What Makes Entities Similar? A Similarity Flooding Perspective for Multi-sourced Knowledge Graph Embeddings

Zequn Sun [1] **Jiacheng Huang** [1] **Xiaozhou Xu** [2] **Qijin Chen** [2] **Weijun Ren** [2] **Wei Hu** [1 3]

Abstract

Joint representation learning over multi-sourced knowledge graphs (KGs) yields transferable and expressive embeddings that improve downstream tasks. Entity alignment (EA) is a critical step in this process. Despite recent considerable research progress in embedding-based EA, how it works remains to be explored. In this paper, we provide a similarity flooding perspective to explain existing translation-based and aggregation-based EA models. We prove that the embedding learning process of these models actually seeks a fixpoint of pairwise similarities between entities. We also provide experimental evidence to support our theoretical analysis. We propose two simple but effective methods inspired by the fixpoint computation in similarity flooding, and demonstrate their effectiveness on benchmark datasets. Our work bridges the gap between recent embedding-based models and the conventional similarity flooding algorithm. It would improve our understanding of and increase our faith in embedding-based EA.

1. Introduction

A knowledge graph (KG) is a set of relational triplets. Each triplet is in the form of (*subject entity*, *relation*, *object entity*), denoted by (s, r, o) for short. A relational triplet indicates a relation between two entities, such as (*ICML 2023*, *hosted in*, *Hawaii*). Different KGs are created by harvesting various webs of data. They could cover complementary knowledge from different sources and thus aid in resolving the incompleteness issue of each single KG. In recent years, representing multi-sourced KGs in a unified embedding space, as illustrated in Figure 1, has shown promising potential

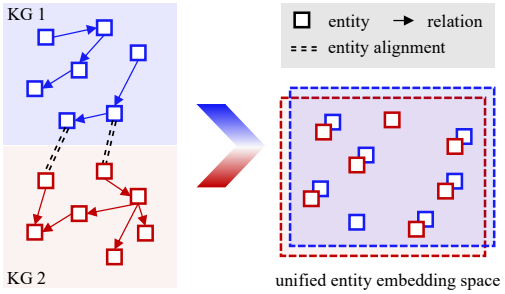

Figure 1. Illustration of representing two KGs in a unified space.

in promoting knowledge fusion and transfer (Trivedi et al., 2018). It uses entity alignment (EA) between different KGs to jump-start joint and transferable representation learning. EA refers to the match of identical entities from different KGs, such as "*ICML*" and "*International Conference on Machine Learning*". The goal of multi-sourced KG embedding is learning to distinguish between identical and dissimilar entities in different KGs while capturing their respective graph structures. By aligning the embeddings of identical entities, an entity in one KG can indirectly capture the graph structures of its counterpart in another KG, resulting in more informative representations to benefit downstream tasks.

Therefore, as a fundamental task, embedding-based EA has drawn increasing attention (Chen et al., 2017; Guo et al., 2019; Sun et al., 2020b; Zhao et al., 2022; Zeng et al., 2021a; Zhang et al., 2022; Guo et al., 2022). The key of embedding-based EA lies in how to generate entity embeddings for alignment learning. Existing techniques fall into two groups, translation-based (Chen et al., 2017; Sun et al., 2017; 2019) and aggregation-based models. A translation-based model adopts TransE (Bordes et al., 2013) or its variants for embedding learning. Given a triplet (s, r, o), TransE interprets a relation embedding as the translation vector from the subject entity embedding to the object entity. Another group of EA models uses graph convolutional networks (GCNs) (Kipf & Welling, 2017) to generate an entity representation by aggregating its neighbor embeddings.

Despite the considerable technical progress in embedding-based EA, a critical question remains unanswered, i.e., *what makes entity embeddings similar in an EA model?* This

[1]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China [2]Alibaba Group, Hangzhou, China [3]National Institute of Healthcare Data Science, Nanjing University, Nanjing, China. Correspondence to: Wei Hu <whu@nju.edu.cn>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

question may also cause some researchers to misunderstand and distrust embedding-based EA techniques. Besides, the connection between embedding-based EA models and traditional symbolic methods remains unexplored. Under the circumstances, we seek to answer the question. We present a similarity flooding (SF) perspective to understand and improve embedding-based EA with both theoretical analysis and experimental evidence. SF is a widely-used algorithm for matching structured data (Melnik et al., 2002; 2001). We show that the essence of recent embedding-based EA is also a variant of SF, and learning embeddings is only a means.

Our main contributions are summarized as follows:

- We present the first theoretical analysis of embedding-based EA techniques to understand how they work. We provide a similarity flooding perspective to unify the basic translation- and aggregation-based EA models. We also build a close connection between embedding-based and traditional symbolic-based EA via the unified perspective of fixpoint computation for entity similarities. This work would improve our understanding of and increase our faith in embedding-based models.
- We propose two simple but effective methods based on our theoretical analysis to improve EA. The first is a variant of similarity flooding that computes the fixpoint of entity similarities using the entity compositions induced from TransE or GCN. This method does not need to learn KG embeddings. The second is inspired by the fact that the similarity fixpoint indicates an embedding fixpoint. It introduces a self-propagation connection in neighborhood aggregation to let entity embeddings have a chance of propagating back to themselves.
- We conduct experiments on DBP15K (Sun et al., 2017) and OpenEA (Sun et al., 2020b) to validate the effectiveness of our EA methods and provide experimental evidence to support our theoretical conclusions. The source code is available at our GitHub repository[1].

2. Preliminaries

We first introduce the EA task, and then discuss the basic translation-based and aggregation-based models. We would like to know how to represent an entity in these models so that we can learn more about what factors influence entity similarities. Finally, we introduce the SF algorithm.

2.1. Problem Definition

Formally, let $\mathcal{X}$ and $\mathcal{Y}$ be the entity sets of the source and target KGs, respectively. In the supervised setting, we are given a set of seed entity alignment pairs $\mathcal{A}$ as training data. For an aligned entity pair $(x, y) \in \mathcal{A}$, where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, the KG embeddings for EA are expected to hold: $\mathbf{x} = \arg\min_{x' \in \mathcal{X}} \pi(\mathbf{x}', \mathbf{y})$. Hereafter, we use boldface type to denote vector embeddings, e.g., $\mathbf{x}$ and $\mathbf{y}$ for the embeddings of x and y, respectively. $\pi(\mathbf{x}, \mathbf{y})$ is a distance measure. In this paper, we consider the Euclidean distance, i.e., $\pi(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2$, where $||\cdot||_2$ denotes the L_2 vector norm. It indicates that, if x and y are aligned entities, $\mathbf{y}$ is expected to be the nearest cross-KG neighbor of $\mathbf{x}$ in the embedding space. To achieve this goal, given a small set of seed alignment, $\mathcal{A} \subset \{(x, y) | x \equiv y\}$, as training data, the general objective of alignment learning is to minimize the embedding distance of entity pairs in $\mathcal{A}$ (Chen et al., 2017):

$$\min_{(x,y)\in\mathcal{A}} \pi(\mathbf{x}, \mathbf{y}). \tag{1}$$

Although many models introduce various negative sampling methods (Sun et al., 2018) to generate dissimilar entity pairs and learn to separate the embeddings of dissimilar entities, Eq. (1) is the most common and indispensable learning objective, which is our focus in this paper.

