Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks

Namyong Park[1*], Andrey Kan[2], Xin Luna Dong[2], Tong Zhao[2], Christos Faloutsos[1*]
{namyongp,christos}@cs.cmu.edu,{avkan,lunadong,zhaoton}@amazon.com
[1]Carnegie Mellon University, [2]Amazon

ABSTRACT

How can we estimate the importance of nodes in a knowledge graph (KG)? A KG is a multi-relational graph that has proven valuable for many tasks including question answering and semantic search. In this paper, we present GENI, a method for tackling the problem of estimating node importance in KGs, which enables several downstream applications such as item recommendation and resource allocation. While a number of approaches have been developed to address this problem for general graphs, they do not fully utilize information available in KGs, or lack flexibility needed to model complex relationship between entities and their importance. To address these limitations, we explore supervised machine learning algorithms. In particular, building upon recent advancement of graph neural networks (GNNs), we develop GENI, a GNN-based method designed to deal with distinctive challenges involved with predicting node importance in KGs. Our method performs an aggregation of importance scores instead of aggregating node embeddings via predicate-aware attention mechanism and flexible centrality adjustment. In our evaluation of GENI and existing methods on predicting node importance in real-world KGs with different characteristics, GENI achieves 5–17% higher NDCG@100 than the state of the art.

CCS CONCEPTS

• **Information systems → Data mining**; • **Computing methodologies → Neural networks**; *Supervised learning*.

KEYWORDS

node importance estimation; knowledge graphs; graph neural networks; attention model

ACM Reference Format:
Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, Christos Faloutsos. 2019. Estimating Node Importance in Knowledge Graphs Using Graph Neural Networks. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330855

*Work performed while at Amazon.

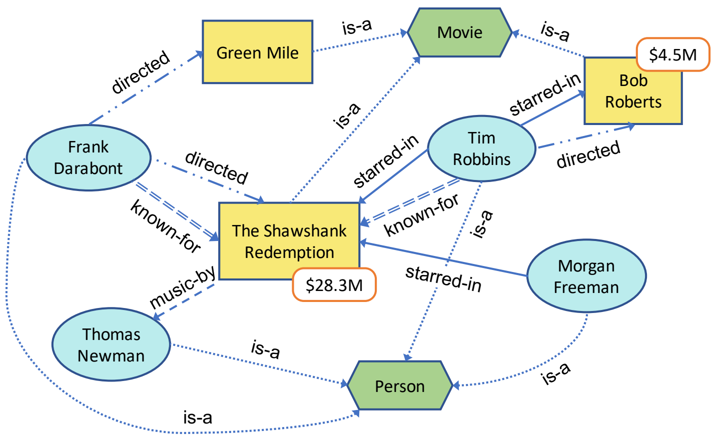


Figure 1: An example knowledge graph on movies and related entities. Different edge types represent different types of relations (e.g., “directed” and “starred-in”), and different shapes denote different entity types. Rounded rectangles are importance scores known in advance for some movies.

1 INTRODUCTION

Knowledge graphs (KGs) such as Freebase [2], YAGO [19], and DBpedia [16] have proven highly valuable resources for many applications including question answering [7], recommendation [26], semantic search [1], and knowledge completion [23]. A KG is a multi-relational graph where nodes correspond to entities, and edges correspond to relations between the two connected entities. An edge in a KG represents a fact stored in the form of “<subject> <predicate> <object>”, (e.g., “<Tim Robbins> <starred-in> <The Shawshank Redemption>”). KGs are different from traditional graphs that have only a single relation; KGs normally consist of multiple, different relations that encode heterogeneous information as illustrated by an example movie KG in Figure 1.

Given a KG, estimating the importance of each node is a crucial task that enables a number of applications such as recommendation, query disambiguation, and resource allocation optimization. For example, consider a situation where a customer issues a voice query “Tell me what Genie is” to a voice assistant backed by a KG. If the KG contains several entities with such a name, the assistant could use their estimated importance to figure out which one to describe. Furthermore, many KGs are large-scale, often containing millions to billions of entities for which the knowledge needs to be enriched or updated to reflect the current state. As validating information in KGs requires a lot of resources due to their size and complexity, node importance can be used to guide the system to allocate limited resources for entities of high importance.

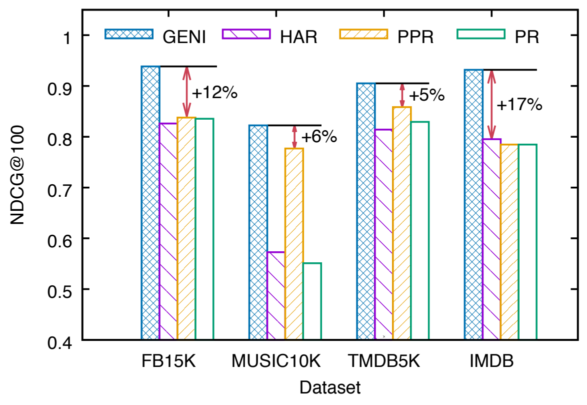


Figure 2: Our method GENI outperforms existing methods in predicting importance of nodes in real-world KGs. Higher values are better. See Section 4.4 and Table 4 for details.

How can we estimate the importance of nodes in a KG? In this paper, we focus on the setting where we are given importance scores of some nodes in a KG. An importance score is a value that represents the significance or popularity of a node in the KG. For example, the number of pageviews of a Wikipedia page can be used as an importance score of the corresponding entity in a KG since important nodes tend to attract a lot of attention and search traffic. Then given a KG, how can we predict node importance by making use of importance scores known for some nodes along with auxiliary information in KGs such as edge types (predicates)?

In the past, several approaches have been developed for node importance estimation. PageRank (PR) [18] is an early work on this problem that revolutionized the field of Web search. However, PR scores are based only on the graph structure, and unaware of importance scores available for some nodes. Personalized PageRank (PPR) [11] dealt with this limitation by letting users provide their own notion of node importance in a graph. PPR, however, does not take edge types into account. HAR [17] extends ideas used by PR and PPR to distinguish between different predicates in KGs while being aware of importance scores and graph topology. Still, we observe that there is much room for improvement, as evidenced by the performance of existing methods on real-world KGs in Figure 2. So far, existing techniques have approached this problem in a non-trainable framework that is based on a fixed model structure determined by their prior assumptions on the propagation of node importance, and involve no learnable parameters that are optimized based on the ground truth.

In this paper, we explore a new family of solutions for the task of predicting node importance in KGs, namely, regularized supervised machine learning algorithms. Our goal is to develop a more flexible supervised approach that learns from ground truth, and makes use of additional information in KGs. Among several supervised algorithms we explore, we focus on graph neural networks (GNNs). Recently, GNNs have received increasing interests, and achieved state-of-the-art performance on node and graph classification tasks across data drawn from several domains [6, 10, 14, 22, 25]. Designed to learn from graph-structured data, and based on neighborhood aggregation framework, GNNs have the potential to make further improvements over earlier approaches. However, existing GNNs have focused on graph representation learning via embedding aggregation, and have not been designed to tackle challenges that arise with supervised estimation of node importance in KGs. Challenges include modeling the relationship between the importance of neighboring nodes, accurate estimation that generalizes across different types of entities, and incorporating prior assumptions on node importance that aid model prediction, which are not addressed at the same time by existing supervised techniques.

We present GENI, a GNN for Estimating Node Importance in KGs. GENI applies an attentive GNN for predicate-aware score aggregation to capture relations between the importance of nodes and their neighbors. GENI also allows flexible score adjustment according to node *centrality*, which captures connectivity of a node in terms of graph topology. Our main contributions are as follows.

- We explore regularized supervised machine learning algorithms for estimating node importance in KGs, as opposed to non-trainable solutions where existing approaches belong.
- We present GENI, a GNN-based method designed to address the challenges involved with supervised estimation of node importance in KGs.
- We provide empirical evidence and an analysis of GENI using real-world KGs. Figure 2 shows that GENI outperforms the state of the art by 5%-17% percentage points on real KGs.

The rest of this paper is organized as follows. We present preliminaries in Section 2, and describe our method in Section 3. After providing experimental results on real KGs in Section 4, we review related works in Section 5, and conclude in Section 6.

