Collective Multi-type Entity Alignment Between Knowledge Graphs

Qi Zhu[1*], Hao Wei[2], Bunyamin Sisman[2], Da Zheng[2], Christos Faloutsos[3], Xin Luna Dong[2], Jiawei Han[1]
[1]University of Illinois at Urbana-Champaign [2]Amazon.com, Inc. [3]Carnegie Mellon University
[1]{qiz3, hanj}@illinois.edu [2]{wehao,bunyamis,dzzhen,lunadong}@amazon.com [3]christos@cs.cmu.edu

ABSTRACT

Knowledge graph (*e.g.* Freebase, YAGO) is a multi-relational graph representing rich factual information among entities of various types. Entity alignment is the key step towards knowledge graph integration from multiple sources. It aims to identify entities across different knowledge graphs that refer to the same real world entity. However, current entity alignment systems overlook the sparsity of different knowledge graphs and can not align multi-type entities by one single model. In this paper, we present a **C**ollective **G**raph neural network for **Mu**lti-type entity **Align**ment, called CG-MuAlign. Different from previous work, CG-MuAlign jointly aligns multiple types of entities, collectively leverages the neighborhood information and generalizes to unlabeled entity types. Specifically, we propose novel collective aggregation function tailored for this task, that (1) relieves the incompleteness of knowledge graphs via both cross-graph and self attentions, (2) scales up efficiently with mini-batch training paradigm and effective neighborhood sampling strategy. We conduct experiments on real world knowledge graphs with millions of entities and observe the superior performance beyond existing methods. In addition, the running time of our approach is much less than the current state-of-the-art deep learning methods.

ACM Reference Format:
Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, Jiawei Han. 2020. Collective Multi-type Entity Alignment Between Knowledge Graphs. *In Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380289

1 INTRODUCTION

Knowledge Graphs (KGs) contain large volumn of relation tuples in the form of ⟨subject, *relation*, object⟩, such as ⟨Aditya Raj, *write*, Don't stop Dreaming⟩ in Figure 1. These relation tuples have a variety of downstream applications including Question Answering [19], Search, and Recommendation [42]. With the booming of structured and semi-structured online data, numerous knowledge graphs are extracted on the same domain [25]. Different KGs, though subject to the incompleteness in varying degrees, usually contain complementary information. Entity alignment (EA) aims to identify entities across different knowledge graphs that refer to the same real world entity. This problem also known as *entity matching/resolution* [12, 14, 16, 27] that matches records in the multi-relational databases.

In a knowledge graph, there are different entity types (*e.g.*, movie, actor, characters) and relation types (*e.g.*, *direct by*, *act by*, *release date*, *etc.*). Given the nature of entity types, the alignment strategy for different entity types could be different. For example, we observe much more characters than films, that share the same name in the IMDB-Freebase dataset. One obvious solution is to develop different models for different entity types; however, the solution falls short for two reasons. First, collecting annotations and training hundreds or even more models for different entity types can be very complex and expensive. Second, an entity may belong to multiple overlapping types (*e.g.* a person can be both a movie director and a novel writer), making it hard to decide which model to apply for each entity. Thus, a multi-type entity alignment algorithm becomes critical for effective knowledge integration [11].

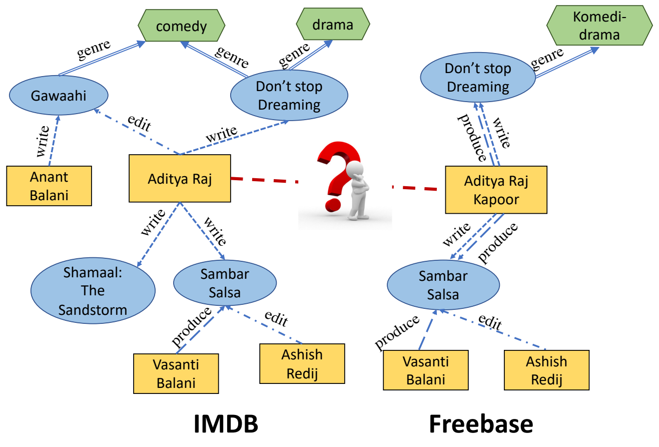


Figure 1: An example of Entity Alignment on person called "Aditya Raj" across IMDB and Freebase. Different edge types indicates different relations(*e.g.* "direct" and "write"). We use different color and shape indicates node types and different arrow types indicates different relations.

* Work performed while at Amazon.

WWW '20, April 20–24, 2020, Taipei, Taiwan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380289

However, previous entity alignment methods [4, 7, 8, 37, 45, 46, 51] suffer from the following challenges presented in the multi-type entity alignment problem.

Transductive → Inductive. Previous methods [7, 8, 37, 51] adopt knowledge graph embeddings to jointly perform the KG completion and entity alignment tasks, thus may not be tuned perfectly for alignment purpose. In particular, they focus only on related entities, *i.e.* transductive setting, ignoring the potentially rich attribute information such as the name and the released date. In addition, when *new* entities are added into the graphs, these methods require complete retraining to predict alignment for new entities.

Labeled Type → Unlabeled Type. Traditional methods[27, 39] can often perform well for entity types with rich training data, but often fail for the types where training data are sparse or even lacking. Intuitively, the rich connections between different types of entities shall help boost performance for the types with small training data, but the connections are not yet effectively leveraged on a large scale.

Inspired by the recent success of Graph Neural Networks (GNN) on various tasks such as node classification [21], link prediction [5, 48] and graph classification [23], we propose to apply GNN to generate structure-aware representations for each entity, and align entities by comparing their representations. The GNN mechanism allows us to incorporate neighborhood information recursively and make inductive predictions on *unseen* entities, thus addressing both of the afore-mentioned challenges. Unfortunately, as we show in our experiments (Section. 4.4), a vanilla application of GNN failed terribly, obtaining only 0.33 F1 score (27% precision and 43% recall) for alignment. The key reason is that the GNN models will generate similar embeddings for the same entity from two different KGs only if both KGs contain fairly complete information about the entity. In reality, most KGs are sparse in different ways, making the embeddings often very different. For example, for the same movie, IMDB may contain editor, director and actor information, while Freebase contains only director and producer information.

This paper presents a novel GNN model that makes collective decisions [2, 36] (*i.e.* related entities alignment are determined jointly) on entity alignment for multple different types. The key of our solution is a carefully designed attention mechanism that effectively leverages shared neighborhoods as positive evidence without ignoring strong negative evidence. First, to be robust on incomplete knowledge graphs, we design the cross-graph attention that allows focusing more on the similar neighborhoods across two graphs. To illustrate the intuition, consider our motivating example in Figure 1. "Aditya Raj" participates in four movies in IMDB, whereas "Aditya Raj Kapoor" writes/produces two movies in Freebase; a vanilla version of GNN will generate different representations for them. Our cross-graph attention gives higher weight to shared neighbors such as "Sambar Salsa", and thus generate similar representations for the two nodes. Second, to be sensitive towards strong negative evidence, we employ relation-aware self-attention on edges that prevents blindly aligning nodes with similar neighborhoods. For example, two movies in the same series are likely to share directors, writers, and some actors; our edge-level attention allows us to pick up key differences in release year and length to distinguish them. Indeed, our experiments show that the two attention mechanisms collectively improve linkage quality by 10% F1 score in average.

Table 1: Comparison of methods for entity alignment. *Inductive*: Making use of node features and generalize to new nodes. *Predicate*: Modeling semantics of different relations. *Collective*: Collecting evidence from neighborhood. *Multi-type*: Handling multiple entity types in one model. *Scalable*: Scaling up to millions of nodes.