2.2. TransE-based EA

The typical learning objective of TransE-based models is to solve two optimization problems, i.e., translational embedding learning and alignment learning, as shown in Eqs. (1) and (2), respectively.

$$\min_{(s,r,o)\in\mathcal{T}} \|\mathbf{s} + \mathbf{r} - \mathbf{o}\|_2^2, \quad \text{s.t. } \|\mathbf{e}\|_2^2 = 1, \forall\, e \in \mathcal{X} \cup \mathcal{Y}, \tag{2}$$

where $\mathcal{T}$ is the set of triplets and e denotes an entity. Considering that the two optimization problems have a trivial optimal solution with all entities and relations having zero vectors, most EA models normalize each entity embedding to a unit vector. Therefore, we further introduce a Lagrange term $\lambda_e \sum_{e\in\mathcal{X}\cup\mathcal{Y}} (||\mathbf{e}||_2^2 - 1)$, where λ_e is the Lagrange multiplier. The combined optimization problem is

$$\mathcal{L}(\mathbf{\Theta}) = \sum_{(s,r,o)\in\mathcal{T}} \|\mathbf{s} + \mathbf{r} - \mathbf{o}\|_2^2 + \sum_{(x,y)\in\mathcal{A}} \|\mathbf{x} - \mathbf{y}\|_2^2 + \lambda_e \sum_{e\in\mathcal{X}\cup\mathcal{Y}} (||\mathbf{e}||_2^2 - 1), \tag{3}$$

where $\mathbf{\Theta}$ denotes the entity and relation embeddings. The optimization problem then shifts to solving the following equation: $\bigtriangledown_{\mathbf{\Theta},\lambda_e} \mathcal{L}(\mathbf{\Theta}) = \mathbf{0}$. Then, we can derive the representations of relations and entities in the model.

Deriving relation representations. We first consider relation embeddings and take the relation r as an example. We are interested in the gradients of the loss in Eq. (3) with respect to r: $\bigtriangledown_{\mathbf{r}} \mathcal{L}(\mathbf{\Theta}) = \bigtriangledown_{\mathbf{r}} \sum_{(s,r,o)\in\mathcal{T}_r} \|\mathbf{s} + \mathbf{r} - \mathbf{o}\|_2^2$, where $\mathcal{T}_r$ denotes the set of triplets involving r. Letting the above derivative be zero, we can derive

$$\mathbf{r} = \frac{1}{|\mathcal{T}_r|} \sum_{(s,r,o)\in\mathcal{T}_r} (\mathbf{o} - \mathbf{s}). \tag{4}$$

The equation aligns with the motivation of TransE that represents a relation as the translation vector between its subject

[1] https://github.com/nju-websoft/Unify-EA-SF

and object entity embeddings. Given this equation, we can use the final entity embeddings to represent a relation.

Deriving entity representations. An entity may appear as the subject or object in a triplet. To simplify the formulations without information loss, we introduce reverse triplets following the convention in KG embedding models (Guo et al., 2019). For each triplet (s, r, o), we add a new triplet (o, r^{-1}, s), where r^{-1} denotes the reverse relation of r. In this way, we only need to consider the outgoing edges of an entity, i.e., the triplets with the given entity as the subject. The original ingoing edges are considered by including their reverse edges. We use $\mathcal{T}_e$ to denote the triplets with e as the subject. Specifically, given entity e, we are interested in the gradients of the loss in Eq. (3) with respect to embedding $\mathbf{e}$, i.e., $\bigtriangledown_{\mathbf{e}}\mathcal{L}(\mathbf{\Theta}) = \bigtriangledown_{\mathbf{e}} \sum_{(e,r,o)\in\mathcal{T}_e} \|\mathbf{e}+\mathbf{r}-\mathbf{o}\|_2^2 + \mathbb{1}_{\exists(e,\hat{e})\in\mathcal{A}} \bigtriangledown_{\mathbf{e}} \|\mathbf{e}-\hat{\mathbf{e}}\|_2^2 + \lambda_e \bigtriangledown_{\mathbf{e}} (||\mathbf{e}||_2^2 - 1)$, where $\mathbb{1}$ is an indicator function. By setting the gradients to be zero vectors, we obtain $\mathbf{e} = \frac{1}{|\mathcal{T}_e|+\lambda_e} \sum_{(e,r,o)\in\mathcal{T}_e} (\mathbf{o}-\mathbf{r}) + \mathbb{1}_{\exists(e,\hat{e})\in\mathcal{A}}(\mathbf{e}-\hat{\mathbf{e}})$. With proper EA training strategies, e.g., parameter sharing (Sun et al., 2020b), e and $\hat{e}$ would have the same embeddings, i.e., $\mathbf{e}-\hat{\mathbf{e}} = \mathbf{0}$. In addition, we can apply normalization to $\mathbf{e}$ to ensure $\|\mathbf{e}\| = 1$, and then we can replace $|\mathcal{T}_e| + \lambda_e$ with $|\mathcal{T}_e|$. Thus, we obtain $\mathbf{e} = \frac{1}{|\mathcal{T}_e|} \sum_{(e,r,o)\in\mathcal{T}_e} (\mathbf{o}-\mathbf{r})$. Note that, in this equation, we still need relation embeddings to represent an entity. To get free of relation embeddings, we can replace them with the composition of related subject and object entity embeddings as shown in Eq. (4), and get

$$\mathbf{e} = \frac{1}{|\mathcal{T}_e|} \sum_{(e,r,o)\in\mathcal{T}_e} \Big(\mathbf{o} - \frac{1}{|\mathcal{T}_r|} \sum_{(s',r,o')\in\mathcal{T}_r} (\mathbf{o}' - \mathbf{s}')\Big). \quad (5)$$

In this way, we represent an entity by the composition of its related entities in the same KG.

2.3. GCN-based EA

In a CGN-based EA method, an entity is first represented by aggregating its neighbors. For brevity, we consider a one-layer GCN layer (Kipf & Welling, 2017) with mean-pooling as the aggregation function, i.e., $G(x) = \frac{1}{|N(x)|} \sum_{x'\in N(x)} \mathbf{x}'$. The entity representation in GCNs is:

$$\mathbf{e} = \frac{1}{|N(e)|} \sum_{e'\in N(e)} \mathbf{e}'. \quad (6)$$

Then, given the output representations, we minimize the embedding distance of identical entities in seed entity alignment for alignment learning, as shown in Eq. (1). Finally, we use kNN search to find the counterpart for a given entity.

