly higher and although studies and results of in-domain and OOD have been done (reported in Tables 4 and 5) where GENI has outperformed other models by all NDCG and SPEARMAN measures, one thing that could have been analyzed along with this was the effect of CA on in-domain and OOD estimations. Second best results are also marked in these tables but as is noticeable, these are not consistently good for any single model across the datasets, to conclude which model could be a close second to GENI.

The next aspect was parameter sensitivity where performance gain was visible when the number of SA heads and SA layers was increased. SA layers as anticipated would have captured information from a larger neighborhood and more SA heads performed better because multiple heads attention to the same information in different ways allowing for better understanding. One exception observed was when predicate embedding size was increased, after a certain value, there was a degradation in performance. This could be due to over-compensation (more values than required to represent certain relation) of representing the comparatively simple relation. When reading on this aspect, one work [2] focused on the discrepancies of representation of same entity in different contexts(Ex: Steve Jobs-director-Apple & Christopher Nolan-director-Inception), an edge-centric translational embedding model(TransEdge) was proposed which seemed an interesting area for handling such relations. According to the plots for parameter sensitivity, NDCG measures saw slight improvement or no improvement while SPEARMAN measures showed significant improvements especially with SA heads and layers increased. Inconsistent behavior was seen with number of hidden layers in the scoring network, but possibly since an increasing trend was seen until a certain point, performance is concluded to have increased with an increase in hidden layers.

4 Conclusion and Future Work

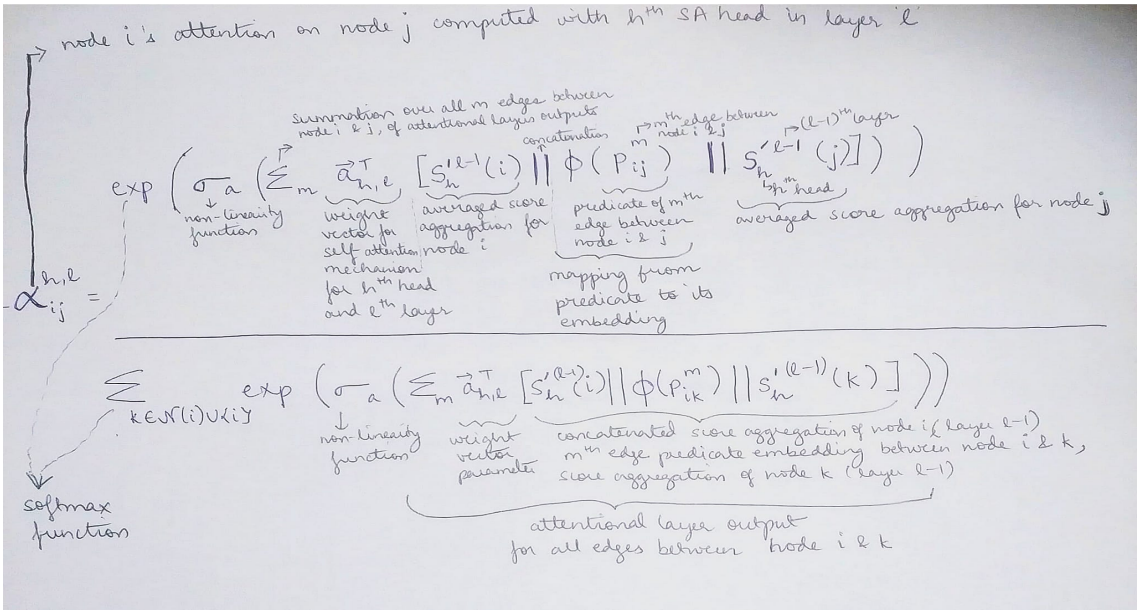
This research is quite wholesome in the sense of covering all bases in analysis and experimentation to gauge the performance on real world data. GENI's performance measures have been compared and proved to be better than existing models in this research. One analysis aspect that can be explored is the effect of CA being fixed or flexible on in-domain and OOD estimations. In GENI training, mean-squared error was used in loss function, other error estimations like log loss, exponential loss and hinge loss could also be experimented with. Another extension could be from the TransEdge model[2] to incorporate relation-contextualized predicate embeddings. One area of exploration suggested within this paper was to estimate NI using various independent input sources. The research and technology stack(as mentioned in Experimental Settings) in this paper is especially interesting to me because I've worked with huge product graphs and performed graph analysis to pick important nodes (most vulnerable code components based on product logs). While that wasn't a knowledge graph, the aspects of node importance to be based on various edge-types and node-types is familiar to me.

5 Appendix

5.1 Abbreviations

1. KG: Knowledge Graphs
2. GNN: Graph Neural Networks
3. NI: Node Importance
4. OOD: Out-of Domain
5. GAT: Graph Attention Network
6. PR: PageRank
7. PPR: Personalized PageRank
8. HAR: Hub, Authority and Relevance Scores
9. NDCG: Normalized Discounted Cumulative Gain
10. CA: Centrality Adjustment
11. SA: Score Aggregation

5.2 Equation breakdown and some explanation for clarity



6 References

[1] Park, Namyong Kan, Andrey Dong, Xin Zhao, Tong Faloutsos, Christos. (2019). Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks. 596-606. 10.1145/3292500.3330855.
[2] Sun, Zequn Huang, Jiacheng Hu, Wei Chen, Muhao Guo, Lingbing Qu, Yuzhong. (2019). TransEdge: Translating Relation-Contextualized Embeddings for Knowledge Graphs. 10.1007/978-3-030-30793-6_35.