	CG-MuAlign	**MuGNN** [4]	**GCN-Align** [45]	**DeepMatcher** [27]
Inductive	✔		✔	✔
Predicate	✔	✔		✔
Collective	✔			
Multi-type	✔	✔	✔	
Scalable	✔			✔

We note that although collectively linking entities is not a new idea [2, 12, 31, 34], our method is the first scalable solution that does not require any manually defined rules (like [31]) or logic (like [34]) for evidence propagation. Similarly, although GNN has been widely adopted for iteratively capturing the neighborhood information, our model, to the best of our knowledge, is the first that allows collective decisions in a GNN. Besides, we develop a scalable GNN framework to support large-scale entity alignment in the experiments. In Table. 1, we compare our method with most recent entity alignment algorithm from five different perspectives. In particular, we made the following contributions.

- We propose a GNN-based knowledge graph entity alignment framework called CG-MuAlign, that collectively align entities of different types. We carefully design the attention mechanisms that can both effectively accumulate positive evidence from the neighborhood, and remain sensitive to strong negative evidence to distinguish similar but different entities.
- We scale up our model to large-scale knowledge graphs by avoiding expensive computation in each layer of the deep neural network and by relation-aware neighborhood sampling.
- Through extensive experiments on two different datasets, we show that our methods obtain high quality linkage (80.5% F1 and 60% recall when precision is 95%) on knowledge graphs with size of two and half millions of nodes. In particular, with the help of labeled film data, we show that CG-MuAlign trained on 2,000 person pairs can reach comparable performance with model trained on ~24,000 person pairs.

The rest of the paper is organized as follows. We first provide the preliminary knowledge and problem definition in Section 2. Our method is presented in Section 3 and we demonstrate the experimental results as well as analysis in Section 4. We review the literature and summarize the differences of our methods in Section 5. At last, we conclude the whole paper in Section 6.

2 PROBLEM DEFINITION

A knowledge graph G is defined as a graph with multi-typed nodes and edges. We denote nodes $\mathcal{V}$ as entities and edges $\mathcal{E}$ as relations. Formally we have $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ with a node type mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}$ and edge type mapping $\psi : \mathcal{E} \rightarrow \mathcal{R}$.

Given two different knowledge graphs G and G′ on same domain, the node type and edge type are $\{\mathcal{T}, \mathcal{T}'\}$ and $\{\mathcal{R}, \mathcal{R}'\}$, respectively. Assuming node and edge types are aligned in advance: $\mathcal{T}^*\{(t, t') \in \mathcal{T} \times \mathcal{T}' | t \Leftrightarrow t'\}$, $\mathcal{R}^*\{(r, r') \in \mathcal{R} \times \mathcal{R}' | r \Leftrightarrow r'\}$, certain amount of ground truth node pairs $\mathcal{S}\{(v_i^{t^*}, v_{i'}^{t^*}) | t^* \in \mathcal{T}^*\}$ are available. Normally, there are only a few aligned seed pairs for some of the aligned node type $\mathcal{T}^*$, *i.e.* $|\mathcal{S}| \ll |\mathcal{V}|$.

Formally, we define the problem of entity alignment as follows.

Definition 2.1 (KG Entity Alignment). Given two knowledge graphs $G = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$ and $G' = (\mathcal{V}', \mathcal{E}', \mathcal{T}, \mathcal{R})$, *entity alignment* aims to find a set of entity pairs $\{(v_i, v_{i'}) \in \mathcal{V} \times \mathcal{V}'\}$ with high precision and recall, such that each pair refers to the same real world entity.

3 METHOD

CG-MuAlign features a collective GNN framework to address the KG Entity Alignment problem. Our model not only bridges the gap between single-type and multi-type alignment model, but also generalize to unlabeled types. In Section 3.1, we describe the overall picture of our alignment model. Then we discuss two proposed attention mechanisms and explain how they contribute to the *collective* setting in Sections 3.3 and 3.4, respectively. At last, we present our model specifications and reason about scalability concerns.

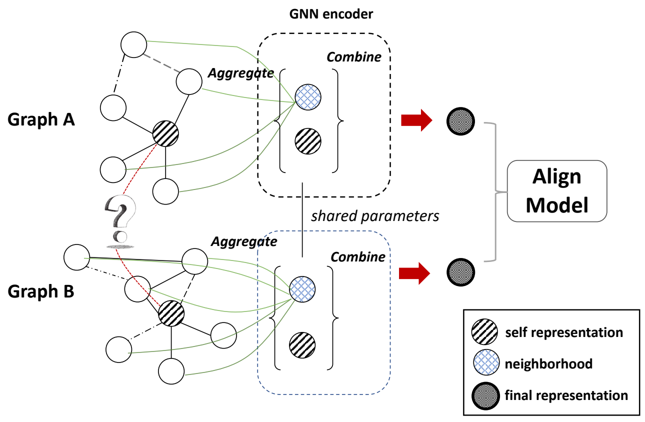


Figure 3: CG-MuAlign **architecture**

3.1 Solution Overview

We model the entity alignment problem as a classification problem, where we predict whether two nodes $v \in V$ and $v' \in V'$ represent the same real-world entity.

The model includes two GNN encoders and an entity alignment loss layer. The GNN encoder takes an K-hop sub-graph derived from target node v, aggregates the neighborhood information and outputs representation h_v^k for node v. In its k-th layer, for node i, the GNN encoder aggregates neighbor information from k-1 layer,

$$z_i^k = \textsc{Aggregate} \circ \textsc{Transform}^{(k)} \left(\{h_j^{k-1}, j \in \mathcal{N}_i\} \right) \tag{1}$$

where h^{k-1} is the hidden representation of the previous layers and $\mathcal{N}_i$ is the neighborhood of node i in the knowledge graph. The output representation h_i^k is the combination of h_i^{k-1} and z_i^k,

$$h_i^k = \textsc{Combine}^{(k)} \left(\{h_i^{k-1}, z_i^k\} \right) \tag{2}$$

For two KGs, we have two K-layer models GNN_1 and GNN_2 with identical structure and shared parameters. For each pair of entities (i, i') in the training data, we sample N negative entities from KG_1 and KG_2. Then we obtain the final representations from two GNN encoders as $(h_i^K, h_{i'}^K)$ and apply a marginal hinge loss on distance between output vector of two nodes,

$$\mathcal{L} = \sum_{(i,i')} \sum_{(i-,i'-)} \max\left(0, d(h_i^K, h_{i'}^K) - d(h_{i-}^K, h_{i-'}^K) + \gamma\right)$$

In the experiments, we use $d(x, y) = ||x - y||_2$ as the distance function. The overall architecture of our solution is shown in Figure 3.

3.2 Collective Graph Neural Networks

In CG-MuAlign, we first group the neighbor nodes by edge type r as $\mathcal{N}_{i,r}$ and apply different $\textsc{Transform}$, *i.e.* W_r. In Figure 1, for example, the target node "Aditya Raj" in the left IMDB sub-graph have $N_{i,\text{write}}$ = {Don't stop Dreaming, Shamaal: The Sandstorm, Sambar Salsa} and $N_{i,\text{edit}}$ = {Gawaahi}. At each layer, we transform the neighborhood $(j \in \mathcal{N}_{i,r})$ information regarding the relation between node i and j as follows,

$$z_{i,j}^k = W_r^k h_j^{k-1}, j \in \mathcal{N}_{i,r} \tag{3}$$

As one entity can belong to multiple overlapping types, the above transformation explicitly differentiate the same person's representations as editor and writer in the aggregation.