2.4. Similarity Flooding

Similarity flooding (Melnik et al., 2002) is an iterative graph matching technique based on fixpoint computation.

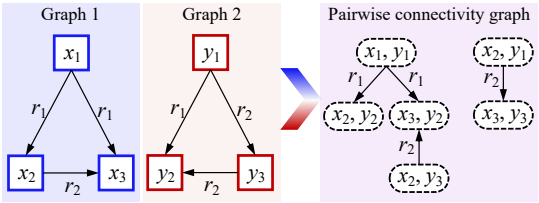

Figure 2. Illustration of how to build the pairwise connectivity graph given two graphs (redrawn based on (Melnik et al., 2002)).

It is a fundamental algorithm and widely used in a variety of graph matching contexts, such as ontology mapping and database schema matching (Shvaiko & Euzenat, 2013). Given two input graphs G_1 and G_2 with the aim of finding the mapping of identical nodes, the similarity flooding algorithm first creates a pairwise connectivity graph (PCG), which is an auxiliary data structure for similarity propagation. As shown in Figure 2, in a PCG, a node is an entity pair (x_1, y_1) with the similarity of $\sigma(x_1, y_1)$ (called a mapping pair), where the two entities are from the two graphs, respectively, i.e., $x_1 \in G_1$ and $y_1 \in G_2$. An edge $\big((x_1, y_1), r_1, (x_2, y_2)\big)$ of the PCG is induced from the two graphs having $(x_1, r_1, x_2) \in G_1$ and $(y_1, r_1, y_2) \in G_2$. The relation r_1 would be further given a weight, called the propagation coefficient, which ranges from 0 to 1 and can be computed in different ways (Melnik et al., 2001). The directed weighted edge $\big((x_1, y_1), r_1, (x_2, y_2)\big)$ indicates how well the similarity of (x_1, y_1) propagates to its neighbor (x_2, y_2). Then, the algorithm propagates the similarity of each node (i.e., mapping pair) over the PCG using fixpoint computation and finally outputs the node mappings. The fixpoint formula for similarity flooding is

$$\Omega = \texttt{normalize}\big(\Omega_0 + \Omega + \varphi(\Omega_0 + \Omega)\big), \quad (7)$$

where Ω_0 is the node similarity matrix, and φ is the propagation function. In conventional graph matching methods, Ω_0 can be computed by string matching. In our work, we follow the supervised setting of embedding-based EA, and use seed entity alignment to initialize Ω_0.

Remark 2.1. The pairwise connectivity graph construction requires the alignment of edge labels in the two graphs.

Remark 2.2. The propagation coefficients of edges in the pairwise connectivity graph are computed heuristically.

3. Connecting Embedding-based EA and SF

Given the derived entity representations from TransE or GCN, we can compute entity similarities. Specifically, given two entity sets, $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, y_2, \ldots, y_m\}$, we denote the derived entity representations by $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ and $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m\}$, respectively. Their pairwise similarity matrix is

$$\Omega = (\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_n)^\top (\mathbf{y}_1; \mathbf{y}_2; \ldots; \mathbf{y}_m) \in \mathbb{R}^{n\times m}. \quad (8)$$

The similarity matrix determines entity alignment pairs.

3.1. Unifying TransE- and GCN-based EA

Theorem 3.1. *The TransE-based EA model seeks a fixpoint of pairwise entity similarities via embedding learning.*

Proof. Eq. (5) shows that we can represent an entity with a composition of other entities. Thus, we can represent an entity (taking $x_i \in \mathcal{X}$ for example) as

$$\mathbf{x}_i = \lambda_{i,1}\mathbf{x}_1 + \lambda_{i,2}\mathbf{x}_2 + \cdots + \lambda_{i,n}\mathbf{x}_n = \sum_{k=1}^{n} \lambda_{i,k}\mathbf{x}_k, \quad (9)$$

where $\lambda_{i,k}$ denotes the composition coefficient of entities x_i and x_k, which can be computed from Eq. (5). Then, the similarity of entities $x_i \in \mathcal{X}$ and $y_j \in \mathcal{Y}$ can be calculated using the inner product[2] as follows:

$$\omega_{i,j} = \mathbf{x}_i \cdot \mathbf{y}_j = \sum_{k=1}^{n}\sum_{l=1}^{m} \lambda_{i,k}\lambda'_{j,l}\, \mathbf{x}_k \cdot \mathbf{y}_l = \sum_{k=1}^{n}\sum_{l=1}^{m} \lambda_{i,k}\lambda'_{j,l}\omega_{k,l}. \quad (10)$$

We can see from the above equation how the similarity of two entities affects their related neighbors. Let the matrix $\Lambda = (\lambda_{i,j})_{i=1,j=1}^{n,n}$ consist of the lambda values for the source KG, and $\Lambda' = (\lambda'_{i,j})_{i=1,j=1}^{m,m}$ for the target KG. Let $\Omega = (\omega_{i,j})_{i=1,j=1}^{n,m}$ denote the pairwise entity similarities of the two KGs. We can rewrite Eq. (10) as

$$\Lambda\Omega(\Lambda')^\top = \Omega, \quad (11)$$

where $(\Lambda')^\top$ is the transposed matrix of Λ'. This equation shows that the entity embedding similarities learned by the translation-based model have a fixpoint of Ω. □

Next, we can connect aggregation-based EA models and similarity flooding in a similar way.

Theorem 3.2. *The GCN-based EA model seeks a fixpoint of pairwise entity similarities via embedding learning.*

Proof. Please refer to the proof of Theorem 3.1. The difference lies in how to compute the lambda values. □

We show how to calculate lambda values below.

Lambda values for TransE. The lambda values for TransE are computed by counting the number of related triples:

$$\lambda_{i,j} = \frac{1}{|\mathcal{T}_{x_i}|}\Big(|R(x_i,x_j)| + \sum_{r\in R}\frac{|\mathcal{T}_{x_i,r}|}{|\mathcal{T}_r|}\big(|\mathcal{T}_{x_j,r}| - |\mathcal{T}_{x_j,r^{-1}}|\big)\Big), \quad (12)$$

where $R(x_i, x_j)$ denotes the set of relations that connect entities x_i and x_j, and R is the set of all relations. $\mathcal{T}_{x_i,r}$ denotes the set of relation triplets with x_i as the subject and r as the relation. $\mathcal{T}_r$ is the set of relation triplets with r as the relation. r^{-1} is the reverse relation for r.

[2] The inner product of two normalized vectors is equal to their cosine similarity.

Lambda values for GCN. For GCN, we have

$$\lambda_{i,j} = \frac{\mathbb{1}_{(x_i,r,x_j)\in\mathcal{T}}}{|\mathcal{T}_{x_i}|}, \quad (13)$$

where $\mathbb{1}$ is an indicator function that returns 1 if there is a relation between x_i and x_j, and 0 otherwise.

Remark 3.3. Embedding learning is just a means and the objective is to seek a fixpoint of pairwise entity similarities.