2 PRELIMINARIES

2.1 Problem Definition

A *knowledge graph* (KG) is a graph $G = (V, E = \{E_1, E_2, \ldots, E_P\})$ that represents multi-relational data where nodes V and edges E correspond to entities and their relationships, respectively; P is the number of types of edges (predicates); and E_p denotes a set of edges of type $p \in \{1, \ldots, P\}$. In KGs, there are often many types of predicates (i.e., $P \gg 1$) between nodes of possibly different types (e.g., movie, actor, and director nodes), whereas in traditional graphs, nodes are connected by just one type of edges (i.e., $P = 1$).

An *importance score* $s \in \mathbb{R}_{\geq 0}$ is a non-negative real number that represents the significance or popularity of a node. For example, the total gross of a movie can be used as an importance score for a movie KG, and the number of pageviews of an entity can be used in a more generic KG such as Freebase [2]. We assume a single set of importance scores, so the scores can compare with each other to reflect importance.

We now define the node importance estimation problem.

Definition 2.1 (Node Importance Estimation). Given a KG $G = (V, E = \{E_1, E_2, \ldots, E_P\})$ and importance scores $\{s\}$ for a subset $V_s \subseteq V$ of nodes, learn a function $S : V \longrightarrow [0, \infty)$ that estimates the importance score of every node in KG.

Figure 1 shows an example KG on movies and related entities with importance scores given in advance for some movies. We approach the importance estimation problem by developing a supervised framework learning a function that maps any node in KG to its score, such that the estimation reflects its true importance as closely as possible.

Table 1: Comparison of methods for estimating node importance. *Neighborhood*: Neighborhood awareness. *Predicate*: Making use of predicates. *Centrality*: Centrality awareness. *Input Score*: Utilizing input importance scores. *Flexibility*: Flexible adaptation.

	GENI	**HAR** [17]	**PPR** [11]	**PR** [18]
Neighborhood	✓	✓	✓	✓
Predicate	✓	✓		
Centrality	✓	✓	✓	✓
Input Score	✓	✓	✓	
Flexibility	✓			

Note that even when importance scores are provided for only one type of nodes (e.g., movies), we aim to do estimation for all types of nodes (e.g,. directors, actors, etc.).

Definition 2.2 (In-Domain and Out-Of-Domain Estimation). Given importance scores for some nodes $V_s \subseteq V$ of type $\mathcal{T}$ (e.g., movies), predicting the importance of nodes of type $\mathcal{T}$ is called an *"in-domain"* estimation, and importance estimation for those nodes whose type is not $\mathcal{T}$ is called an *"out-of-domain"* estimation.

As available importance scores are often limited in terms of numbers and types, developing a method that generalizes well for both classes of estimation is an important challenge for supervised node importance estimation.

2.2 Desiderata for Modeling Node Importance in KGs

Based on our discussion on prior approaches (PR, PPR, and HAR), we present the desiderata that have guided the development of our method for tackling node importance estimation problem. Table 1 summarizes GENI and existing methods in terms of these desiderata.

Neighborhood Awareness. In a graph, a node is connected to other nodes, except for the special case of isolated nodes. As neighboring entities interact with each other, and they tend to share common characteristics (network homophily), neighborhoods should be taken into account when node importance is modeled.

Making Use of Predicates. KGs consist of multiple types of predicates. Under the assumption that different predicates could play a different role in determining node importance, models should make predictions using information from predicates.

Centrality Awareness. Without any other information, it is reasonable to assume that highly central nodes are more important than less central ones. Therefore, scores need to be estimated in consideration of node centrality, capturing connectivity of a node.

Utilizing Input Importance Scores. In addition to graph topology, input importance scores provide valuable information to infer relationships between nodes and their importance. Thus, models should tap into both the graph structure and input scores for more accurate prediction.

Flexible Adaptation. Our assumption regarding node importance such as the one on centrality may not conform to the real distribution of input scores over KGs. Also, we do not limit models to a specific type of input scores. On the other hand, models can be provided with input scores that possess different characteristics. It is thus critical that a model can flexibly adapt to the importance that input scores reflect.

2.3 Graph Neural Networks

In this section, we present a generic definition of graph neural networks (GNNs). GNNs are mainly based on neighborhood aggregation architecture [8, 10, 14, 22, 25]. In a GNN with L layers, its ℓ-th layer ($\ell = 1, \ldots, L$) receives a feature vector $\vec{h}_i^{\ell-1}$ for each node i from the $(\ell-1)$-th layer (where $\vec{h}_i^0$ is an input node feature $\vec{z}_i$), and updates it by aggregating feature vectors from the neighborhood $\mathcal{N}(i)$ of node i, possibly using a different weight $w_{i,j}^{\ell}$ for neighbor j. As updated feature vectors become the input to the $(\ell+1)$-th layer, repeated aggregation procedure through L layers in principle captures L-th order neighbors in learning a node's representation. This process of learning representation $\vec{h}_i^{\ell}$ of node i by ℓ-th layer is commonly expressed as [10, 24, 25]:

$$\vec{h}_{\mathcal{N}(i)}^{\ell} \leftarrow \textsc{Transform}^{\ell}\left(\textsc{Aggregate}\left(\left\{\left(\vec{h}_j^{\ell-1}, w_{i,j}^{\ell}\right) \mid j \in \mathcal{N}(i)\right\}\right)\right) \tag{1}$$

$$\vec{h}_i^{\ell} \leftarrow \textsc{Combine}\left(\vec{h}_i^{\ell-1}, \vec{h}_{\mathcal{N}(i)}^{\ell}\right) \tag{2}$$

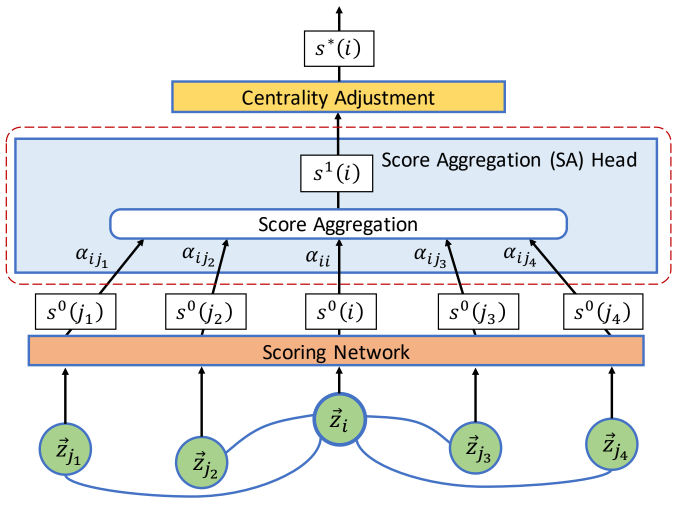
where Aggregate is an aggregation function defined by the model (e.g., averaging or max-pooling operation); Transform is a model-specific function that performs a (non-linear) transformation of node embeddings via parameters in ℓ-th layer shared by all nodes (e.g., multiplication with a shared weight matrix $\mathbf{W}^{\ell}$ followed by some non-linearity $\sigma(\cdot)$); Combine is a function that merges the aggregated neighborhood representation with the node's representation (e.g., concatenation).

3 METHOD

Effective estimation of node importance in KGs involves addressing the requirements presented in Section 2.2. As a supervised learning method, the GNN framework naturally allows us to *utilize input importance scores* to train a model with *flexible adaptation*. Its propagation mechanism also allows us to be *neighborhood aware*. In this section, we present GENI, which further enhances the model in three ways.