We calculate node-level attention α (details in Section 3.3), edge-level attention β (details in Section 3.4) and $\textsc{Aggregate}$ neighborhood as,

$$z_i^k = \sum_{\cup \mathcal{N}_{i,r}} \alpha_{ij} \beta_{ij} z_{i,j}^k, \Sigma_j \alpha_{ij} \beta_{ij} = 1 \tag{4}$$

Then we proposes the following $\textsc{Combine}$ function:

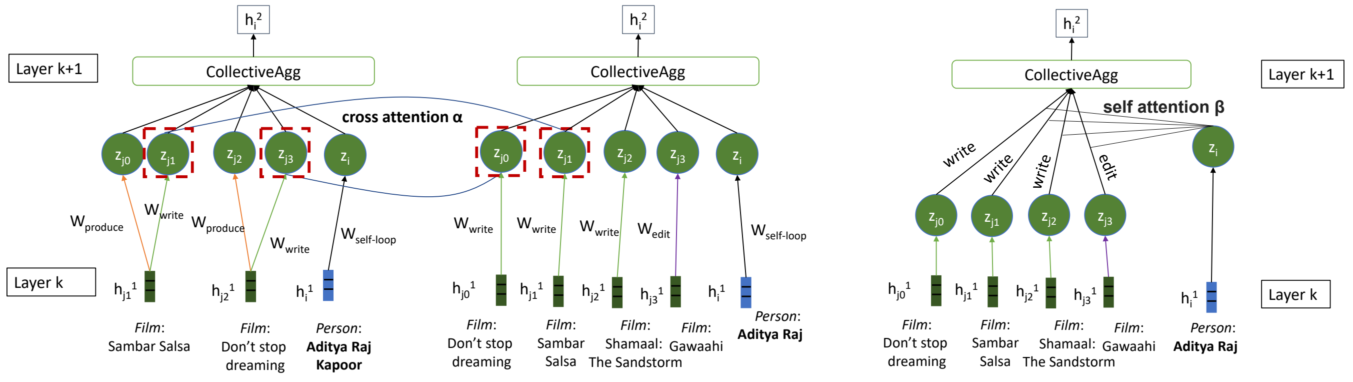
$$h_i^k = \sigma\left([W_{self}^k h_i^{k-1} || z_i^k] \right) \tag{5}$$

Intuitively, we concatenate the self information and neighborhood information to make the alignment decision on self information and neighborhood information independently. And we name this layer as $\textsc{CollectiveAgg}$.

In CG-MuAlign, we stack multiple layers in each GNN encoder, where the inputs at layer k is the output representation of layer k-1. The layer-0 representation is the input node features and we allow entities of different types to have different length of features. Let the hidden dimension of the model be m, we have the first layer of relation matrices $W_r^1 \in \mathbb{R}^{d_r \times \frac{m}{2}}$, where d_r is the feature length of entity in neighbor group $\mathcal{N}_r$. After concatenation as depicted in Equation 5, the hidden representation is then $\frac{m}{2} + \frac{m}{2} = m$. For the layer $k = 2, 3, ..., K$, we have $W_r^k \in \mathbb{R}^{m \times \frac{m}{2}}$ Then we describe how we compute the two attentions α and β.

3.3 Node-level Cross-graph Attention

Existing GNN-based entity alignment methods reconcile structural difference across two knowledge graphs by implicit means, such as graph matching objective [46] and rule grounding [4]. As we discussed in the introduction, the structural differences are mainly raised by the nature of incompleteness in a knowledge graph. In CG-MuAlign, we address this problem by collective aggregation of *confident* neighborhood information. Namely, we explicitly assign higher weights for those neighbors that are likely to have the corresponding ones in the other graph. We achieve this by employing a cross-graph attention mechanism that attends over the neighbor's feature vectors.


Layer k+1
Layer k
CollectiveAgg
cross attention α
self attention β
h_i^2
z_{j0} z_{j1} z_{j2} z_{j3} z_i
W_produce
W_write
W_self-loop
W_edit
write
edit
h_{j0}^1 h_{j1}^1 h_{j2}^1 h_{j3}^1 h_i^1
Film: Sambar Salsa
Film: Don't stop dreaming
Person: Aditya Raj Kapoor
Film: Shamaal: The Sandstorm
Film: Gawaahi
Person: Aditya Raj

(a) Cross-graph Attention (b) Relation-aware Self-Attention

Figure 2: Illustration of node-level and edge-level attention in CG-MuAlign

Given the candidate node pair (i, i'), we have $\mathcal{N}_i$ and $\mathcal{N}_{i'}$ as neighborhood of node i and node i', respectively. We make *soft* decisions by calculating similarity of pairs $(p, q) \in \mathcal{N}_i \times \mathcal{N}_{i'}$,

$$\alpha_p = \frac{\sum\limits_{q \in \mathcal{N}_{i'}} \exp^{-||z_p - z_q||_2}}{\sum\limits_{p \in \mathcal{N}_i} \sum\limits_{q \in \mathcal{N}_{i'}} \exp^{-||z_p - z_q||_2}}, \alpha_q = \frac{\sum\limits_{p \in \mathcal{N}_i} \exp^{-||z_q - z_p||_2}}{\sum\limits_{q \in \mathcal{N}_{i'}} \sum\limits_{p \in \mathcal{N}_i} \exp^{-||z_q - z_p||_2}}$$

The hidden representation z_p and z_q are calculated in Equation 3. For $p_1, p_2 \in \mathcal{N}_i$, $\alpha_{p_1} > \alpha_{p_2}$ if the accumulated similarity between p_1 and neighbors $\mathcal{N}_{i'}$ in Graph G' is larger than p_2. In computation, weight α_p and α_q are the row-wise and column-wise normalized vector for the cross-graph attention matrix $A_{i,i'} \in \mathbb{R}^{|\mathcal{N}_i| \times |\mathcal{N}_{i'}|}$. In Figure 2a, we turn the 1-hop neighbor in Figure 1 into actual computation graph in our CollectiveAgg layer. The neighborhood for "Aditya Raj" two knowledge graphs are {Gawaahi:*edit*, Don's stop Dreaming:*write* , The Sandstorm:*write*, Sambar Salsa:*write* } and {Don's stop Dreaming:*write*, Don's stop Dreaming:*produce*, Sambar Salsa:*write*, Sambar Salsa:*produce* }. The cross-graph attention will give high weights to neighbor nodes {Sambar Salsa:*write*, Don's stop Dreaming Salsa:*write*} as their hidden representation is similar. Thus, the proposed cross-graph attention leverages the *positive* evidence to the collective decisions.

3.4 Edge-level Relation-aware Self-attention

Yet, cross-graph attention neglects the *negative* evidence across the graphs. If the neighborhood aggregation only relies on the cross-attention, it fails to predict "negative" when only unimportant nodes are softly aligned. In our music data set at Figure 2, when aligning song "Radioactive" by American rock band Imagine Dragons between Amazon Music and Wikipedia, cross-graph attention produce *positive* evidence on most of the neighbors such as song writer, producer and one performer. However, it is an unmatched pair since the one in Amazon is a deluxe version collaborated with "Kendrick Lamar". In other words, different relations shall play different roles in alignment prediction. For example, *performed by* is more informative than *written by*.

In fact, the computation of cross-graph attention focuses on the neighbor nodes similarity and considers each relation equally important. In light of this issue, similar with Graph Attention

Table 2: Alignment Example for song Radioactive. Neighbor nodes are grouped by relations as described in Section 3.2. Bold font indicates the neighbor node with large cross-attention weights.

	Amazon Music	Wikipedia
Attributes		
Title Duration	Radioactive 2M19S	Radioactive 2M19S
Neighbors		
Song writer	**Wayne Sermon** **A. Grant** **Dan Reynolds** **Josh Mosser**	**Wayne Sermon** **Alexander Grant** **Dan Reynolds** **Josh Mosser**
Song producer	Alex Da Kid	
Album	Night Visions (Deluxe)	Night Visions
Main performer	**Imagine Dragons** Kendrick Lamar	**Imagine Dragons**

Networks [41], we adjust the cross-graph attention with an edge-level self-attention that considers the edge(tuple) information, *i.e.* ⟨Radioactive, *perform by*, Kendrick Lamar⟩ we calculate an edge-level self-attention by a weight vector $\vec{a}_r$ to estimate the importance of an edge composed of subject, object and relation.

$$\beta_{ij} = \frac{\exp(\sigma(\vec{a}_r^T [z_i || z_j]))}{\sum\limits_{k \in \mathcal{N}_i} \exp(\sigma(\vec{a}_r^T [z_i || z_k]))}$$

We use $\sigma(\cdot)$ as LeakyReLU suggested in [41]. As depicted in Figure 2b, self-attention measures the importance of a relation tuple with the relation aware linear layer a_r. In the previous example, the attention score of ⟨Radioactive, *perform by*, Kendrick Lamar⟩ is similar with ⟨Radioactive, *perform by*, Imagine Dragons⟩ and much larger than grouped neighbors such as writer and producer.