3.2. An Interpretation of Embedding-based EA

Given the fixpoint view of EA, we further discover a mathematical interpretation of TransE- and GCN-based models. We show that identical entities have isomorphic structures in the entity compositions of embedding-based EA.

Theorem 3.4. *The entity alignment pairs found by the above embedding-based models yield a function* $f : \{1, 2, \ldots, n\} \to \{0, 1, 2, \ldots, m\}$*, such that* $\forall i, j, f(i) > 0 \wedge f(j) > 0 \to \lambda'_{f(i),f(j)} \approx \lambda_{i,j}$.

Proof. Let us consider aligning $\mathcal{Y}$ with itself. We have

$$\Lambda'\mathbf{I}_m(\Lambda')^\top \approx \mathbf{I}_m, \quad (14)$$

where $\mathbf{I}_m$ is an identity matrix. Suppose that the alignment found by the above embedding-based models is $\hat{\mathcal{A}}$, which can be denoted by a 0-1 matrix $\hat{\Omega}$ such that $\hat{\omega}_{i,j} = 1$ if and only if $(x_i, x_j) \in \hat{\mathcal{A}}$. Similar to most EA settings, we assume that in $\hat{\mathcal{A}}$, each entity is aligned to at most one entity in another KG. Notice that $\hat{\Omega}$ approximately equals a fixpoint of Eq. (7). Thus, we have

$$\hat{\Omega}^\top\Lambda\hat{\Omega}(\Lambda')^\top \approx \hat{\Omega}^\top\hat{\Omega} = \hat{\mathbf{I}}_m, \quad (15)$$

where $\hat{\mathbf{I}}_m$ is a diagonal matrix, where $\hat{\mathbf{I}}_{j,j} = 1$ if and only if y_j appears in one pair in $\hat{\mathcal{A}}$. As $\Lambda'\mathbf{I}_m(\Lambda')^\top \approx \mathbf{I}_m$, we have $\hat{\Omega}^\top\Lambda\hat{\Omega} \approx \hat{\mathbf{I}}_m\Lambda'$. Let f be a function defined as

$$f(i) = \begin{cases} j, & (x_i, y_j) \in \hat{\mathcal{A}} \\ 0, & \forall y_j \in \mathcal{Y}, (x_i, y_j) \notin \mathcal{A} \end{cases}. \quad (16)$$

When $f(i) > 0$ and $f(j) > 0$, we have $(\hat{\Omega}^\top\Lambda\hat{\Omega})_{f(i),f(j)} = \lambda_{i,j}$, i.e., $\lambda'_{f(i),f(j)} \approx \lambda_{i,j}$. □

Based on Theorem 3.4, we find that for each KG, the entity compositions derived from EA models generate a matrix (e.g., Λ) that only depends on graph structures. It finds a mapping function that makes the two KGs' matrices the same and this function determines the alignment results. Although different KGs may have heterogeneous structures, the entity compositions in embedding-based EA models reconstruct a new structure (represented by Λ), in which aligned entities have isomorphic subgraphs.

Remark 3.5. If we view these matrices as edge weights between nodes in KGs, these embedding-based EA models mathematically conduct graph matching.

Algorithm 1 Similarity flooding via entity compositions

Input: KG_1, KG_2, seed entity alignment $\mathcal{A}$, the maximum number of iterations T, a small threshold value ϵ for algorithm termination, an embedding model $\mathcal{M}$.
while true **do**
  Derive the entity compositions from M;
  Compute lambda matrices Λ and Λ' in the compositions;
  $\Omega_0 \leftarrow (0)_{i=1,j=1}^{n,m}$;
  for $i, j \in \mathcal{A}$ **do**
    $\Omega_{0\ i,j} \leftarrow 1$;
  end for
  for $t = 1, 2, \ldots, T$ **do**
    $\Omega_t \leftarrow \texttt{normalize}\big(\Lambda\Omega_{t-1}(\Lambda')^\top\big)$;
  end for
  if $\triangle(\Omega_t, \Omega_{t-1}) < \epsilon$ **then**
    break
  end if
end while

4. Experimental Evidence

In this section, we propose two methods to improve EA: similarity flooding via entity compositions and self-propagation in GCNs. We evaluate them on benchmark datasets, providing experimental evidence to support our theorem.

4.1. Similarity Flooding via Entity Compositions

We have shown by Eqs. (5) and (6) that the entity representations derived from TransE- and GCN-based models can be reformulated to be independent from relations. Our theorems in Section 3 show that entity similarities are determined by other entity similarities, and entity embeddings are unnecessary in this computation. Then, a natural question arises: *Is the embedding learning process a prerequisite for achieving the fixpoint of entity similarities*?

Given Eq. (11), we design a similarity flooding style algorithm to propagate the entity similarities that are computed based on the entity composition representations induced from an embedding model. It is presented in Algorithm 1. We first derive the entity compositions from the embedding model. Then, we calculate the lambda values Λ and Λ' in the compositions. We initialize the similarity matrix Ω to be a zero matrix and set the values that indicate seed EA similarities to be 1. The similarity matrix is further updated to achieve the fixpoint as shown in Eq. (11). After each update, we normalize the values to range from -1 to 1. The computation is performed in an iterative manner until it converges or reaches the maximum number of iterations.

We hereby discuss the advantages of the algorithm. First, it does not need relation alignment. It represents an entity without using relations and does not need to build the PCG. As different KGs usually have heterogeneous schemata, it is difficult to obtain the accurate relation alignment. By contrast, the conventional similarity flooding algorithm relies on relation alignment to build the PCG. Second, our algorithm does not need to compute propagation coefficients for similarity flooding. The lambda values act as "propagation coefficients", but they are calculated by counting the number of related triplets without using heuristic methods. Third, our algorithm does not need to learn embeddings, but it needs an embedding model to derive the entity compositions. Our optimization objective is to directly achieve the fixpoint of entity similarities. Embedding-based models seek this goal by an indirect way of updating embeddings.

Our algorithm has the disadvantage of requiring matrix manipulation. If the KG scale is large, it would consume a lot of memory. We can solve this problem by using advanced and parallel matrix manipulation implementations.

4.1.1. Evaluation

We implement two variants of our algorithm, namely TransFlood and GCNFlood, using TransE and GCN, respectively.

Baselines. We choose the translation-based model MTransE and aggregation-based model GCN-Align as baselines.

- **MTransE** (Chen et al., 2017) is one of the earliest studies that explore translational embeddings for EA. It uses TransE (Bordes et al., 2013) to learn the entity embeddings of two KGs meanwhile learning a linear mapping to find identical entities.
- **GCN-Align** (Wang et al., 2018) is the first work that considers GCNs for KG EA. It employs the vanilla GCN (Kipf & Welling, 2017) to generate entity embeddings and uses the marginal ranking loss with uniform negative sampling for alignment learning.