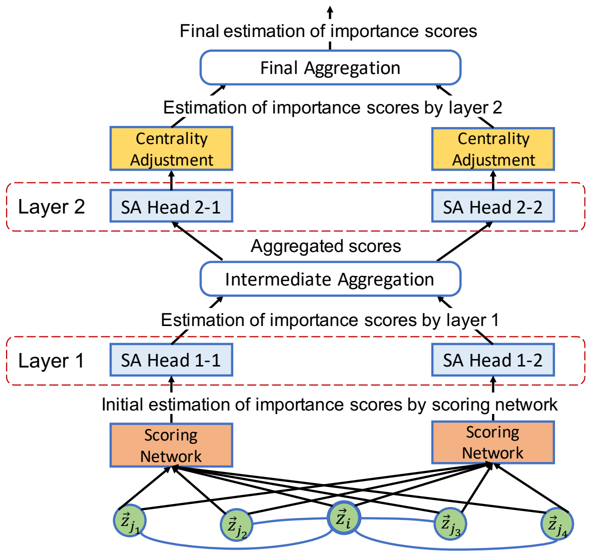
- *Neighborhood Importance Awareness*: GNN normally propagates information between neighbors through node embedding. This is to model the assumption that an entity and its neighbors affect each other, and thus the representation of an entity can be better represented in terms of the representation of its neighbors. In the context of node importance estimation, neighboring importance scores play a major role on the importance of a node, whereas other neighboring features may have little effect, if any. We thus directly aggregate importance scores from neighbors (Section 3.1), and show empirically that it outperforms embedding propagation (Section 4.4).
- *Making Use of Predicates*: We design predicate-aware attention mechanism that models how predicates affect the importance of connected entities (Section 3.2).
- *Centrality Awareness*: We apply centrality adjustment to incorporate node centrality into the estimation (Section 3.3).

An overview of GENI is provided in Figure 3. In Sections 3.1 to 3.3, we describe the three main enhancements using the basic building blocks of GENI shown in Figure 3(a). Then we discuss an extension to a general architecture in Section 3.4. Table 2 provides the definition of symbols used in this paper.



$s^*(i)$
Centrality Adjustment
$s^1(i)$
Score Aggregation (SA) Head
Score Aggregation
α_{ij_1} α_{ij_2} α_{ii} α_{ij_3} α_{ij_4}
$s^0(j_1)$ $s^0(j_2)$ $s^0(i)$ $s^0(j_3)$ $s^0(j_4)$
Scoring Network
$\vec{z}_{j_1}$ $\vec{z}_{j_2}$ $\vec{z}_i$ $\vec{z}_{j_3}$ $\vec{z}_{j_4}$

(a)



Final estimation of importance scores
Final Aggregation
Estimation of importance scores by layer 2
Centrality Adjustment
Centrality Adjustment
Layer 2
SA Head 2-1
SA Head 2-2
Aggregated scores
Intermediate Aggregation
Estimation of importance scores by layer 1
Layer 1
SA Head 1-1
SA Head 1-2
Initial estimation of importance scores by scoring network
Scoring Network
Scoring Network
$\vec{z}_{j_1}$ $\vec{z}_{j_2}$ $\vec{z}_i$ $\vec{z}_{j_3}$ $\vec{z}_{j_4}$

(b)

Figure 3: Description of node importance estimation by GENI. (a): Estimation of the importance of node i based on the embeddings of node i and its neighbors $j_1, \ldots, j_4$ (connected by blue edges). The final estimation $s^*(i)$ is produced via three components of GENI shown in colored boxes, which are described in Sections 3.1 to 3.3. (b): An illustration of the proposed model that consists of two layers, each of which contains two score aggregation heads. Note that the model can consist of different numbers of layers, and each layer can also have different numbers of score aggregation heads. A discussion on the extension of the basic model in (a) to a more comprehensive architecture in (b) is given in Section 3.4.

Table 2: Table of symbols.

Symbol	Definition
V_s	set of nodes with known importance scores
$\vec{z}_i$	real-valued feature vector of node i
$\mathcal{N}(i)$	neighbors of node i
L	total number of score aggregation (SA) layers
ℓ	index for an SA layer
H^ℓ	number of SA heads in ℓ-th layer
p_{ij}^m	predicate of m-th edge between nodes i and j
$\phi(e)$	learnable embedding of predicate e
σ_a, σ_s	non-linearities for attention computation and score estimation
$s_h^\ell(i)$	estimated score of node i by h-th SA head in ℓ-th layer
$s^*(i)$	centrality-adjusted score estimation of node i
$\|\|$	concatenation operator
$d(i)$	in-degree of node i
$c(i)$	centrality score of node i
$c_h^*(i)$	centrality score of node i scaled and shifted by h-th SA head
γ_h, β_h	learnable scale and shift parameters used by h-th SA head
$\vec{a}_{h,\ell}$	learnable parameter vector to compute $\alpha_{ij}^{h,\ell}$ by h-th SA head in ℓ-th layer
$\alpha_{ij}^{h,\ell}$	node i's attention on node j computed with h-th SA head in ℓ-th layer
$g(i)$	known importance score of node i

3.1 Score Aggregation

To directly model the relationship between the importance of neighboring nodes, we propose a score aggregation framework, rather than embedding aggregation. Specifically, in Equations (1) and (2), we replace the hidden embedding $\vec{h}_j^{\ell-1}$ of node j with its score estimation $s^{\ell-1}(j)$ and combine them as follows:

$$s^\ell(i) = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^\ell \, s^{\ell-1}(j) \tag{3}$$

where $\mathcal{N}(i)$ denotes the neighbors of node i, which will be a set of the first-order neighbors of node i in our experiments. Here, α_{ij}^ℓ is a learnable weight between nodes i and j for the ℓ-th layer ($\ell = 1, \ldots, L$). We train it via a shared attention mechanism which is computed by a pre-defined model with shared parameters and predicate embeddings, as we explain soon. In other words, GENI computes the aggregated score $s^\ell(i)$ by performing a weighted aggregation of intermediate scores from node i and its neighbors. Note that GENI does not apply Transform$^\ell$ function after aggregation as in Equation (1), since GENI aggregates scores. Propagating scores instead of node embeddings has the additional benefit of reducing the number of model parameters.

To compute the initial estimation $s^0(i)$, GENI uses input node features. In the simplest case, they can be one-hot vectors that represent each node. More generally, they are real-valued vectors representing the nodes, which are extracted manually based on domain knowledge, or generated with methods for learning node embeddings. Let $\vec{z}_i$ be the input feature vector of node i. Then GENI computes the initial score of i as

$$s^0(i) = \text{ScoringNetwork}(\vec{z}_i) \tag{4}$$

where ScoringNetwork can be any neural network that takes in a node feature vector and returns an estimation of its importance. We used a simple fully-connected neural network for our experiments.

3.2 Predicate-Aware Attention Mechanism

Inspired by recent work that showcased successful application of attention mechanism, we employ a predicate-aware attention mechanism that attends over the neighbor's intermediate scores.

Our attention considers two factors. First, we consider the predicate between the nodes because different predicates can play different roles for score propagation. For example, even though a movie may be released in a popular (i.e., important) country, the movie itself may not be popular; on the other hand, a movie directed by a famous (i.e., important) director is more likely to be popular. Second, we consider the neighboring score itself in deciding the attention. A director who directed a few famous (i.e., important) movies is likely to be important; the fact that he also directed some not-so-famous movies in his life is less likely to make him unimportant.

GENI incorporates predicates into attention computation by using shared predicate embeddings; i.e., each predicate is represented by a feature vector of predefined length, and this representation is shared by nodes across all layers. Further, predicate embeddings are learned so as to maximize the predictive performance of the model in a flexible fashion. Note that in KGs, there could be multiple edges of different types between two nodes (e.g., see Figure 1). We use p_{ij}^m to denote the predicate of m-th edge between nodes i and j, and $\phi(\cdot)$ to denote a mapping from a predicate to its embedding.

In GENI, we use a simple, shared self-attention mechanism, which is a single layer feedforward neural network parameterized by the weight vector $\vec{a}$. Relation between the intermediate scores of two nodes i and j, and the role an in-between predicate plays are captured by the attentional layer that takes in the concatenation of all relevant information. Outputs from the attentional layer are first transformed by non-linearity $\sigma(\cdot)$, and then normalized via the softmax function. Formally, GENI computes the attention α_{ij}^ℓ of node i on node j for ℓ-th layer as:

$$\alpha_{ij}^\ell = \frac{\exp\left(\sigma_a\left(\sum_m \vec{a}_\ell^\top [s^\ell(i) || \phi(p_{ij}^m) || s^\ell(j)]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\sigma_a\left(\sum_m \vec{a}_\ell^\top [s^\ell(i) || \phi(p_{ik}^m) || s^\ell(k)]\right)\right)} \tag{5}$$

where σ_a is a non-linearity, $\vec{a}_\ell$ is a weight vector for ℓ-th layer, and $||$ is a concatenation operator.