3.5 Scaling up

Despite the effectiveness of the proposed GNN model, training and applying it is very expensive. We scale it up in three ways: by carefully removing unnecessary computation, by strategically sampling the neighborhood, and by selectively considering the matching candidates.

Simplifying Computation: We now analyze the effectiveness of CollectiveAgg under the Open World Assumption[1], that is, no knowledge graph has complete knowledge. We assume graph G and G' observes p portion and q portion from the underlying complete knowledge G_u. In our example in Figure 1, both IMDB and Freebase contains only partial information of "Aditya". Given a ground truth pair (i, i'), that both refers to the real world entity e, the number of neighborhood of e in the real world is $\mathcal{N}_e$. We now quantify the *Collective Power* by counting numer of shared (same) nodes regarding order of the neighbors.

Theorem 3.1. *If G and G' have the same number of nodes,* i.e. $|\mathcal{V}_1| = |\mathcal{V}_2|$ *and there exists a injective function $\mathcal{F} : \mathcal{V}_1 \rightarrow \mathcal{V}_2$. Let K denote the order of the neighborhood, $|\mathcal{E}|$ is the total number of edges in the underlying graph G_u, the expected Collective Power decays geometrically as K increases.*

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1} CP(K) \le |\mathcal{E}| \cdot p^{\frac{K}{2}} q^{\frac{K}{2}}$$

Proof. According to the definition of p and q. Let p_i and q_i be the actual observed ratio for node v_i and $\mathcal{F}(v_i)$ in graph G and G', we have,

$$p = \frac{\sum_{i=1}^{|\mathcal{V}_1|} |\mathcal{N}_i| \cdot p_i}{|\mathcal{E}|}, q = \frac{\sum_{i=1}^{|\mathcal{V}_2|} |\mathcal{N}_i| \cdot q_i}{|\mathcal{E}|}$$

For a specific node i, the expected number of same neighborhood from a uniform distribution in two graphs is $|\mathcal{N}_e| p_i q_i$. Thus, when $K = 1$,

$$\mathbb{E}_{(v,\mathcal{F}(v))\sim G_1} CP(1) = \sum_i |\mathcal{N}_e| p_i q_i \tag{6}$$

$$\le \sqrt{\sum_i (\sqrt{\mathcal{N}_e} p_i)^2 \sum_i (\sqrt{\mathcal{N}_e} q_i)^2} \tag{7}$$

$$\le \sqrt{\sum_i \mathcal{N}_e p_i \sum_i \mathcal{N}_e q_i} = |\mathcal{E}| \cdot \sqrt{pq} \tag{8}$$

Recursively, we repeat the same calculation on shared neighbor nodes in previous step, that is, $\mathbb{E}[CP(K+1)] = \mathbb{E}[CP(K)] \cdot \sqrt{pq}$ □

The above theorem can be explained as jaccard similarity of neighborhood follows a long-tail distribution as K grows, because only same first-order neighbor nodes may contain the same second-order neighbor nodes in principle. According to this, we employ the CollectiveAgg as the Aggregate only at the last layer to reduce the computation cost as the collective power decrease. That is,

$$h_i^k = \begin{cases} \text{CollectiveAgg}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right), & k = K-1 \\ \text{AverageAgg}\left(\{h_j^{k-1}, j \in \mathcal{N}_i \cup \{i\}\}\right) & k < K-1 \end{cases} \tag{9}$$

where the AverageAgg replaces the $\alpha_{ij}\beta_{ij}$ in Equation 4 as $\frac{1}{|\mathcal{N}_i|}$.

Mini-batch Training and Neighborhood Sampling. Traditional graph neural nets are trained globally, which is infeasible when the graph is large. Instead, we sample a batch of positive pairs from training data and construct a K-hop sub-graph from G and G'. To further speed up the training, we adopt neighborhood sampling to control the size of the computation graph.

Lemma 3.2. *Let the maximum neighborhood size as N and batch size as B, the space complexity of* CG-MuAlign *is $O(BN^K)$. Without batch training or sampling, the space complexity is $O(|\mathcal{V}| \cdot K)$. For training data of size S, the expected running time is $O(S \cdot N^K)$.*

Additionally, we adopt a relation-aware neighborhood sampling to leverage the maximal collective power, which samples those "one-to-one" relation first. The probability of sampling possibly matched neighbor node is greater than those "one-to-many" relations. For example, one movie usually has only one director but many actors, knowing whether the director is same is more informative than knowing one actor is same. For each type of entity v^t, we calculate the average number $avg\_N_r^t$ of neighbors connected by relation r. During the neighborhood sampling process for node i of type t, we sample from the neighborhood group $\mathcal{N}_{i,r}$ with probability

$$\Pr(n) \propto \left(\frac{avg\_N_r^t}{\sum_r avg\_N_r^t}\right)^{-1}$$

Therefore, director neighbors are more likely to be sampled compared with characters and actors due to their large population. It helps make the collective decisions when we sample a small number of neighborhoods.

Candidate Generation. Though the training cost is controlled by number of GNN layers and number of sampled neighbors, the inference cost remains as a problem. Naive one-versus-all comparison leads to time complexity up to $O(|\mathcal{V}|!)$. To scale up to millions of entities, we employ candidate generation during the inference stage, also known as blocking. For each test node, we use several strong keys(*e.g.* name and date of birth for person) to collect possible match entities and use CG-MuAlign to predict alignment score within candidate pairs.

3.6 Relations with other GNN variants

Now we summarize the key differences of proposed CollectiveAgg with previous popular GNN framework.

Similar with RGCN [32], we adopt multi-relational matrices to model the semantics of different relations when aggregating the neighbors. Our self-attention modules shares similar motivation with GATGAT [41]. Both GraphSage and CollectiveAgg characterize with concatenating self representation and neighborhood representations. The GraphSage GNN layer includes concatenation and aggregate function, like average

$$h_i^k = \sigma\left(W_1\left[h_i^{k-1} || \sigma\left(W_2 \cdot \text{MEAN}\{h_j^{k-1}, j \in \mathcal{N}_{i,r}\}\right)\right]\right), \tag{10}$$

There are two differences between CollectiveAgg and GraphSage. First, we have multi-relational projection matrix W_r in the hidden layer. Second, we use weighted average (attention) Λ instead of averaging or max pooling.

$$h_i^k = \sigma\left(W_1\left[h_i^{k-1} || \Lambda(W_r \cdot \{h_j^{k-1}, j \in \mathcal{N}_i\})\right]\right) \tag{11}$$

In the toy example below, all kinds of previous aggregation function, *e.g.* MEAN/MAX/SUM, fail to fit the label if node id is the only input feature. A learnable mask Λ on neighborhood, instead,

[1]the assumption that the truth value of a statement may be true irrespective of whether or not it is known to be true, from wikipedia:https://en.wikipedia.org/wiki/Open-world_assumption

Table 3: Overall Dataset Statistics

Dataset	# Nodes	# Edges	# Node Types	# Edge Types
Movie	2,684,233	6,851,166	8	8/8
Music	1,768,983	10,723,141	6	4/5

Table 4: Movie Dataset

Dataset	# Films	# People	# Characters	# Genres	# Train/Test
Freebase	273,526	314,869	859,289	599	53,405/53,405
IMDB	423,118	600,909	211,895	28	

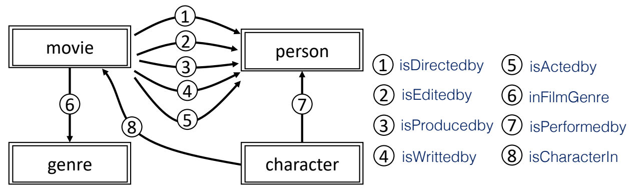


Figure 4: The schema of the Movie Graph

can fit the label by masking out node c and d. To some extent, CollectiveAgg has a greater representation power for the task of entity alignment when data is sparse.

