JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 40, 317-339 (2024)
DOI: 10.6688/JISE.202403_40(2).0008

Embedding-based Two-Stage Entity Alignment for Cross-Lingual Knowledge Graphs*

YUXIANG SUN[1] AND YONGJU LEE[2,+]
[1]*Software Technology Research Center*
[2]*School of Computer Science and Engineering*
Kyungpook National University
Daegu, 41566 Republic of Korea
E-mail: syx921120@gmail.com; yongju@knu.ac.kr

In the knowledge graph alignment process, most researchers focus only on heterogeneous entities, thereby, ignoring the impact of homogeneous entities on matching accuracy and efficiency. In this study, we propose a two-stage strategy that corresponds to homogeneous and heterogeneous entities. In the first stage, an embedding-based semantic clustering algorithm is applied to divide the semantics into multiple clusters, which are paired according to the centroid distance. Additionally, homogeneous entities were matched by combining the Linked Lists and *k*-dimensional trees. In the second stage, we propose an embedding-based graph convolutional neural network (E-GCN) model that assigns different weights to relations based on the aligned homogeneous entities in the first stage. Compared with other GCN-based models, the entity alignment (EA) accuracy of the E-GCN model was the best, and the training time was reduced by 43.2%. Experimental results reveal that the proposed two-stage method significantly improves EA performance compared with state-of-the-art EA models. Moreover, the Canberra semantic distance is most suitable for representing the similarity between entities and the exponential linear unit (ELU) activation function accelerated the convergence of the E-GCN model.

***Keywords*:** knowledge graph embedding, entity alignment, graph convolutional network, heterogeneous entities, homogeneous entities, semantic distance calculation

1. INTRODUCTION

Heterogeneous graph problems among different knowledge graphs (KGs) have not been considered despite the success of graph convolutional networks (GCNs) in KGs. Although most of the GCN methods are restricted to graphs with homogeneous edges, real-world KGs typically consist of various types of nodes and edges. In KGs, vertices and directed edges are called entities and triplets. Edges are represented in relational triplets (*h*, *r*, *t*) or attribute triplets <head entity; attribute; and attribute value>. A technique that can integrate entities among heterogeneous KGs is required because heterogeneous graphs are usually incomplete and complementary to each other. The entity alignment (EA) technique automatically extracts equivalent entities when the heterogeneity of names and semantics in different KGs is encountered.

Received September 12, 2022; revised February 18 & April 17, 2023; accepted April 27, 2023.
Communicated by De-Nian Yang.
+ Corresponding author.
* This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2016R1D1A1B02008553). This study was supported by the BK21 FOUR project (AI-driven Convergence Software Education Research Program) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (4199990214394).

Recently, many EA studies have been conducted to solve KG heterogeneity problems; however, most of them are translation-based models, such as MTransE [1], IPTransE [2], and BootEA [3]. The translation-based methods require less human involvement in feature extraction and are easily scalable to large-scale KGs compared to the existing convolutional neural networks (CNNs). However, owing to the limited assumption that $h + r \approx t$, they are inefficient in capturing more complex relationship information, especially in multiple relationship graphs.

A novel research direction for EA has progressed by utilizing GCN models. However, limitations in terms of the proper modeling of relational information exist in this approach as well. Valuable relational information in KGs can be ignored because most of the GCN methods operate on undirected and unlabeled graphs. Relational GCNs (R-GCNs) [4] can be used to model multi-relationship graphs; however, they require an excessive set of parameters and use a separate weight matrix for each relationship. Dual-Primal Graph CNN (DPGCNN) [5] determines vertex and edge features by alternately performing convolution operations between the vertex-centered primal graph and its dual graph. Furthermore, it repeatedly applies the graph attention mechanism to explore complex edge structures. Inspired by the DPGCNN, a relation-aware dual GCN (RDGCN) [6] was proposed, which can better represent and characterize the relationships between different KGs. Although DPGCNN and RDGCN provide a novel research direction, their matching accuracy is still low because they only use deep learning functions.