3.3 Centrality Adjustment

Existing methods such as PR, PPR, and HAR make a common assumption that the importance of a node positively correlates with its centrality in the graph. In the context of KGs, it is also natural to assume that more central nodes would be more important than less central ones, unless the given importance scores present contradictory evidence. Making use of this prior knowledge becomes especially beneficial in cases where we are given a small number of importance scores compared to the total number of entities, and in cases where the importance scores are given for entities of a specific type out of the many types in KG.

Given that the in-degree $d(i)$ of node i is a common proxy for its centrality and popularity, we define the initial centrality $c(i)$ of node i to be

$$c(i) = \log(d(i) + \epsilon) \tag{6}$$

where ϵ is a small positive constant.

While node centrality provides useful information on the importance of a node, strictly adhering to the node centrality could have a detrimental effect on model prediction. We need flexibility to account for the possible discrepancy between the node's centrality in a given KG and the provided input importance score of the node. To this end, we use a scaled and shifted centrality $c^*(i)$ as our notion of node centrality:

$$c^*(i) = \gamma \cdot c(i) + \beta \tag{7}$$

where γ and β are learnable parameters for scaling and shifting. As we show in Section 4.5, this flexibility allows better performance when in-degree is not the best proxy of centrality.

To compute the final score, we apply centrality adjustment to the score estimation $s^L(i)$ from the last layer, and apply a non-linearity σ_s as follows:

$$s^*(i) = \sigma_s\left(c^*(i) \cdot s^L(i)\right) \tag{8}$$

3.4 Model Architecture

The simple architecture depicted in Figure 3(a) consists of a scoring network and a single score aggregation (SA) layer (i.e., $L = 1$), followed by a centrality adjustment component. Figure 3(b) extends it to a more general architecture in two ways. First, we extend the framework to contain multiple SA layers; that is, $L > 1$. As a single SA layer aggregates the scores of direct neighbors, stacking multiple SA layers enables aggregating scores from a larger neighborhood. Second, we design each SA layer to contain a variable number of SA heads, which perform score aggregation and attention computation independently of each other. Empirically, we find using multiple SA heads to be helpful for the model performance and the stability of optimization procedure (Section 4.5).

Let h be an index of an SA head, and H^ℓ be the number of SA heads in ℓ-th layer. We define $s_h'^{\ell-1}(i)$ to be node i's score that is estimated by $(\ell-1)$-th layer, and fed into h-th SA head in ℓ-th (i.e., the next) layer, which in turn produces an aggregation $s_h^\ell(i)$ of these scores:

$$s_h^\ell(i) = \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij}^{h,\ell}\; s_h'^{\ell-1}(j) \tag{9}$$

where $\alpha_{ij}^{h,\ell}$ is the attention coefficient between nodes i and j computed by SA head h in layer ℓ.

In the first SA layer, each SA head h receives input scores from a separate scoring network $\textsc{ScoringNetwork}_h$, which provides the initial estimation $s_h^0(i)$ of node importance. For the following layers, output from the previous SA layer becomes the input estimation. Since in ℓ-th ($\ell \geq 1$) SA layer, H^ℓ SA heads independently produce H^ℓ score estimations in total, we perform an aggregation of these scores by averaging, which is provided to the next layer. That is,

$$s_h'^\ell(i) = \begin{cases} \textsc{ScoringNetwork}_h(\vec{z}_i) & \text{if } \ell = 0 \\ \textsc{Average}\left(\left\{s_h^\ell(i) \mid h = 1, \ldots, H^\ell\right\}\right) & \text{if } \ell \geq 1 \end{cases} \tag{10}$$

Multiple SA heads in ℓ-th layer compute attention between neighboring nodes in the same way as in Equation (5), yet independently of each other using its own parameters $\vec{a}_{h,\ell}$:

$$\alpha_{ij}^{h,\ell} = \frac{\exp\left(\sigma_a\left(\sum_m \vec{a}_{h,\ell}^\top [s_h'^{\ell-1}(i) || \phi(p_{ij}^m) || s_h'^{\ell-1}(j)]\right)\right)}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp\left(\sigma_a\left(\sum_m \vec{a}_{h,\ell}^\top [s_h'^{\ell-1}(i) || \phi(p_{ik}^m) || s_h'^{\ell-1}(k)]\right)\right)} \tag{11}$$

Centrality adjustment is applied to the output from the final SA layer. In order to enable independent scaling and shifting by each SA head, separate parameters γ_h and β_h are used for each head h. Then centrality adjustment by h-th SA head in the final layer is:

$$c_h^*(i) = \gamma_h \cdot c(i) + \beta_h \tag{12}$$

With H^L SA heads in the final L-th layer, we perform additional aggregation of centrality-adjusted scores by averaging, and apply a non-linearity σ_s, obtaining the final estimation $s^*(i)$:

$$s^*(i) = \sigma_s \left(\text{Average} \left(\left\{ c_h^*(i) \cdot s_h^L(i) \mid h = 1, \ldots, H^L \right\} \right) \right) \tag{13}$$

3.5 Model Training

In order to predict node importance with input importance scores known for a subset of nodes $V_s \subseteq V$, we train GENI using mean squared error between the given importance score $g(i)$ and the model estimation $s^*(i)$ for node $i \in V_s$; thus, the loss function is

$$\frac{1}{|V_s|} \sum_{i \in V_s} \left(s^*(i) - g(i) \right)^2 \tag{14}$$

Note that ScoringNetwork is trained jointly with the rest of GENI. To avoid overfitting, we apply weight decay with an early stopping criterion based on the model performance on validation entities.

4 EXPERIMENTS

In this section, we aim to answer the following questions.

- How do GENI and baselines perform on real-world KGs with different characteristics? In particular, how well do methods perform in- and out-of-domain estimation (Definition 2.2)?
- How do the components of GENI, such as centrality adjustment, and different parameter values affect its estimation?

We describe datasets, baselines, and evaluation plans in Sections 4.1 to 4.3, and answer the above questions in Sections 4.4 and 4.5.

4.1 Datasets

In our experiments, we use four real-world KGs with different characteristics. Here we introduce these KGs along with the importance scores used for in- and out-of-domain (OOD) evaluations (see Definition 2.2). Summaries of the datasets (such as the number of nodes, edges, and predicates) are given in Table 3. More details such as data sources and how they are constructed can be found in Appendix A.

fb15k is a subset of Freebase, which is a large collaborative knowledge base containing general facts, and has been widely used for research and practical applications [2, 3]. fb15k has a much larger number of predicates and a higher density than other KGs we evaluated. For each entity, we use the number of pageviews for the corresponding Wikipedia page as its score. Note that we do not perform OOD evaluation for fb15k since importance scores for fb15k apply to all types of entities.

music10k is a music KG sampled from the Million Song Dataset[1], which includes information about songs such as the primary artist and the album the song belongs to. The dataset provides two types of popularity scores called "song hotttnesss" and "artist hotttnesss" computed by the Echo Nest platform by considering data from many sources such as mentions on the web, play counts, etc[2]. We use "song hotttnesss" as input importance scores, and "artist hotttnesss" for OOD performance evaluation.

tmdb5k is a movie KG derived from the TMDb 5000 movie dataset[3]. It contains movies and related entities such as movie genres, companies, countries, crews, and casts. We use the "popularity" information for movies as importance scores, which is provided by the original dataset. For OOD evaluation, we use a ranking of top-200 highest grossing directors[4]. Worldwide box office grosses given in the ranking are used as importance scores for directors.

imdb is a movie KG created from the public IMDb dataset, which includes information such as movies, genres, directors, casts, and crews. imdb is the largest KG among those we evaluate, with 12.6× as many nodes as tmdb5k. IMDb dataset provides the number of votes a movie received, which we use as importance scores. For OOD evaluation, we use the same director ranking used for tmdb5k.

4.2 Baselines

Methods for node importance estimation in KGs can be classified into two families of algorithms.