Example 3.3. For node $a \in G$ and $a' \in G'$, we have first-order neighbors $\{b, c, d\}$ in graph G and $\{b\}$ in graph G', the training label is 1.

4 EXPERIMENTS

We compare CG-MuAlign with other knowledge graph alignment algorithms to examine our three major claims one by one in Section 4.4.

- CG-MuAlign outperforms existing methods on real-world large-scale dataset.
- Collective alignment is not sensitive to the amount of training data.
- CG-MuAlign generalizes to unlabeled type effectively with limited labels.

4.1 Datasets

In our experiments, we use two different knowledge graph alignment data sets and evaluate the performance under inductive settings. Both (*i.e.* Movie and Music domain) contain abundant node attributes and feature with millions of nodes and tens of millions edges of different types. We report basic graph statistics in Table 3 and then introduce them in more details. The number of nodes and number of edges are summed over two knowledge graphs.

Movie Dataset contains a subset of IMDB (an online database of information related to films) and Freebase (a large knowledge base on general domains). The latter originally has a large number of edge types compared with IMDB. We sample a subset of Freebase that is related to the movie domain. It has ground truth links to the IMDB ID for some of the films and people. We split the ground truth pairs into training and testing data. It has four different entity types and eight different relations, the schema can be found in Figure 4.

Table 5: Music Dataset

Dataset	# Songs	# Albums	# Artists	# Train/Test
Wikipedia	104,179	188,602	71,409	57,062/23,485
Amazon-Music	999,900	200,911	201,550	

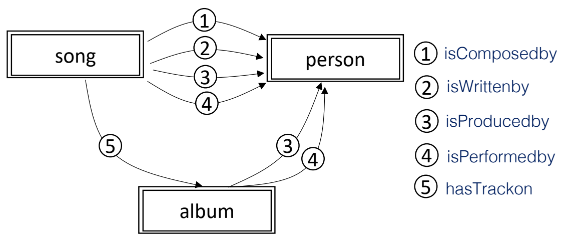


Figure 5: The schema of the Music Graph

In Table 4, we report the distribution of entity types and the size of the training/testing data.

Music Dataset contains two music knowledge graph from Amazon Music and wikipedia. There are three major types in this dataset: song, album and artist. The five relations among them can be found in Figure 5. The positive pairs on songs and albums are generated with noise and we ask annotators to label testing pairs among a candidate pool for two types. Detailed number of entities can be found in Table 5.

4.2 Baselines

We consider methods from three families: (1) link prediction (2) entity alignment between graphs (3) entity matching in multi-relational database.

Link prediction. Between two knowledge graphs G and G', we can add *equivalent* edges between ground truth node pairs $\{(v_i^t, v_j^{t}{}')\}$. We then run advanced graph embedding algorithm with node features to embed nodes from different graphs in the same unified space. Later, we train a two-layer perceptron on the labeled *equivalent* edges. Specifically, we consider the following method that consider the node attributes:

- GraphSage [18] is the first large-scale inductive representation learning algorithm.

We denote this method as **GraphSage+NN** along with another baseline named **Feature+NN** to verify the feature effectiveness and inspect how different methods gain improvement over its performance.

Knowledge Graph Alignment. Most of the previous work focus on the transductive setting. Some recent work [4, 45, 46] based on Graph Neural Networks, start to extend graph alignment problem under inductive setting. We group these methods into **transductive only**: MuGNN [4] and BootEA [37] that both models knowledge graph embedding and entity alignment simultaneously and **inductive**: MultiKE [49] and AttrE [39] further incorporate attribute information into embedding-based entity alignment. GCN-Align [45] models both structure and attribute features with same relation matrices for different relations. As we found embedding-based methods fail to scale up to graphs with millions of entities, we carefully verify the effectiveness of proposed GNN model with following recent GNN variants.

- GCN-Align [45] models both structure and attribute features with the original graph convolutional network [21].
- GraphSage [18] concatenates the self feature vector and neighborhood aggregation vector.
- GAT [41] aggregates neighborhood information with multi-head attention.
- RGCN [32] differs GCN with multi-relational linear transformation matrices.
- R-GraphSage is a variant of GraphSage with multi-relational linear transformation matrices.

To address the scalability issue, we re-implement all of them in PyTorch [28] under DGL framework [43]. CG-MuAlign and above GNN variants adopt same mini-batch paradigm training described in Section. 3.5 with the batch size of 32. We sample 10 negative entities from each graph and have total 20 negative samples for every positive pair. We use Adam [20] as our optimizer with learning rate as 0.003. We set the max neighborhood size as 10 in the neighbor sampler function. The number of layers for all GNN methods are set as two. And we set hidden dimension as 100 for link prediction and graph alignment baselines.

Entity Matching. We refer methods that finds all tuple pairs (a, b) across different multi-relational databases into this category. We explore the performance of two representative methods:

- Magellan [22] is end-to-end entity matching framework that supports different matching functions like linear regression, SVM, random forest, *etc.* We choose random forest as the matching function.
- DeepMatcher [27] is a recent deep learning entity matching algorithm, we use its “hybrid” mode in our experiments.
- PARIS [36] is an unsupervised RDF ontologies alignment model, which makes collective decisions based on iterative probability estimates.

4.3 Experimental Settings

Now we describe how we conduct the experiments and evaluate the performance.

Data Processing. For all of the GNN-based methods, we pre-compute the feature vector of different entities. There are two major types of features: string and numerical. We use fastText [26] to encode string features. For numerical features, we preserve the original value except time values. For time values, like duration, date of birth, we use periodical function $\sin(\cdot)$ to encode each periodical segment, *e.g.* seconds, minutes. Finally, we concatenate all of the features into a unified vector as the node feature.

For entity matching baselines, we convert one-hop neighbor node in the knowledge graph into the format **relation@attribute**, *e.g.* for a movie record, we have the field **isDirectedby@Name** indicating movie director’s name. Thus, we can turn the first order information in the knowledge graph into a multi-relational table in a lossless way. In Magellan [22], the features used in the random forest are automatically generated by the model. Different string similarities are computed as features, such as jaccard similarity, levenshtein edit distance between attributes of entity pairs.

Evaluation Metrics. We evaluate different methods on both labeled and unlabeled settings and report their Recall@Precision=0.95, F1, PRAUC (precision-recall area under curve) and hit@1 on three data sets. Typically, previous research mainly use hit@1 since the evaluation data set is small. It is infeasible to conduct one-versus-all comparison when there are millions of candidate nodes. Thus, we use candidate generation introduced in Section. 3.5 in the testing stage and report hit@1 based on the candidate set. We report the precision and recall curve while tuning the alignment threshold. PRAUC and best F1 provide more insights how different methods perform without knowing all positive pairs. We will later show in Table 6, methods have similar hit@1 result could produce rather different PRAUC and F1. Besides, we propose metric Recall@Precision=0.95 to evaluate model performance when high precision is required.

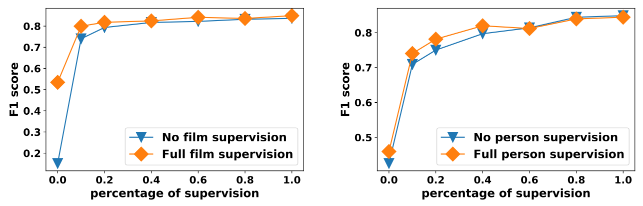
Evaluation Settings. The ground truth links between person and movie serve as positive data in training and testing. During training, we adopt the same random sampling to construct negative samples for different methods as we assume no prior knowledge of the target domain. We construct the testing data by joining output of candidate generation and the test split of ground truth links. Specifically, we use blocking function in Magellan [22] to generate candidates. For example, we use person’s name and date of birth(allow missing) as the blocking key. Note that on the music domain, the ground truth links are also noisy. We annotate a subset of the candidates, thus, hit@1 metric is not included for music data. For unlabeled type evaluation, we use the same way to generate the evaluation data.