Datasets. We consider two datasets in our experiment. One is the widely-used dataset DBP15K (Sun et al., 2017) It aims to align the cross-lingual entities extracted from DBpedia (Lehmann et al., 2015). It has three EA settings: ZH-EN (Chinese-English), JA-EN (Japanese-English) and FR-EN (French-English). The triples in these KGs are extracted from the infobox data of multilingual Wikipedia. They have similar rather than identical schemata because the data is not mapped to a unified ontology. Each setting has $15,000$ pairs of identical entities for alignment learning and test. We follow the data splits of DBP15K and use 30% of entity alignment as training data. The other dataset is OpenEA (Sun et al., 2020b) and we choose its 15K V1 versions of D-W (DBpedia-Wikidata) and D-Y (DBpedia-YAGO), in each of which the two KGs have different schemata. Each setting also has $15,000$ entity alignment pairs and we follow its data splits and use 20% of entity alignment as training data.

Metrics. Following the conventions, we choose Hits@k ($k = 1, 10$) and mean reciprocal rank (MRR) as metrics to assess EA performance. Hits@k measures the proportion

Table 1. EA results on DBP15K as well as OpenEA D-W and D-Y. The best scores in each group are marked in bold. The results of MTransE are taken from (Sun et al., 2017). The results of GCN-Align are taken from its paper. "-" denotes their unreported metrics.

Models	DBP15K ZH-EN			DBP15K JA-EN			DBP15K FR-EN			OpenEA D-W 15K			OpenEA D-Y 15K		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE	0.308	0.614	-	0.279	0.575	-	0.244	0.556	-	0.259	-	0.354	0.463	-	0.559
TransFlood (ours)	**0.315**	**0.707**	0.451	**0.372**	**0.757**	0.505	**0.347**	**0.752**	0.484	**0.294**	0.699	**0.427**	**0.503**	0.880	**0.641**
GCN-Align	**0.413**	0.744	-	**0.399**	0.745	-	**0.373**	0.745	-	**0.364**	-	0.461	0.465	-	0.536
GCNFlood (ours)	0.349	**0.761**	0.490	0.376	**0.770**	0.512	0.349	**0.761**	0.490	0.358	0.739	**0.486**	**0.478**	0.754	**0.583**

of correctly-aligned entities ranked in the top k. MRR is the average of the reciprocal ranks. Higher Hits@k and MRR scores indicate better performance.

Main results. We present the results in Table 1. We can observe that the proposed TransFlood achieves much better performance than MTransE on all datasets. For example, on FR-EN, the Hits@1 score of TransFlood is 0.347, outperforming MTransE by 0.103. We find that, as a learning model, MTransE is easy to overfit. Our model is also derived from TransE but our iteration algorithm can enable our model to get a more stable solution than the learning method. For aggregation-based EA, our GCNFlood achieves comparative Hits@1 results and better Hits@10 scores compared to GCN-Align. Our GCNFlood only considers one-hop neighbors to generate entity similarities (i.e., a one-layer GCN), whose information is less than that in GCN-Align (a two-layer GCN). However, its advantage lies in that it converges directly to the fixpoint, while the embedding learning method cannot guarantee this. Overall, TransFlood and GCNFlood that do not need learning can achieve comparable or even better performance than embedding learning baselines.

Running time comparison. We compare the running time of our algorithm variants against MTransE and GCN-Align on ZH-EN. This experiment is conducted using a personal workstation with an Intel Xeon E3 3.3GHz CPU, 128GB memory and a NVIDIA GeForce GTX 1080Ti GPU. The results are shown in Figure 3. We observed similar results on the other two datasets. MTransE uses the least time because it is a shallow model that can be easily optimized. GCN-Align takes the most time. We find that it converges very slowly and takes many training epochs. Our TransFlood and GCNFlood take very similar time, which is also less than that of GCN-Align. In our algorithm, resolving Eq. (11) costs the most in training time. Overall, our algorithm, which does not need to learn embeddings, can achieve comparable or even better performance in both effectiveness and efficiency than embedding learning models.

Results using text features. Our similarity flooding algorithm can also use text features to improve performance. We use multilingual word embeddings (Bojanowski et al., 2017) to encode entity names for computing the similarity matrix, which is further combined with Ω in our Algorithm 1. We

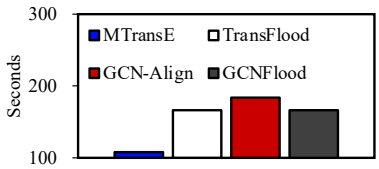

Figure 3. Total running time (in seconds) on ZH-EN.

conduct experiments on DBP15K and present the results in Table 2. We choose RDGCN (Wu et al., 2019) as a baseline. We can see that our TransFlood + Text and GCNFlood + Text achieve slightly lower results than RDGCN. On FR-EN, TransFlood + Text achieves comparable results with RDGCN. Moreover, by using text features, both TransFlood and GCNFlood get greatly improved. These results show the generalization ability of our algorithm.

4.2. Self-propagation in Neighborhood Aggregation

Based on our theoretical analysis of embedding-based EA and similarity flooding, we derive a new aggregation scheme for EA: self-propagation and neighbor aggregation.

As previously stated, an embedding-based EA model aims to establish a fixpoint of pairwise entity similarities by updating entity embeddings throughout the training process. Considering that entity similarities are computed using entity embeddings, the output of GCNs also achieves a fixpoint. We can rewrite neighborhood aggregation as

$$\mathbf{e} = f\big(\mathbf{e}, \oplus_{z \in N_e}(\mathbf{z})\big), \tag{17}$$

which means that the entity embeddings remain "unchanged" after aggregation. For brevity, we use the function $G()$ to denote a GCN layer and consider a two-layer GCN. Given input embedding $\mathbf{e}^0$, in the fixpoint, we expect to hold

$$\mathbf{e}^2 = G(\mathbf{e}^1) = \mathbf{e}^1 = G(\mathbf{e}^0) = \mathbf{e}^0. \tag{18}$$

However, this equation has limitations. First, in this case, the aggregation function degenerates into an identity mapping. Second, it almost loses the neighborhood information. To resolve the issues, inspired by (Klicpera et al., 2019), we enable the GCN output to have a probability of backing to the input. The aggregation function is rewritten as:

$$\mathbf{e}^{i+1} = (1-\alpha) \oplus_{z \in N_e} (\mathbf{z}) + \alpha f(\mathbf{e}^i), \tag{19}$$

Table 2. EA results using text features on DBP15K.