Non-Trainable Approaches. Previously developed methods mostly belong to this category. We evaluate the following methods:

- PageRank (PR) [18]
- Personalized PageRank (PPR) [11]
- HAR [17]

Supervised Approaches. We explore the performance of representative supervised algorithms on node importance estimation:

- Linear regression (LR): an ordinary least squares algorithm.
- Random forests (RF): a random forest regression model.
- Neural networks (NN): a fully-connected neural network.
- Graph attention networks (GAT) [22]: This is a GNN model reviewed in Section 2.3. We add a final layer that takes the node embedding and outputs the importance score of a node.

All these methods and GENI use the same data (node features and input importance scores). In our experiments, node features are generated using node2vec [9]. Depending on the type of KGs, other types of node features, such as bag-of-words representation, can also be used. Note that the graph structure is explicitly used only by GAT, although other supervised baselines make an implicit use of it when node features encode graph structural information.

We will denote each method by the name in parentheses. Experimental settings for baselines and GENI are provided in Appendix B.

4.3 Performance Evaluation

We evaluate methods based on their in- and out-of-domain (OOD) performance. We performed 5-fold cross validation, and report the average and standard deviation of the following metrics on ranking quality and correlation: normalized discounted cumulative gain and Spearman correlation coefficient. Higher values are better for all metrics. We now provide their formal definitions.

Normalized discounted cumulative gain (NDCG) is a measure of ranking quality. Given a list of nodes ranked by predicted scores, and their graded relevance values (which are non-negative, real-valued ground truth scores in our setting), discounted cumulative gain at position k ($DCG@k$) is defined as:

$$DCG@k = \sum_{i=1}^{k} \frac{r_i}{\log_2(i+1)} \tag{15}$$

where r_i denotes the graded relevance of the node at position i. Note that due to the logarithmic reduction factor, the gain r_i of each node is penalized at lower ranks. Consider an ideal DCG at rank position k ($IDCG@k$) which is obtained by an ideal ordering of

[1]https://labrosa.ee.columbia.edu/millionsong/
[2]https://musicmachinery.com/tag/hotttnesss/
[3]https://www.kaggle.com/tmdb/tmdb-movie-metadata
[4]https://www.the-numbers.com/box-office-star-records/worldwide/lifetime-specific-technical-role/director

Table 3: Real-world KGs. See Section 4.1 and Appendix A for details. SCC: Strongly connected component. OOD: Out-of-domain.

Name	# Nodes	# Edges	# Predicates	# SCCs.	Input Score Type	# Nodes w/ Scores	Data for OOD Evaluation
FB15K	14,951	592,213	1,345	9	# Pageviews	14,108 (94%)	N/A
MUSIC10K	24,830	71,846	10	130	Song hotttnesss	4,214 (17%)	Artist hotttnesss
TMDB5K	123,906	532,058	22	15	Movie popularity	4,803 (4%)	Director ranking
IMDB	1,567,045	14,067,776	28	1	# Votes for movies	215,769 (14%)	Director ranking

nodes based on their relevance scores. Normalized DCG at position k ($NDCG@k$) is then computed as:

$$NDCG@k = \frac{DCG@k}{IDCG@k} \tag{16}$$

Our motivation for using $NDCG@k$ is to test the quality of ranking for the top k entities.

Spearman correlation coefficient (SPEARMAN) measures the rank correlation between the ground truth scores $\vec{g}$ and predicted scores $\vec{s}$; that is, the strength and direction of the monotonic relationship between the rank values of $\vec{g}$ and $\vec{s}$. Converting $\vec{g}$ and $\vec{s}$ into ranks $\vec{g}_r$ and $\vec{s}_r$, respectively, Spearman correlation coefficient is computed as:

$$Spearman = \frac{\sum_i (g_{r\,i} - \overline{g}_r)(s_{r\,i} - \overline{s}_r)}{\sqrt{\sum_i (g_{r\,i} - \overline{g}_r)^2}\sqrt{\sum_i (s_{r\,i} - \overline{s}_r)^2}} \tag{17}$$

where $\overline{g}_r$ and $\overline{s}_r$ are the mean of $\vec{g}_r$ and $\vec{s}_r$.

For in-domain evaluation, we use NDCG@100 and SPEARMAN as they complement each other: NDCG@100 looks at the top-100 predictions, and SPEARMAN considers the ranking of all entities with known scores. For NDCG, we also tried different cut-off thresholds and observed similar results. Note that we often have a small volume of data for OOD evaluation. For example, for TMDB5K and IMDB, we used a ranking of 200 directors with known scores, while TMDB5K and IMDB have 2,578 and 287,739 directors, respectively. Thus SPEARMAN is not suitable for OOD evaluation as it considers only those small number of entities in the ranking, and ignores all others, even if they are predicted to be highly important; thus, for OOD evaluation, we report NDCG@100 and NDCG@2000.

Additionally, we report regression performance in Appendix C.2.

4.4 Importance Estimation on Real-World Data

We evaluate GENI and baselines in terms of in- and out-of-domain (OOD) predictive performance.

4.4.1 In-Domain Prediction. Table 4 summarizes in-domain prediction performance. GENI outperforms all baselines on four datasets in terms of both NDCG@100 and SPEARMAN. It is noteworthy that supervised approaches generally perform better in-domain prediction than non-trainable ones, especially on FB15K and IMDB, which are more complex and larger than the other two. It demonstrates the applicability of supervised models to our problem. On all KGs except MUSIC10K, GAT outperforms other supervised baselines, which use the same node features but do not explicitly take the graph network structure into account. This shows the benefit of directly utilizing network connectivity. By modeling the relation between scores of neighboring entities, GENI achieves further performance improvement over GAT. Among non-trainable baselines, HAR often performs worse than PR and PPR, which suggests that considering predicates could hurt performance if predicate weight adjustment is not done properly.

4.4.2 Out-Of-Domain Prediction. Table 5 summarizes OOD prediction results. GENI achieves the best results for all KGs in terms of both NDCG@100 and NDCG@2000. In contrast to in-domain prediction where supervised baselines generally outperform non-trainable ones, we observe that non-trainable methods achieve higher OOD results than supervised baselines on MUSIC10K and TMDB5K. In these KGs, only about 4,000 entities have known scores. Given scarce ground truth, non-trainable baselines could perform better by relying on a prior assumption on the propagation of node importance. Further, note that the difference between non-trainable and supervised baselines is more drastic on TMDB5K where the proportion of nodes with scores is the smallest (4%). On the other hand, on IMDB, which is our largest KG with the greatest number of ground truth, supervised baselines mostly outperform non-trainable methods. In particular, none of the top-100 directors in IMDB predicted by PR and PPR belong to the ground truth director ranking. With 14% of nodes in IMDB associated with known scores, supervised methods learn to generalize better for OOD prediction. Although neighborhood aware, GAT is not better than other supervised baselines. By applying centrality adjustment, GENI achieves superior performance to both classes of baselines regardless of the number of available known scores.

4.5 Analysis of GENI

4.5.1 Effect of Considering Predicates. To see how the consideration of predicates affects model performance, we run GENI on FB15K, which has the largest number of predicates, and report NDCG@100 and SPEARMAN when a single embedding is used for all predicates (denoted by "shared embedding") vs. when each predicate uses its own embedding (denoted by "distinct embedding"). Note that using "shared embedding", GENI loses the ability to distinguish between different predicates. In the results given in Table 6, we observe that NDCG@100 and SPEARMAN are increased by 3.6% and 12.7%, respectively, when a dedicated embedding is used for each predicate. This shows that GENI successfully makes use of predicates for modeling the relation between node importance; this is especially crucial in KGs such as FB15K that consist of a large number of predicates.

4.5.2 Flexibility for Centrality Adjustment. In Equation (7), we perform scaling and shifting of $c(i)$ for flexible centrality adjustment (CA). Here we evaluate the model with fixed CA without scaling and shifting where the final estimation $s^*(i) = \sigma_s(c(i) \cdot s^L(i))$. In Table 7, we report the performance of GENI on FB15K and TMDB5K obtained with fixed and flexible CA while all other parameters were identical. When node centrality strongly correlates with input scores, fixed CA obtains similar results to flexible CA. This is reflected on the result of TMDB5K dataset, where PR and log in-degree baseline (LID), which estimates node importance as the log of its in-degree, both estimate node importance close to the input scores.

Table 4: In-domain prediction results on real-world datasets. GENI consistently outperforms all baselines. Numbers after ± symbol are standard deviation from 5-fold cross validation. Best results are in bold, and second best results are underlined.