4.4 Experiments and Performance Study

Alignment Result on labeled types. We train a unified model for multiple entity types and report all of baselines including GNN variants. From Table 6, we can conclude CG-MuAlign outperforms all other method. On the movie dataset, it yields a large margin over the second best method - DeepMatcher. It is mainly because IMDB and Freebase have rather different relation distributions and they suffer from data incompleteness differently. DeepMatcher considers the difference between attribute sets from two graphs, thus, it performs better than the remaining ones. It is quite surprising that Feature+NN outperforms most of the GNN variants, which indicates the neighborhood information affects the performance negatively in those methods. Although other GNN algorithms suffer from the sparsity of knowledge graphs while our collective aggregation layer avoid performance drop by aggregating mostly aligned neighborhood via cross-graph attention. Specifically, among three GNN variants that do not consider multi-relational structure (GCN, GrageSage, GAT) perform worse than those includes multi-relational transformation as expected. We find the concatenation mechanism first introduced in GraphSage benefit the task. The reason could be self-information is critical to the alignment task and mixing it with neighborhoods confuses the model predictions. On the music data set, CG-MuAlign gain a smaller margin over other baselines as we observe the music graphs are much denser. The performance difference is similar with the movie dataset, vanilla GNN perform badly while GraphSage and R-GraphSage obtain reasonable results compared with Feature+NN. We notice the link

Table 6: Alignment Result on labeled types for inductive setting. For simplicity, transductive only methods are not included in this table. We report the standard deviation by 5-runs of each method except DeepMatcher, which takes long time for one run.

	Method	Movie Dataset				Music Dataset		
		Rec@Prec=.95	PRAUC	F1	hit@1	Rec@Prec=.95	PRAUC	F1
	Feature+NN	0	0.3672 ± 0.053	0.6380 ± 0.000	0.7197 ± 0.001	0.0025 ± 0.002	0.7251 ± 0.027	0.6795 ± 0.009
	GraphSage+NN	0.0155 ± 0.001	0.3229 ± 0.003	0.3557 ± 0.001	0.4503 ± 0.003	0.0002 ± 0.000	0.2468 ± 0.018	0.3134 ± 0.012
	Magellan	0.4387 ± 0.000	0.7067 ± 0.000	0.6945 ± 0.000	0.7974 ± 0.000	0.1071 ± 0.000	0.7461 ± 0.000	0.6760 ± 0.000
	DeepMatcher	0	0.5829 ± 0.000	0.7549 ± 0.000	0.8468 ± 0.000	0	0.1748 ± 0.000	0.3559 ± 0.000
	PARIS	0.5840 ± 0.000	0.7759 ± 0.000	0.7661 ± 0.000	0.7725 ± 0.000	0.2333	0.4175 ± 0.000	0.4640 ± 0.000
GNN variants	GCN	0.0098 ± 0.001	0.2831 ± 0.006	0.3313 ± 0.004	0.4896 ± 0.003	0.0020 ± 0.002	0.3829 ± 0.009	0.4190 ± 0.003
	GraphSage	0.1900 ± 0.007	0.5589 ± 0.004	0.5251 ± 0.003	0.6605 ± 0.009	0.2868 ± 0.029	0.8252 ± 0.003	0.7637 ± 0.001
	GAT	0.0147 ± 0.002	0.3448 ± 0.006	0.3793 ± 0.004	0.5483 ± 0.003	0.0004 ± 0.001	0.4485 ± 0.014	0.4819 ± 0.007
	RGCN	0.0106 ± 0.002	0.4247 ± 0.003	0.4435 ± 0.001	0.5450 ± 0.002	0.0025 ± 0.004	0.4419 ± 0.024	0.4625 ± 0.020
	R-GraphSage	0.2829 ± 0.009	0.6573 ± 0.003	0.6110 ± 0.004	0.7125 ± 0.003	0.4081 ± 0.029	0.8335 ± 0.004	0.7646 ± 0.003
	CG-MuAlign	**0.6010 ± 0.004**	**0.8548 ± 0.004**	**0.8050 ± 0.006**	**0.8869 ± 0.002**	**0.4437 ± 0.023**	**0.8400 ± 0.008**	**0.7762 ± 0.004**



(a) Test on Person **(b) Test on Film**

Figure 6: Sensitivity to the amount of training data. The orange curve indicates the collective setting, *i.e.* supervision of other types are provided. The blue curve indicates the non-collective setting.

Table 7: Alignment Result on unlabeled types for few-shot setting. We mark the best and second-best result. Column person stands for unlabeled type in the evaluation.

	Method	Person		Film	
		PRAUC	F1	PRAUC	F1
	Node features	0.8285	**0.8563**	0.4231	0.4780
	PARIS	0.7303	0.7489	0.8392	0.7961
GNN variants	GCN	0.3492	0.4659	0.2589	0.3223
	GraphSage	0.5495	0.6069	0.4269	0.4158
	GAT	0.3518	0.3791	0.4926	0.4818
	RGCN	0.3130	0.3518	0.4288	0.4369
	R-GraphSage	0.8065	0.7582	0.5008	0.4705
	Few-shot	0.8403	0.8033	0.8505	0.8136
	Fully-supervised	**0.8543**	0.8214	**0.9101**	**0.8794**

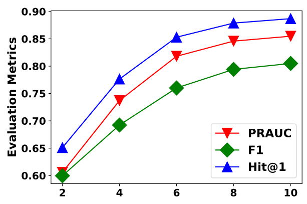
prediction baseline - **GraphSage+NN** achieves worse results than **Feature+NN**. GraphSage embedding models every edges of different types in the objective and the "equivalent" edges contributes than 1% in total. The "equivalent" relation may be biased by other links, therefore, predicts unsatisfactory results. Three entity matching baselines reports competitive performance on movie dataset but DeepMatcher performs much worse on the music dataset. Moreover, it achieves almost zero on the metric Rec@Prec.95. It may be caused by overfitting the noisy training data with huge amount of parameters(20 million). PARIS, though unsupervised, yield second-best F1 on the movie dataset, which proves the collective design works very nice on incomplete knowledge graphs. However, it can not tell the subtle difference between songs with slightly different names and duration due to lack of supervisions. Overall, Magellan and CG-MuAlign are the most robust methods under different datasets and metrics. As we know, the feature importance of random forest depends on the training data. So Magellan produces more false positives than ours, while CG-MuAlign reports above 50% recall when precision is at 95% across two datasets.

Sensitivity to the amount of training data. Then we investigate how much supervision needed in CG-MuAlign, since label scarcity is quite normal in the real applications. Also, we are interested in how collective alignment benefits from different types in this case. Therefore, we range ratio of supervision from 0 to 1.0 on type A and test it on type A on two conditions: (1) 100% training type of type B (2) 0% training type of type B. The result is plotted in Figure. 6. When we do not have any labeled data for person type, the model can achieve 0.53 F1 already and adding 10% of training data make the performance quickly converge to the final model in Figure 6a. Note that 10% of training data in our setting is about 2K samples only. When we have about 40% of training data, both settings are on a par with full supervision model. On the film type, the trends are similar but result is not that satisfactory when supervision is limited. We explain it as film alignment is more tricky since different films could have partially overlapped titles and same movies across different graphs could have multi-lingual names. Both figure shows that CG-MuAlign does not rely on large amount of training data.