Models	ZH-EN			JA-EN			FR-EN		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
RDGCN (Wu et al., 2019)	0.708	0.846	-	0.767	0.895	-	0.886	0.957	-
TransFlood	0.315	0.707	0.451	0.372	0.757	0.505	0.347	0.752	0.484
GCNFlood	0.349	0.761	0.490	0.376	0.770	0.512	0.349	0.761	0.490
TransFlood + Text	0.670	0.786	0.713	0.747	0.868	0.794	0.881	0.949	0.908
GCNFlood + Text	0.651	0.823	0.716	0.712	0.882	0.777	0.842	0.957	0.887

where α is a hyper-parameter indicating the probability of backing to the input. Here, we use $f()$ to denote a dense layer. $\mathbf{e}^0$ is randomly initialized as the input embedding of entity e. The output of the GCN, i.e., $\mathbf{e}^2$ for a two-layer GCN, is used for alignment learning and search. Please note that, although the conventional GCNs also consider the entity itself in neighbor aggregation, the work (Klicpera et al., 2019) shows that they would still lose the local focus of the entity itself during layer-by-layer aggregation.

4.2.1. Properties of Self-propagation

Taking a deep learning perspective, we find that the proposed self-propagation has several good properties.

Model complexity. The proposed self-propagation can be easily combined with any aggregation function without adding additional computational complexity. It only introduces a dense layer for feature transformation. Considering that the number of entity embedding parameters is much larger than that of a dense layer, we argue that the parameter complexity of self-propagation remains similar to that of other aggregation functions.

Relation to PageRank-based GCNs. PageRank-based GCNs introduce the possibility of resetting the neighborhood aggregation to its initial state during training (Klicpera et al., 2019; Roth & Liebig, 2022). These studies are relevant to random walks with restarts on graphs where the random walk has a probability of backing to the start node after several steps. The idea is similar to ours. The difference is that we do not seek the representation of an entity to return to itself after several times of neighborhood aggregation. Instead, we seek to increase the local focus on the entity representation itself within the iterative neighborhood aggregation. Self-propagation is also helpful to resolve the over-smoothing issue.

Relation to residual learning. The self-propagation can be regarded as a special case of residual learning (He et al., 2016) because it builds a skipping connection between two GCN layers. Given the input $\mathbf{x}$, let $F(\mathbf{x})$ be a representation function (e.g., the $G()$ in our paper), and $H(\mathbf{x})$ be the expected output representation. Residual learning indicates that directly optimizing $F(\mathbf{x})$ to fit $H(\mathbf{x})$ is more difficult than letting $F(\mathbf{x})$ fit the residual part $H(\mathbf{x})-\mathbf{x}$. For aggregation-based EA, we cannot let $H(\mathbf{x}) = \mathbf{x}$, in which case the function $F()$ has no representation ability. Therefore, we introduce the transformation function $f()$, and let $G(\mathbf{x})$ fit $H(\mathbf{x}) - f(\mathbf{x})$. A related work (Guo et al., 2019) shows that the skipping connection would also improve the optimization of KG embeddings.

4.2.2. Evaluation

We present our experimental results on DBP15K and OpenEA in terms of Hits@1, Hits@10 and MRR scores.

Implementation. The performance of an EA model relates to not only the embedding learning model (e.g., TransE or GCN) but also other modules, including the alignment learning loss, the negative sampling method, and even the tricks in deep learning such as the parameter initialization method and the loss optimizer. To study the real effectiveness of self-propagation, we do not develop a new aggregation-based model from scratch. Instead, we choose four representative aggregation-based models: GCN-Align (see Section 4.1.1), AliNet, Dual-AMN and RoadEA, and incorporate self-propagation into them to see performance changes.

- **AliNet** (Sun et al., 2020a) extends GCN-Align by introducing distant neighbors in the aggregation function. Its learning objective is to minimize the limit-based loss with truncated negative sampling (Sun et al., 2018). It concatenates the output of multiple layers as representations for alignment learning and search.
- **Dual-AMN** (Mao et al., 2021) is the state-of-the-art aggregation-based model according to our knowledge. It designs several advanced implementations, including the proxy matching attention, normalized hard sample mining and loss normalization. It achieves prominent performance in both effectiveness and efficiency.
- **RoadEA** (Sun et al., 2022) is a recent GCN-based EA method that considers relations in neighborhood aggregation. It combines relation embeddings and their corresponding neighbor embeddings as relation-neighbor representations and uses graph attention networks (Velickovic et al., 2017) to aggregate them.

Table 3. EA results on DBP15K as well as OpenEA D-W and D-Y. The best scores in each group are marked in bold. The results of baseline models are taken from their papers, respectively, and "-" denotes the unreported metric in the corresponding original paper. The results of RoadEA on DBP15K and D-Y are produced using its code. Its results on D-W are taken from its paper.

Models	DBP15K ZH-EN			DBP15K JA-EN			DBP15K FR-EN			OpenEA D-W 15K			OpenEA D-Y 15K		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
GCN-Align	0.413	0.744	-	0.399	0.745	-	0.373	0.745	-	0.364	-	0.461	0.465	-	0.536
GCN-Align + SPA (ours)	**0.441**	**0.751**	-	**0.446**	**0.759**	-	**0.414**	**0.763**	-	**0.378**	0.628	**0.464**	**0.495**	**0.688**	**0.565**
AliNet	0.539	0.826	0.628	0.549	**0.831**	0.645	0.552	0.852	0.657	0.440	**0.672**	0.522	0.559	**0.713**	0.617
AliNet + SPA (ours)	**0.575**	**0.829**	**0.664**	**0.570**	0.821	**0.658**	**0.581**	**0.857**	**0.678**	**0.451**	0.668	**0.529**	**0.563**	0.702	**0.624**
Dual-AMN	0.731	0.923	0.799	0.726	0.927	0.799	0.756	0.948	0.827	0.683	0.893	0.761	0.767	0.908	0.823
Dual-AMN + SPA (ours)	**0.733**	**0.925**	**0.804**	**0.735**	**0.936**	**0.807**	**0.767**	**0.951**	**0.835**	**0.695**	**0.898**	**0.771**	**0.779**	**0.912**	**0.832**
RoadEA	0.570	0.848	0.667	0.569	0.857	0.669	0.578	0.875	0.680	0.495	-	0.584	0.212	0.296	0.244
RoadEA + SPA (ours)	**0.579**	**0.849**	**0.673**	**0.577**	**0.858**	**0.676**	**0.593**	**0.876**	**0.690**	**0.502**	0.744	**0.587**	**0.235**	**0.309**	**0.261**

For each baseline model, we adopt its official code and incorporate the proposed self-propagation into its aggregation function. To be specific, in each of their layers, we add a self-propagation connection between their input and output. We leave other modules, including the alignment learning loss, the negative sampling method, and the alignment search strategy, unchanged. As a result, we get four GCN-based model variants, namely "GCN-Align + SPA", "AliNet + SPA", "Dual-AMN + SPA", and "RoadEA + SPA".

Settings. To ensure a fair comparison, the hyper-parameter values in our experiment follow the default settings of the corresponding baselines. The only exception is that the embedding dimensions of the input and two GCN layers in AliNet+SP are 384, 384 and 384, respectively, which are different from the original settings of 500, 400 and 300 in AliNet. The reason that we keep these layers with the same output dimension is that we can directly compare the representations of an entity in different AliNet layers (see Section 4.2.2). Note that AliNet concatenates the output of all layers as the final entity representations for alignment learning and search. In our model, the final embedding dimension is $384+384+384 = 1152$, slightly smaller than that of AliNet ($500 + 400 + 300 = 1200$). We find that such a small dimension difference has no observed impact on performance. In our models, $\alpha = 0.1$ for all datasets.