Method	FB15K		MUSIC10K		TMDB5K		IMDB	
	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN
PR	0.8354 ± 0.016	0.3515 ± 0.015	0.5510 ± 0.021	−0.0926 ± 0.034	0.8293 ± 0.026	0.5901 ± 0.011	0.7847 ± 0.048	0.0881 ± 0.004
PPR	0.8377 ± 0.015	0.3667 ± 0.015	0.7768 ± 0.009	0.3524 ± 0.046	0.8584 ± 0.013	0.7385 ± 0.010	0.7847 ± 0.048	0.0881 ± 0.004
HAR	0.8261 ± 0.005	0.2020 ± 0.012	0.5727 ± 0.017	0.0324 ± 0.044	0.8141 ± 0.021	0.4976 ± 0.014	0.7952 ± 0.036	0.1318 ± 0.005
LR	0.8750 ± 0.005	0.4626 ± 0.019	0.7301 ± 0.023	0.3069 ± 0.032	0.8743 ± 0.015	0.6881 ± 0.013	0.7365 ± 0.009	0.5013 ± 0.002
RF	0.8734 ± 0.005	0.5122 ± 0.019	0.8129 ± 0.012	0.4577 ± 0.012	0.8503 ± 0.016	0.5959 ± 0.022	0.7651 ± 0.010	0.4753 ± 0.005
NN	0.9003 ± 0.005	0.6031 ± 0.012	0.8015 ± 0.017	0.4491 ± 0.027	0.8715 ± 0.006	0.7009 ± 0.009	0.8850 ± 0.016	0.5120 ± 0.008
GAT	0.9205 ± 0.009	0.7054 ± 0.013	0.7666 ± 0.016	0.4276 ± 0.023	0.8865 ± 0.011	0.7180 ± 0.010	0.9110 ± 0.011	0.7060 ± 0.007
GENI	**0.9385 ± 0.004**	**0.7772 ± 0.006**	**0.8224 ± 0.018**	**0.4783 ± 0.009**	**0.9051 ± 0.005**	**0.7796 ± 0.009**	**0.9318 ± 0.005**	**0.7387 ± 0.002**

Table 5: Out-of-domain prediction results on real-world datasets. GENI consistently outperforms all baselines. Numbers after ± symbol are standard deviation from 5-fold cross validation. Best results are in bold, and second best results are underlined.

Method	MUSIC10K		TMDB5K		IMDB	
	NDCG@100	NDCG@2000	NDCG@100	NDCG@2000	NDCG@100	NDCG@2000
PR	0.6520 ± 0.000	0.8779 ± 0.000	0.8337 ± 0.000	0.8079 ± 0.000	0.0000 ± 0.000	0.1599 ± 0.000
PPR	0.7324 ± 0.006	0.9118 ± 0.002	0.8060 ± 0.041	0.7819 ± 0.022	0.0000 ± 0.000	0.1599 ± 0.000
HAR	0.7113 ± 0.004	0.8982 ± 0.001	0.8913 ± 0.010	0.8563 ± 0.007	0.2551 ± 0.019	0.3272 ± 0.005
LR	0.6644 ± 0.006	0.8667 ± 0.001	0.4990 ± 0.013	0.5984 ± 0.002	0.3064 ± 0.007	0.2755 ± 0.003
RF	0.6898 ± 0.022	0.8796 ± 0.003	0.5993 ± 0.040	0.6236 ± 0.005	0.4066 ± 0.145	0.3719 ± 0.040
NN	0.6981 ± 0.017	0.8836 ± 0.005	0.5675 ± 0.023	0.6172 ± 0.009	0.2158 ± 0.035	0.3105 ± 0.019
GAT	0.6909 ± 0.009	0.8834 ± 0.003	0.5349 ± 0.016	0.5999 ± 0.007	0.3858 ± 0.065	0.4209 ± 0.016
GENI	**0.7964 ± 0.007**	**0.9121 ± 0.002**	**0.9078 ± 0.004**	**0.8776 ± 0.002**	**0.4519 ± 0.051**	**0.4962 ± 0.025**

Table 6: Performance of GENI on FB15K when a single embedding is used for all predicates (shared embedding) vs. when each predicate uses its own embedding (distinct embedding).

Metric	Shared Embedding	Distinct Embedding
NDCG@100	0.9062 ± 0.008	**0.9385 ± 0.004**
SPEARMAN	0.6894 ± 0.007	**0.7772 ± 0.006**

Table 7: Performance of PR, log in-degree baseline, and GENI with fixed and flexible centrality adjustment (CA) on FB15K and TMDB5K.

Method	FB15K		TMDB5K	
	NDCG@100	SPEARMAN	NDCG@100	SPEARMAN
PR	0.835 ± 0.02	0.352 ± 0.02	0.829 ± 0.03	0.590 ± 0.01
Log In-Degree	0.810 ± 0.02	0.300 ± 0.03	0.852 ± 0.02	0.685 ± 0.02
GENI-Fixed CA	0.868 ± 0.01	0.613 ± 0.01	0.899 ± 0.01	0.771 ± 0.01
GENI-Flexible CA	**0.938 ± 0.00**	**0.777 ± 0.01**	**0.905 ± 0.01**	**0.780 ± 0.01**

On the other hand, when node centrality is not in good agreement with input scores, as demonstrated by the poor performance of PR and LID as on FB15K, flexible CA performs much better than fixed CA (8% higher NDCG@100, and 27% higher SPEARMAN on FB15K).

4.5.3 Parameter Sensitivity. We evaluate the parameter sensitivity of GENI by measuring performance on FB15K varying one of the following parameters while fixing others to their default values (shown in parentheses): number of score aggregation (SA) layers (1), number of SA heads in each SA layer (1), dimension of predicate embedding (10), and number of hidden layers in scoring networks (1 layer with 48 units). Results presented in Figure 4 shows that the model performance tends to improve as we use a greater number of SA layers and SA heads. For example, SPEARMAN increases from 0.72 to 0.77 as the number of SA heads is increased from 1 to 5. Using more hidden layers for scoring networks also tends to boost performance, although exceptions are observed. Increasing the dimension of predicate embedding beyond an appropriate value negatively affects the model performance, although GENI still achieves high SPEARMAN compared to baselines.

5 RELATED WORK

Node Importance Estimation. Many approaches have been developed for node importance estimation [11, 13, 15, 17, 18, 20]. PageRank (PR) [18] is based on the random surfer model where an imaginary surfer randomly moves to a neighboring node with probability d, or teleports to any other node randomly with probability $1 - d$. PR predicts the node importance to be the limiting probability of the random surfer being at each node. Accordingly, PR scores are determined only by the graph structure, and unaware of input importance scores. Personalized PageRank (PPR) [11] deals with this limitation by biasing the random walk to teleport to a set of nodes relevant to some specific topic, or alternatively, nodes with known importance scores. Random walk with restart (RWR) [13, 20] is a closely related method that addresses a special case of PPR where teleporting is restricted to a single node. PPR and RWR, however,

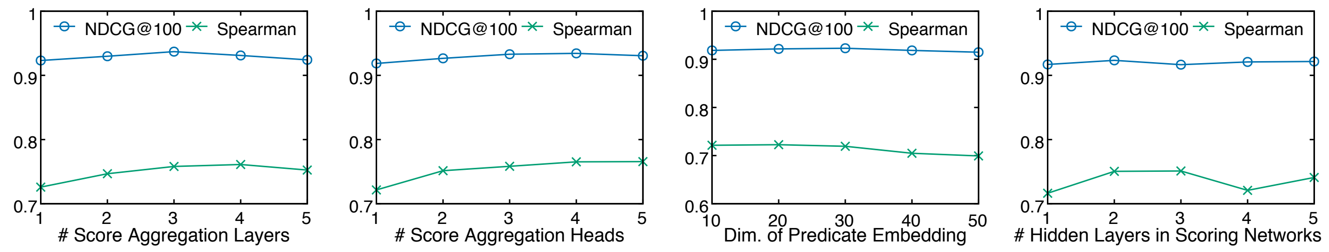


Figure 4: Parameter sensitivity of GENI on FB15K. We report results varying one parameter on x-axis, while fixing all others.

are not well suited for KGs since they do not consider edge types. To make a better use of rich information in KGs, HAR [17] extends the idea of random walk used by PR and PPR to solve limiting probabilities arising from multi-relational data, and distinguishes between different predicates in KGs while being aware of importance scores. Previous methods can be categorized as non-trainable approaches with a fixed model structure that do not involve model parameter optimization. In this paper, we explore supervised machine learning algorithms with a focus on graph neural networks.