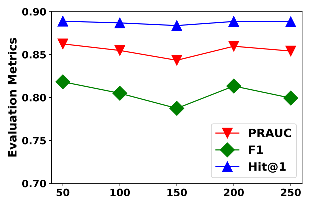
Few-shot alignment on unlabeled types. In order to investigate the generalization capability of different methods, we design few-shot alignment experiments, that first train a model on type A and fine-tune the model with only a few (*i.e.* 2,000) training pairs of type B. The model is evaluated on the test data of new type. We train and test on person and film alternatively. Magellan and DeepMatcher are trained on one type is not compatible with the new type, so we do not report their performance in Table 7. In addition, we add cosine similarity of node features as an unsupervised benchmark in the table. When tested on person, we observe the cosine similarity of feature vector is a very competitive method, since people who

Table 8: **Effectiveness of proposed attention mechanism. We report the averaged metrics on movie and music data set.**

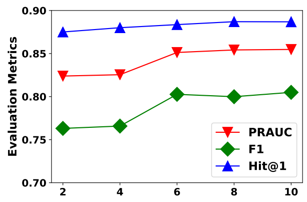
Method	Averaged-PRAUC	Averaged-F1
w.o. cross attention	0.7654	0.7034
w.o. self attention	0.8342	0.7880
CG-MuAlign	0.8474	0.7906



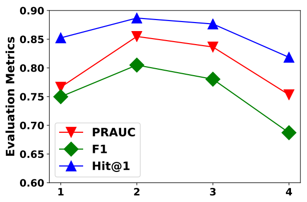
(a) Varying #sampled neighbors



(b) Varying #hidden dimensions



(c) Varying #negative samples



(d) Varying #layers

Figure 7: Parameter sensitivity of CG-MuAlign **on movie dataset. Each figure shows the result of varying the x-axis parameter.**

have the same names are likely to be the same person. Another unsupervised baseline PARIS reports promising results thanks to the collective probabilistic alignment. Most of the GNN variants report poor performance except CG-MuAlign and R-GraphSage that consider the self information (person name) separately. On type of film, few-shot CG-MuAlign achieves 81.4% F1 when feature similarity only obtains 47.8%. All other methods perform worse or slightly better than node feature similarity. The result clearly demonstrates the effectiveness of collective design, especially when the training data is limited. We want to note that alignment result reported in Table 6 are for multi-type alignment, but here the result is evaluated for each type separately. Our fully supervised model achieves better results than the reported figures in Table 6, because multi-type alignment is more challenging. Overall, our few-shot results are quite close to the fully-supervised version.

4.5 Analysis of CG-MuAlign

Effectiveness of Collective Aggregation. Overall, CG-MuAlign outperforms baselines by 10% F1 in average. We report the quantitative improvement by two different mechanism in Table 8. It shows the cross attention boost the performance by 7% in averaged-PRAUC and by 5% in averaged-F1. The self-attention further improves 1% PRAUC and F1. The cross attention contributes the major part of the performance boost. But when the training data is not that sparse, *i.e.* the music dataset, self-attention help identify the strong negative evidence.

Parameter Study. To measure the parameter sensitivity of CG-MuAlign, we evaluate the performance on movie dataset varying one of the following parameters while fixing the others to the default values mentioned above: (1) number of neighbor sampled for each node (2) hidden dimension size of each GNN layer (3) number of negative samples for each positive pair (4) number of GNN layers. The result is shown at Figure 7, where PRAUC, F1 and Hit@1 are reported under different parameters. First, the performance on three metrics improves when number of sampled neighbors, *i.e.* 2,4,6. It is mainly because many useful neighborhood information is dropped, when number of neighborhood size is small. When the neighbor window size is reasonably large, the performance tends to be similar. Then in Figure 7b, we observe the performance is similar under different hidden dimensions, which reveals that CG-MuAlign neither overfits nor relies on huge amount of parameters. Since we have the similar positive ratio in training and testing (10:1), we are also interested whether CG-MuAlign achieves good performance under different positive ratio. Result presented in Figure 7c shows the positive ratio does not affect the model performance. At last, we notice our model performs best with 2 alignment layers. More layers negatively affects the model performance, besides the running time increases as number of layers grows. The results shows that 1-hop neighborhood information is not sufficient for the task of entity alignment, while higher (> 2) order information deteriorates the performance as lots of asymmetric information appears. It is consistent with our scaling up analysis in Section 3.5.

Efficiency Study. We compare the training time, number of parameters and the average F1 score on two large-scale entity alignment datasets with two existing systems, *i.e.* Magellan [22] and DeepMatcher [27]. As shown in Table 9, CG-MuAlign achieves the best performance with the least training time. Compared with the other Deep Learning solution, *i.e.* DeepMatcher, we yield better performance with 100 times fewer parameters and 20X speed up. All of our experiments run on AWS EC2 instance with one 16G NVIDIA V100 Tensor Core GPU.

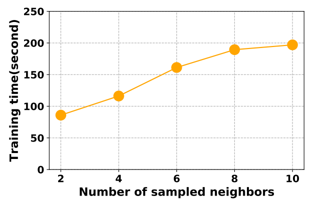
Table 9: Efficiency study of three different methods

Method	Training Time	# Parameters	Averaged-F1
Magellan	7m13s	9,300	0.6641
DeepMatcher	13h40m	17,757,810	0.6014
CG-MuAlign	30m47s	175,134	0.7925

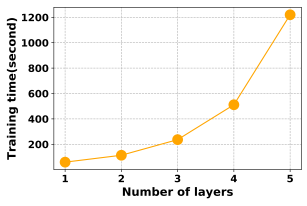
The running time of CG-MuAlign is affected by number of sampled neighbors and number of the GNN layers as discussed in Lemma 3.2. Thus, we report the training time for single epoch by varying these two parameters. In Figure 8a, we change the number of sampled neighbors while fixing #layers=2. The training time increases slowly from 80 seconds to 200 seconds, because only a few nodes have many neighbors. In Figure 8b, we set the number of neighbors as 4 and increase the number of layers in GNN. Although the running time grows exponentially, our discussion in Figure 7d supports that the 2-hop model works best in terms of the performance-efficiency trade-offs.

5 RELATED WORK

In this section, we review the literature of entity alignment from four different perspectives.



(a) Varying #sampled neighbors



(b) Varying #layers

Figure 8: Running time per epoch varying # neighbors and # layers.

Link Prediction. Entity Alignment problem can be formulated as predicting "equivalent" links among two graphs. For large-scale graph structured data, network embeddings [17, 29, 38] tranform nodes in networks into dense vectors that helps predict missing links conveniently. methods [3, 24, 35, 47] capture the first-order information and facilitate logical inference with explicit relation embeddings. To capture higher order information, previous work models the heterogenous types of nodes and edges with pre-defined meta-structures [6, 9, 13, 15, 33]. HetG [48] proposes a random walk based sampling strategy and models each type of neighborhood nodes differently. GATNE [5] studies the attributed multiplex heterogenous network embedding under transductive and inductive settings. HAN [44] aggregates neighbor nodes along meta-path via hierarchical attention mechanism.

Graph Alignment. Previous efforts on graph alignment mainly span on the transductive setting, where the entity attribute is not available. Traditional network alignment methods [40, 50] focus on graphs with a small number of relation types and optimize the alignment objective based on the topology. Taking the advantage of knowledge graph representation learning [3], embedding based methods [7, 8, 37, 51] embed entities from different space along with an alignment objective on the training data. Starting with limited alignment seeds, people propose to use either bootstrapping [37] or iteratively [51] align the entities.

Recently, graph convolutional neural network [21] sheds light on inductive graph representation learning. GAT [41] aggregates neighborhood information via multi-head attention mechanism. RGCN [32] applies multi-relational Graph Convolutional Networks and improves the performance of link prediction and node classification on knowledge graphs. GraphSage [18] introduces inductive representation learning on large scale graph data with neighborhood sampling. Besides, graph neural network methods designed for entity alignment [4, 45, 46] demonstrate great performance improvement for multi-lingual KG alignment. Although studies [4, 46] already observe the structural heterogeneity, our proposed framework is the first to leverage the collective alignment on two KGs directly.