Main results. Table 3 presents the EA results of baselines and our model variants on DBP15K. We can see that our model variants can bring stable improvement on DBP15K, especially on Hits@1 and MRR, compared with the corresponding baselines. For example, AliNet+SPA outperforms AliNet by 0.036 on Hits@1. Even when compared to the state-of-the-art model Dual-AMN, our Dual-AMN+SPA still achieves higher performance, especially on JA-EN and FR-EN, establishing a new state-of-the-art. As we have discussed in Section 4.2.2, Dual-AMN has many advanced designs to improve performance. Boosting its performance to a higher level is much more difficult than that for GCN-Align and AliNet. We find that RoadEA fails to achieve promising results on D-Y. We think this is because DBpedia and YAGO have an unbalanced number of relations, which affects the relational attention mechanism in RoadEA. However, our self-propagation still improves it, showing good robustness. To summarize, this comparison demonstrates the effectiveness and generalization of the proposed self-propagation for EA. We conduct additional experiments in the following two subsections to further investigate the reasons for the good performance of self-propagation.

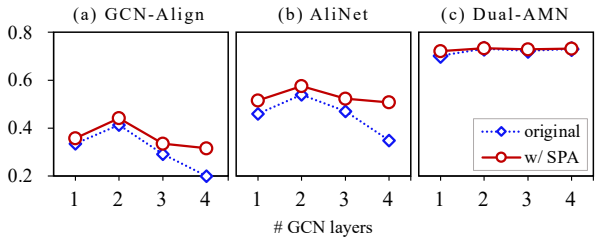

Figure 4. Hits@1 on ZH-EN w.r.t. the number of GCN layers.

Effectiveness of self-propagation against over-smoothing. The over-smoothing issue of GCNs refers to the fact that the output representations tend to be similar if too many layers are used for neighborhood aggregation (Oono & Suzuki, 2020; Chen et al., 2020). It is obvious that such an issue has a negative impact on embedding-based EA. The default settings of GCN layer numbers in GCN-Align, AliNet and Dual-AMN are all 2. To investigate the over-smoothing issue in EA, we show in Figure 4 the Hits@1 results of these baselines (in blue) and our model variants (in red) on ZH-EN when their layer numbers are set as $1, 2, 3, 4$, respectively. Both GCN-Align and AliNet suffer from over-smoothing. Their results decrease as the GCNs go deeper with more than two layers. By adding the self-propagation connection, their performance degradation is reduced. By contrast, Dual-AMN shows good robustness against over-smoothing. Its performance changes little when the layer number increases. Dual-AMN+SPA also benefits from such robustness. Dual-AMN uses the normalized hard sample mining method with a large number of negative examples, enabling dissimilar entities to have distinguishable representations.

Layer output representation comparison. We further compare the output representation distance of the last two layers in AliNet and AliNet+SPA. Figure 5 shows the average Euclidean distance w.r.t. the first 80 training epochs on ZH-EN. We can see that the output representation distance of both AliNet and AliNet+SPA becomes smaller as the validation performance increases. Furthermore, by adding the proposed self-propagation connection, the layer output distance of AliNet+SPA is smaller than that of AliNet. These results provide experimental evidence to support our design of self-propagation to connect two GCN layers and increase the local focus on the entity embedding itself.

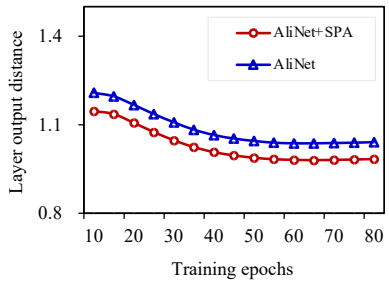

Figure 5. Output representation distance of the last two layers in AliNet and AliNet+SPA on ZH-EN.

5. Related Work

Our work is relevant to multi-sourced KG embedding learning and iteration-based node matching methods for graphs.

5.1. Multi-sourced KG Embeddings

Multi-sourced KG representation learning starts with the research on embedding-based EA. An embedding-based EA model learns and measures entity embeddings to compute entity similarities. It usually has two learning objectives. One is for embedding learning and the other is for alignment learning. Translation-based EA models (Chen et al., 2017; Zhu et al., 2017; Sun et al., 2017; 2019; Pei et al., 2019) adopt TransE (Bordes et al., 2013) or its variants for embedding learning. Aggregation-based EA models adopt GNNs to generate entity embeddings, including the vanilla GCNs (Wang et al., 2018), multi-hop GCNs (Sun et al., 2020a), relational GCNs (Yu et al., 2020), graph attention networks (Zhu et al., 2020; Mao et al., 2021; Sun et al., 2022), self-supervised GCNs (Liu et al., 2022) and temporal GCNs (Xu et al., 2022b). Our proposed self-propagation is a plug-in for GNNs. It adds a direct connection between entity representations and the aggregated neighbor representations. In addition to the above two types of basic models for EA, other studies consider using semi-supervised or active learning techniques to augment EA (Sun et al., 2018; Chen et al., 2018; Li & Song, 2022; Berrendorf et al., 2021; Liu et al., 2021; Zeng et al., 2021b) or introduce some text features (e.g., entity names, attributes and descriptions) (Sun et al., 2017; Trisedya et al., 2019; Wu et al., 2019) or temporal information (Xu et al., 2021; 2022a) to enhance embedding learning. These studies are not relevant to our work. Interested readers can refer to the survey (Zeng et al., 2021a; Zhao et al., 2022; Zhang et al., 2022) for more details. However, our work can also benefit from side features.

5.2. Iteration-based Graph Matching

Computing node similarities in graphs is a long-standing research topic in many areas, such as databases. Our work is relevant to iteration-based similarity computation methods, including similarity flooding (Melnik et al., 2002), SimRank (Jeh & Widom, 2002) and NetAlignMP (Bayati et al., 2013). Their key assumption is that "two nodes are similar if their neighbors are similar". They first compute the similarity of some pairs of nodes. Then, they propagate these similarities to other related node pairs using different heuristic rules iteratively, until they achieve a fixpoint of node pairwise similarities. Our work shows that the embedding-based EA models follow the same key assumption as the conventional iteration-based graph alignment methods. We build a connection between the two types of methods, which would help users acquire deep insights into them.

6. Conclusions and Future Work

In this paper, we present a similarity flooding perspective to understand translation-based and aggregation-based EA models. We prove that these models essentially seek a fixpoint of entity pairwise similarities through embedding learning. Based on this finding, we propose two methods, i.e., similarity flooding via entity compositions and self propagation, for improving EA. Experiments on benchmark datasets demonstrate their effectiveness. Our work fills the gap between recent embedding-based EA and the conventional iteration-based graph matching.