Graph Neural Networks (GNNs). GNNs are a class of neural networks that learn from arbitrarily structured graph data. Many GNN formulations have been based on the notion of graph convolutions. The pioneering work of Bruna et al. [4] defined the convolution operator in the Fourier domain, which involved performing the eigendecomposition of the graph Laplacian; as a result, its filters were not spatially localized, and computationally costly. A number of works followed to address these limitations. Henaff et al. [12] introduced a localization of spectral filters via the spline parameterization. Defferrard et al. [6] designed more efficient, strictly localized convolutional filters. Kipf and Welling [14] further simplified localized spectral convolutions via a first-order approximation. To reduce the computational footprint and improve performance, recent works explored different ways of neighborhood aggregation. One direction has been to restrict neighborhoods via sampling techniques such as uniform neighbor sampling [10], vertex importance sampling [5], and random walk-based neighbor importance sampling [25]. Graph attention networks (GAT) [22], which is most closely related to our method, explores an orthogonal direction of assigning different importance to different neighbors by employing self-attention over neighbors [21]. While GAT exhibited state-of-the-art results, it was applied only to node classifications, and is unaware of predicates. Building upon recent developments in GNNs, GENI tackles the challenges for node importance estimation in KGs, which have not been addressed by existing GNNs.

6 CONCLUSION

Estimating node importance in KGs is an important problem with many applications such as item recommendation and resource allocation. In this paper, we present a method GENI that addresses this problem by utilizing rich information available in KGs in a flexible manner which is required to model complex relation between entities and their importance. Our main ideas can be summarized as score aggregation via predicate-aware attention mechanism and flexible centrality adjustment. Experimental results on predicting node importance in real-world KGs show that GENI outperforms existing approaches, achieving 5–17% higher NDCG@100 than the state of the art. For future work, we will consider multiple independent input sources for node importance.

REFERENCES

[1] Denilson Barbosa, Haixun Wang, and Cong Yu. 2013. Shallow Information Extraction for the knowledge Web. In *ICDE*. 1264–1267.
[2] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*. 1247–1250.
[3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*. 2787–2795.
[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *ICLR*.
[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.
[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*. 3837–3845.
[7] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question Answering over Freebase with Multi-Column Convolutional Neural Networks. In *ACL*. 260–269.
[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.
[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *KDD*. 855–864.
[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*. 1025–1035.
[11] Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *WWW*. 517–526.
[12] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep Convolutional Networks on Graph-Structured Data. *CoRR* abs/1506.05163 (2015).
[13] Jinhong Jung, Namyong Park, Lee Sael, and U. Kang. 2017. BePI: Fast and Memory-Efficient Method for Billion-Scale Random Walk with Restart. In *SIGMOD*.
[14] Thomas N. Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR* abs/1609.02907 (2016).
[15] Jon M. Kleinberg. 1999. Authoritative Sources in a Hyperlinked Environment. *J. ACM* 46, 5 (1999), 604–632.
[16] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.
[17] Xutao Li, Michael K. Ng, and Yunming Ye. 2012. HAR: Hub, Authority and Relevance Scores in Multi-Relational Data for Query Search. In *SDM*. 141–152.
[18] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.
[19] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *WWW*. 697–706.
[20] Hanghang Tong, Christos Faloutsos, and Jia-Yu Pan. 2008. Random walk with restart: fast solutions and applications. *Knowl. Inf. Syst.* 14, 3 (2008), 327–346.
[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 6000–6010.
[22] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[23] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *WWW*. 515–526.
[24] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*. 5449–5458.
[25] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. 974–983.
[26] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.

In the appendix, we provide details on datasets, experimental settings, and additional experimental results, such as a case study on TMDB5K and regression performance evaluation for in-domain predictions.

A DATASETS

We perform evaluation using four real-world KGs that have different characteristics. All KGs were constructed from public data sources, which we specify in the footnote. Summaries of these datasets (such as the number of nodes, edges, and predicates) are given in Table 3. Below, we provide details on the construction of each KG.

FB15K. We used a sample of Freebase[5] used by [3]. The original dataset is divided into training, validation, and test sets. We combined them into a single dataset, and later divided them randomly into three sets based on our proportion for training, validation, and test data. In order to find the number of pageviews of a Wikipedia page, which is the importance score used for FB15K, we used Freebase/Wikidata mapping[6]. Most entities in FB15K can be mapped to the corresponding Wikidata page, from which we found the link to the item's English Wikipedia page, which provides several information including the number of pageviews in the past 30 days.

MUSIC10K. We built MUSIC10K from the sample[7] of the Million Song Dataset[8]. This dataset is a collection of audio features and metadata for one million popular songs. Among others, this dataset includes information about songs such as the primary artist and the album the song belongs to. We constructed MUSIC10K by adding nodes for these three entities (i.e., songs, artists, and albums), and edges of corresponding types between them as appropriate. Note that MUSIC10K is much more fragmented than other datasets.

TMDB5K. We constructed TMDB5K from the TMDb 5000 movie dataset[9]. This dataset contains movies and relevant information such as movie genres, companies, countries, crews, and casts in a tabular form. We added nodes for each of these entities, and added edges between two related entities with appropriate types. For instance, given that "Steven Spielberg" directed "Schindler's List", we added two corresponding director and movie nodes, and added an edge of type "directed" between them.

IMDB. We created IMDB from public IMDb datasets[10]. IMDb datasets consist of several tables, which contain information such as titles, genres, directors, writers, principal casts and crews. As for TMDB5K, we added nodes for these entities, and connected them with edges of corresponding types. In creating IMDB, we focused on entities related to movies, and excluded other entities that have no relation with movies. In addition, IMDb datasets include titles each person is known for; we added edges between a person and these titles to represent this special relationship.

Scores. For FB15K, TMDB5K, IMDB, we added 1 to the importance scores as an offset, and log-transformed them as the scores were highly skewed. For MUSIC10K, two types of provided scores were all between 0 and 1, and we used them without log transformation.

[5]https://everest.hds.utc.fr/doku.php?id=en:smemlj12
[6]https://developers.google.com/freebase/
[7]https://think.cs.vt.edu/corgis/csv/music/music.html
[8]https://labrosa.ee.columbia.edu/millionsong/
[9]https://www.kaggle.com/tmdb/tmdb-movie-metadata
[10]https://www.imdb.com/interfaces/

B EXPERIMENTAL SETTINGS

B.1 Cross Validation and Early Stopping

We performed 5-fold cross validation; i.e., for each fold, 80% of the ground truth scores were used for training, and the other 20% were used for testing. For methods based on neural networks, we applied early stopping by using 15% of the original training data for validation and the remaining 85% for training, with a patience of 50. That is, the training was stopped if the validation loss did not decrease for 50 consecutive epochs, and the model with the best validation performance was used for testing.

B.2 Software

We used several open source libraries, and used Python 3.6 for our implementation.

Graph Library. We used NetworkX 2.1 for graphs and graph algorithms: *MultiDiGraph* class was used for all KGs as there can be multiple edges of different types between two entities; NetworkX's *pagerank_scipy* function was used for PR and PPR.

Machine Learning Library. We chose TensorFlow 1.12 as our deep learning framework. We used scikit-learn 0.20.0 for other machine learning algorithms such as random forest and linear regression.

Other Libraries and Algorithms. For GAT, we used the reference TensorFlow implementation provided by the authors[11]. We implemented HAR in Python 3.6 based on the algorithm description presented in [17]. For node2vec, we used the implementation available from the project page[12]. NumPy 1.15 and SciPy 1.1.0 were used for data manipulation.

B.3 Hyperparameters and Configurations

PageRank (PR) and Personalized PageRank (PPR). We used the default values for NetworkX's *pagerank_scipy* function with 0.85 as a damping factor.