This task is also related with the problem of graph matching. Two most recent works [1, 23] introduce cross-attention mechanism while criterion of graph matching is much more rigorous than entity alignment. Same entities in two KGs can have rather different k-hop neighborhood sub-graph, whereas it is more likely to be an unmatched case in graph matching.

Entity Matching. Entity Matching (EM) techniques [14, 22, 27] find all tuple pairs between two relational tables. It is composed by two major components: (1) blocking [10] utilizes heuristics to remove obviously non-matched pairs between two tables and (2) matching [22] predicts the match score for the remaining pairs. Megellan [22] is an open-source and scalable entity matching framework that uses handcraft features and various machine learning algorithms as the matching function such as SVM, random forest, *etc.*. Later, DeepMatcher [27] computes the pairwise similarity from the attribute embeddings and adopts deep neural network as the matching function. DeepER [14] is the first to consider sequence model like LSTM to model the textual attributes automatically. Entity matching methods mostly use 1-hop neighbor information, but CG-MuAlign can easily absorb multiple hops of neighbor entities.

Collective Entity Resolution. Researchers have noticed the correlations between labeled entities and unlabeled neighbor entities in multi-relational database and network structured data [2, 12]. Collective entity resolution considers that the decisions made on related entities are affected by each other. An unsupervised method [52] is proposed to reduce the entity resolution problem into a graph summarization problem by clustering the co-referenced entities across different knowledge graphs into a super node. People design collective features upon human-curated entity resolution rules [31]. PARIS is an unsupervised probabilistic model for ontologies alignment. HolisticEM [30] builds a graph where each node represents similarity of a pair of entities, and propagates similarity by Personalized Random Walk to make collective decisions. We carefully consider collective decisions between sub-graphs sampled around candidate pairs and boost the performance of GNN greatly.

6 CONCLUSION

In this paper, we propose CG-MuAlign to align entities of different types between two knowledge graphs. CG-MuAlign leverages both attribute information and neighborhood information during the alignment and a novel cross-graph attention mechanism is designed to deal with the data incompleteness of knowledge graphs. Our experiments show CG-MuAlign outperforms the baselines by 10% on PRAUC and F1 measure. CG-MuAlign can be generalized to entity types of low supervision/few-shot and more than 20% boost. CG-MuAlign training is highly efficient, 20 times faster than the state-of-the-art deep learning methods. The collective mechanisms shows great potential in pairwise prediction tasks. Interesting future work can be (1) extend CG-MuAlign to the transductive setting, where collective decisions need be to carefully carried out upon structural information. (2) handle multiple (> 2) knowledge graphs alignment simultaneously. We are also seeking to align entity and relation jointly in a unified GNN framework.

7 ACKNOWLEDGEMENTS

Research was sponsored in part by DARPA under Agreements No. W911NF-17-C-0099 and FA8750-19-2-1004, National Science Foundation IIS 16-18481, IIS 17-04532, and IIS-17-41317, and DTRA HDTRA11810026. Any opinions, findings, and conclusions or recommendations expressed in this document are those of the author(s) and should not be interpreted as the views of any U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] R. Al-Rfou, B. Perozzi, and D. Zelle. Ddgk: Learning graph representations for deep divergence graph kernels. In *The World Wide Web Conference*, pages 37–48. ACM, 2019.

[2] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5, 2007.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[4] Y. Cao, Z. Liu, C. Li, J. Li, and T.-S. Chua. Multi-channel graph neural network for entity alignment. *arXiv preprint arXiv:1908.09898*, 2019.

[5] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, and J. Tang. Representation learning for attributed multiplex heterogeneous network. *arXiv preprint arXiv:1905.01669*, 2019.

[6] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128. ACM, 2015.

[7] M. Chen, Y. Tian, K.-W. Chang, S. Skiena, and C. Zaniolo. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. *arXiv preprint arXiv:1806.06478*, 2018.

[8] M. Chen, Y. Tian, M. Yang, and C. Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1511–1517. AAAI Press, 2017.

[9] T. Chen and Y. Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 295–304. ACM, 2017.

[10] X. Chu, I. F. Ilyas, and P. Koutris. Distributed data deduplication. *Proceedings of the VLDB Endowment*, 9(11):864–875, 2016.

[11] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[12] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96. ACM, 2005.

[13] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 135–144. ACM, 2017.

[14] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Deeper–deep entity resolution. *arXiv preprint arXiv:1710.00597*, 2017.

[15] T.-y. Fu, W.-C. Lee, and Z. Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806. ACM, 2017.

[16] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.

[17] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[18] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[19] D. Khashabi, T. Khot, A. Sabharwal, P. Clark, O. Etzioni, and D. Roth. Question answering via integer programming over semi-structured knowledge. *arXiv preprint arXiv:1604.06076*, 2016.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[22] P. Konda, S. Das, P. Suganthan GC, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, et al. Magellan: Toward building entity matching management systems. *Proceedings of the VLDB Endowment*, 9(12):1197–1208, 2016.

[23] Y. Li, C. Gu, T. Dullien, O. Vinyals, and P. Kohli. Graph matching networks for learning the similarity of graph structured objects. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 3835–3845, 2019.

[24] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[25] C. Lockard, P. Shiralkar, and X. L. Dong. Openceres: When open information extraction meets the semi-structured web. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3047–3056, 2019.

[26] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[27] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pages 19–34. ACM, 2018.

[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[29] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[30] M. Pershina, M. Yakout, and K. Chakrabarti. Holistic entity matching across knowledge graphs. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 1585–1590. IEEE, 2015.

[31] J. Pujara and L. Getoor. Generic statistical relational entity resolution in knowledge graphs. *arXiv preprint arXiv:1607.00992*, 2016.

[32] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[33] J. Shang, M. Qu, J. Liu, L. M. Kaplan, J. Han, and J. Peng. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv preprint arXiv:1610.09769*, 2016.

[34] P. Singla and P. Domingos. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 572–582. IEEE, 2006.

[35] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[36] F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *arXiv preprint arXiv:1111.7164*, 2011.

[37] Z. Sun, W. Hu, Q. Zhang, and Y. Qu. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, pages 4396–4402, 2018.

[38] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[39] B. D. Trisedya, J. Qi, and R. Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 297–304, 2019.

[40] H. T. Trung, N. T. Toan, T. Van Vinh, H. T. Dat, D. C. Thang, N. Q. V. Hung, and A. Sattar. A comparative study on network alignment techniques. *Expert Systems with Applications*, 140:112883, 2020.

[41] P. Velivcković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[42] H. Wang, F. Zhang, X. Xie, and M. Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*, pages 1835–1844. International World Wide Web Conferences Steering Committee, 2018.

[43] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, Z. Huang, Q. Guo, H. Zhang, H. Lin, J. Zhao, J. Li, A. J. Smola, and Z. Zhang. Deep graph library: Towards efficient and scalable deep learning on graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[44] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2022–2032, 2019.

[45] Z. Wang, Q. Lv, X. Lan, and Y. Zhang. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357, 2018.

[46] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. *arXiv preprint arXiv:1905.11605*, 2019.

[47] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[48] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla. Heterogeneous Graph Neural Network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '19*, pages 793–803. ACM Press, 2019.

[49] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, and Y. Qu. Multi-view knowledge graph embedding for entity alignment. *arXiv preprint arXiv:1906.02390*, 2019.

[50] S. Zhang and H. Tong. Attributed network alignment: Problem definitions and fast solutions. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[51] H. Zhu, R. Xie, Z. Liu, and M. Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pages 4258–4264, 2017.

[52] L. Zhu, M. Ghasemi-Gol, P. Szekely, A. Galstyan, and C. A. Knoblock. Unsupervised entity resolution on multi-type graphs. In *International semantic web conference*, pages 649–667. Springer, 2016.