We think there are two promising directions for future work. The first is to develop neural-symbolic EA models that take advantage of both the representation learning ability of neural models and the interpretability of conventional symbolic methods. The second is, given EA, to learn more expressive and transferable multi-sourced KG embeddings to improve downstream knowledge-enhanced tasks. A KG-enhanced task can be extended into a multi-sourced KG-enhanced task. The latter can benefit from the knowledge transfer in multi-sourced KGs and thus get further improvement.

Acknowledgments

This work is funded by the National Natural Science Foundation of China (No. 62272219) and the Alibaba Group through Alibaba Research Fellowship Program.

References

Bayati, M., Gleich, D. F., Saberi, A., and Wang, Y. Message-passing algorithms for sparse network alignment. *ACM Trans. Knowl. Discov. Data*, 7(1):3:1–3:31, 2013.

Berrendorf, M., Faerman, E., and Tresp, V. Active learning for entity alignment. In *ECIR*, pp. 48–62, 2021.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

Bordes, A., Usunier, N., García-Durán, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *NIPS*, pp. 2787–2795, 2013.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, pp. 3438–3445, 2020.

Chen, M., Tian, Y., Yang, M., and Zaniolo, C. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *IJCAI*, pp. 1511–1517, 2017.

Chen, M., Tian, Y., Chang, K., Skiena, S., and Zaniolo, C. Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment. In *IJCAI*, pp. 3998–4004, 2018.

Guo, L., Sun, Z., and Hu, W. Learning to exploit long-term relational dependencies in knowledge graphs. In *ICML*, pp. 2505–2514, 2019.

Guo, L., Zhang, Q., Sun, Z., Chen, M., Hu, W., and Chen, H. Understanding and improving knowledge graph embedding for entity alignment. In *ICML*, pp. 8145–8156, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Jeh, G. and Widom, J. Simrank: a measure of structural-context similarity. In *SIGKDD*, pp. 538–543, 2002.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Li, J. and Song, D. Uncertainty-aware pseudo label refinery for entity alignment. In *WWW*, pp. 829–837, 2022.

Liu, B., Scells, H., Zuccon, G., Hua, W., and Zhao, G. Activeea: Active learning for neural entity alignment. In *EMNLP*, pp. 3364–3374, 2021.

Liu, X., Hong, H., Wang, X., Chen, Z., Kharlamov, E., Dong, Y., and Tang, J. Selfkg: Self-supervised entity alignment in knowledge graphs. In *WWW*, pp. 860–870, 2022.

Mao, X., Wang, W., Wu, Y., and Lan, M. Boosting the speed of entity alignment 10 ×: Dual attention matching network with normalized hard sample mining. In *WWW*, pp. 821–832, 2021.

Melnik, S., Garcia-Molina, H., and Rahm, E. Similarity flooding: A versatile graph matching algorithm (extended technical report). 2001.

Melnik, S., Garcia-Molina, H., and Rahm, E. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pp. 117–128, 2002.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Pei, S., Yu, L., Hoehndorf, R., and Zhang, X. Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference. In *WWW*, pp. 3130–3136, 2019.

Roth, A. and Liebig, T. Transforming pagerank into an infinite-depth graph neural network. *CoRR*, abs/2207.00684, 2022.

Shvaiko, P. and Euzenat, J. Ontology matching: State of the art and future challenges. *IEEE Trans. Knowl. Data Eng.*, 25(1):158–176, 2013.

Sun, Z., Hu, W., and Li, C. Cross-lingual entity alignment via joint attribute-preserving embedding. In *ISWC*, pp. 628–644, 2017.

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

Sun, Z., Huang, J., Hu, W., Chen, M., Guo, L., and Qu, Y. Transedge: Translating relation-contextualized embeddings for knowledge graphs. In *ISWC*, pp. 612–629, 2019.

Sun, Z., Wang, C., Hu, W., Chen, M., Dai, J., Zhang, W., and Qu, Y. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI*, pp. 222–229, 2020a.

Sun, Z., Zhang, Q., Hu, W., Wang, C., Chen, M., Akrami, F., and Li, C. A benchmarking study of embedding-based entity alignment for knowledge graphs. *PVLDB*, 13(11): 2326–2340, 2020b.

Sun, Z., Hu, W., Wang, C., Wang, Y., and Qu, Y. Revisiting embedding-based entity alignment: A robust and adaptive method. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2022. doi: 10.1109/TKDE.2022.3200981.

Trisedya, B. D., Qi, J., and Zhang, R. Entity alignment between knowledge graphs using attribute embeddings. In *AAAI*, pp. 297–304, 2019.

Trivedi, R., Sisman, B., Dong, X. L., Faloutsos, C., Ma, J., and Zha, H. Linknbed: Multi-graph representation learning with entity linkage. In *ACL*, pp. 252–262, 2018.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *CoRR*, abs/1710.10903, 2017.

Wang, Z., Lv, Q., Lan, X., and Zhang, Y. Cross-lingual knowledge graph alignment via graph convolutional networks. In *EMNLP*, pp. 349–357, 2018.

Wu, Y., Liu, X., Feng, Y., Wang, Z., Yan, R., and Zhao, D. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, pp. 5278–5284, 2019.

Xu, C., Su, F., and Lehmann, J. Time-aware graph neural network for entity alignment between temporal knowledge graphs. In *EMNLP*, pp. 8999–9010, 2021.

Xu, C., Su, F., Xiong, B., and Lehmann, J. Time-aware entity alignment using temporal relational attention. In *WWW*, pp. 788–797, 2022a.

Xu, C., Su, F., Xiong, B., and Lehmann, J. Time-aware entity alignment using temporal relational attention. In *WWW*, pp. 788–797. ACM, 2022b.

Yu, D., Yang, Y., Zhang, R., and Wu, Y. Generalized multi-relational graph convolution network. *CoRR*, abs/2006.07331, 2020.

Zeng, K., Li, C., Hou, L., Li, J., and Feng, L. A comprehensive survey of entity alignment for knowledge graphs. *AI Open*, 2:1–13, 2021a.

Zeng, W., Zhao, X., Tang, J., and Fan, C. Reinforced active entity alignment. In *CIKM*, pp. 2477–2486, 2021b.

Zhang, R., Trisedya, B. D., Li, M., Jiang, Y., and Qi, J. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *VLDBJ*, 31(5):1143–1168, 2022.

Zhao, X., Zeng, W., Tang, J., Wang, W., and Suchanek, F. M. An experimental study of state-of-the-art entity alignment approaches. *TKDE*, 34(6):2610–2625, 2022.

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

Zhu, Q., Wei, H., Sisman, B., Zheng, D., Faloutsos, C., Dong, X. L., and Han, J. Collective multi-type entity alignment between knowledge graphs. In *WWW*, pp. 2241–2252, 2020.