HAR [17]. As in PPR, normalized input scores were used as probabilities for entities; equal probability was assigned to all relations. We set $\alpha = 0.15$, $\beta = 0.15$, $\gamma = 0$. The maximum number of iterations was set to 30. Note that HAR is designed to compute two types of importance scores, hub and authority. For MUSIC10K, TMDB5K, and IMDB KGs, these scores are identical since each edge in these graphs has a matching edge with an inverse predicate going in the opposite direction. Thus for these KGs, we only report authority scores. For FB15K, we compute both types of scores, and report authority scores as hub scores are slightly worse overall.

Linear Regression (LR) and Random Forests (RF). For both methods, we used default parameter values defined by scikit-learn.

Neural Networks (NN). Let $[n_1, n_2, n_3, n_4]$ denote a 3-layer neural network where n_1, n_2, n_3 and n_4 are the number of neurons in the input, first hidden, second hidden, and output layers, respectively. For NN, we used an architecture of $[N_F, 0.5{\times}N_F, 0.25{\times}N_F, 1]$ where N_F is the dimension of node features. We applied a rectified linear unit (ReLU) non-linearity at each layer, and used Adam optimizer with a learning rate $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.0005.

Graph Attention Networks (GAT) [22]. We used a GAT model with two attentional layers, each of which consists of four attention

[11]https://github.com/PetarV-/GAT
[12]https://snap.stanford.edu/node2vec/

Table 8: Top-10 movies and directors with highest predicted importance scores by GENI, HAR, and GAT on TMDB5K. "*ground truth rank*"−"*estimated rank*" is shown for each prediction.

(a) Top-10 movies (in-domain estimation). A *ground truth rank* is computed from known importance scores of movies used for testing.

	GENI		HAR		GAT	
1	The Dark Knight Rises	11	Jason Bourne	63	The Dark Knight Rises	11
2	The Lego Movie	70	The Wolf of Wall Street	21	Clash of the Titans	103
3	Spectre	10	Rock of Ages	278	Ant-Man	4
4	Les Misérables	94	Les Misérables	94	The Lego Movie	68
5	The Amazing Spider-Man	22	The Dark Knight Rises	7	Jack the Giant Slayer	126
6	Toy Story 2	39	V for Vendetta	27	Spectre	7
7	V for Vendetta	26	Now You See Me 2	81	The Wolf of Wall Street	16
8	Clash of the Titans	97	Spectre	5	The 5th Wave	67
9	Ant-Man	-2	Austin Powers in Goldmember	140	The Hunger Games: Mockingjay - Part 2	-4
10	Iron Man 2	29	Alexander	141	X-Men: First Class	767

(b) Top-10 directors (out-of-domain estimation). A *ground truth rank* corresponds to the rank in a director ranking (N/A indicates that the director is not in the director ranking).

	GENI		HAR		GAT	
1	Steven Spielberg	0	Steven Spielberg	0	Noam Murro	N/A
2	Tim Burton	9	Martin Scorsese	44	J Blakeson	N/A
3	Ridley Scott	6	Ridley Scott	6	Pitof	N/A
4	Martin Scorsese	42	Clint Eastwood	19	Paul Tibbitt	N/A
5	Francis Ford Coppola	158	Woody Allen	112	Rupert Sanders	N/A
6	Peter Jackson	-4	Robert Zemeckis	1	Alan Taylor	145
7	Robert Rodriguez	127	Tim Burton	4	Peter Landesman	N/A
8	Gore Verbinski	8	David Fincher	40	Hideo Nakata	N/A
9	Joel Schumacher	63	Oliver Stone	105	Drew Goddard	N/A
10	Robert Zemeckis	-3	Ron Howard	-2	Tim Miller	N/A

heads, which is followed by a fully connected NN (FCNN). Following the settings in [22], we used a Leaky ReLU with a negative slope of 0.2 for attention coefficient computation, and applied an exponential linear unit (ELU) non-linearity to the output of each attention head. The output dimension of an attention head in all layers except the last was set to $\max(0.25 \times N_F, 20)$. For FCNN after the attentional layers, we used an architecture of $[0.75 \times N_F,\ 1]$ with ReLU as non-linearity. Adam optimizer was applied with a learning rate $\alpha = 0.005$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.0005.

GENI. We used an architecture where each score aggregation (SA) layer contains four SA heads. For FB15K, we used a model with three SA layers, and for other KGs, we used a model with one SA layer. For SCORINGNETWORK, a two-layer FCNN with an architecture of $[N_F, 0.75 \times N_F, 1]$ was used. GENI was trained with Adam optimizer using a learning rate $\alpha = 0.005$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.0005. The dimension of predicate embedding was set to 10 for all KGs. We used a Leaky ReLU with a negative slope of 0.2 for attention coefficient computation (σ_a), and a RELU for the final score estimation (σ_s). We defined $\mathcal{N}(i)$ as outgoing neighbors of node i. Similar results were observed when we defined $\mathcal{N}(i)$ to include both outgoing and incoming neighbors of node i. Since the initial values for γ and β (parameters for centrality adjustment) affect model performance, we determined these initial values for each dataset based on the validation performance.

node2vec [9]. We set the number of output dimensions to 64 for FB15K, MUSIC10K, and TMDB5K, and 128 for IMDB. Other parameters were left to their default values. Note that node2vec was used in our experiments to generate node features for supervised methods.

C ADDITIONAL EVALUATION

C.1 Case Study

We take a look at the predictions made by GENI, HAR, and GAT on TMDB5K. Given popularity scores for some movies, methods estimate the importance score of all other entities in TMDB5K. Among them, Table 8 reports the top-10 movies and directors that are estimated to have the highest importance scores by each method with "ground truth rank"−"estimated rank" shown for each entity.

In-domain estimation is presented in Table 8(a). A ground truth rank is computed from the known importance scores of movies reserved for testing. The top-10 movies predicted by GENI is qualitatively better than the two others. For example, among the ten predictions of GAT and HAR, the difference between ground truth rank and predicted rank is greater than 100 for three movies. On the other hand, the rank difference for GENI is less than 100 for all predictions.

Out-of-domain estimation is presented in Table 8(b). As importance scores for directors are unknown, we use the director ranking introduced in Section 4.1. A ground truth rank denotes the rank in the director ranking, and "N/A" indicates that the director is not included in the director ranking. The quality of the top-10 directors estimated by GENI and HAR is similar to each other with five directors appearing in both rankings (e.g., Steven Spielberg). Although GAT is not considerably worse than GENI for in-domain estimation, its out-of-domain estimation is significantly worse than others: nine out of ten predictions are not even included in the list of top-200 highest earning directors. By respecting node centrality, GENI yields a much better ranking consistent with ground truth.

C.2 Regression Performance Evaluation for In-Domain Predictions

In order to see how accurately supervised approaches recover the importance of nodes, we measure the regression performance of their in-domain predictions. In particular, we report RMSE (root-mean-squared error) of supervised methods in Table 9. Non-trainable methods are excluded since their output is not in the same scale as the input scores. GENI performs better than other supervised methods on all four real-world datasets. Overall, the regression performance of supervised approaches follows a similar trend to their performance in terms of ranking measures reported in Table 4.

Table 9: RMSE (root-mean-squared error) of in-domain prediction for supervised methods. Lower RMSE is better. GENI consistently outperforms all baselines. Numbers after ± symbol are standard deviation from 5-fold cross validation. Best results are in bold, and second best results are underlined.

Method	FB15K	MUSIC10K	TMDB5K	IMDB
LR	1.3536 ± 0.017	0.1599 ± 0.002	0.8431 ± 0.028	1.7534 ± 0.005
RF	1.2999 ± 0.024	<u>0.1494 ± 0.002</u>	0.9223 ± 0.015	1.8181 ± 0.011
NN	1.2463 ± 0.015	0.1622 ± 0.009	0.8496 ± 0.012	2.0279 ± 0.033
GAT	<u>1.0798 ± 0.031</u>	0.1635 ± 0.007	<u>0.8020 ± 0.010</u>	<u>1.2972 ± 0.018</u>
GENI	**0.9471 ± 0.017**	**0.1491 ± 0.002**	**0.7150 ± 0.003**	**1.2079 ± 0.011**