Homogeneous entities have the same semantics and belong to the same synonym set. However, heterogeneous entities with the same semantics cannot be matched because they do not belong to the same synonym set. Fig. 1 depicts an example of the heterogeneity and homogeneity of KG where the grey rectangular boxes indicate the heterogeneity. Additionally, there are some entities with the same expression among the different KGs and are called homogeneous entities (*e.g.*, Mark's circular image). The number of homogeneous entities is large, and the representation of the triples is identical. Therefore, it is not advisable to adopt complex GCN-based approaches for homogeneous entities.

Here, we have adopted a two-stage strategy to solve the heterogeneity and homogeneity problems separately. In the first stage, homogeneous entities between KGs were extracted through a combination of an embedding-based semantic clustering method and a *k*-dimensional tree. Owing to the utilization of the index structure, the matching efficiency was significantly improved. In the second stage, a homogeneous weight layer was added before the dual attention layer of the RDGCN model to weigh the relation between two homogeneous entities. In other words, if two entities have a homogeneous relationship, the edge connected to these entities should be assigned more weight. This method can enrich the relationship between entities and accelerate the convergence of our GCN model. The contributions of our study are summarized as follows,

- To the best of our knowledge, this is the first attempt to study the impact of homogeneous entities on EA. The experimental results reveal that assigning different weights to the edges of homogeneous entities can effectively improve the accuracy and convergence speed of the EA.
- Our method can improve the EA performance without altering the existing GCN-based model structure. In addition, the silhouette coefficient (SC) and time matching ratio (TMR) are utilized to quickly find the optimal model parameters using semantic clus-

tering and alignment for homogeneous entities.

- We have proposed an embedding-based GCN (E-GCN) model. Moreover, to improve the semantic expression ability of nodes in the KGs, we have adopted a graph attention mechanism. The experimental results reveal that the Canberra semantic distance is appropriate for the semantic similarity representation, and the exponential linear unit (ELU) activation function is the best option for considering accuracy and efficiency.

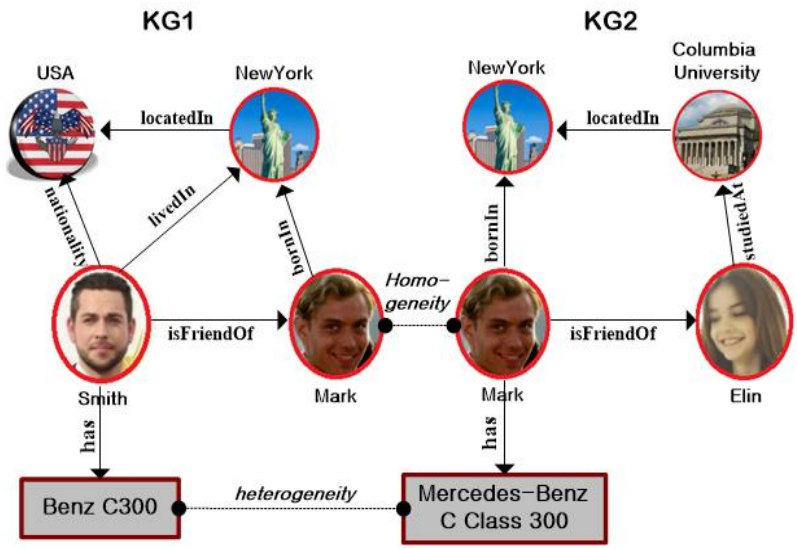


Fig. 1. An example of the heterogeneity and homogeneity of the KG. Circles represent entity identifiers; arrows represent various relationships, and rectangular boxes store the attribute values. Dashed lines connect the seed entity pairs.

The remaining sections of this paper are organized as follows. In Section 2, the related studies are described. In Section 3, the two-stage EA architecture and flowchart for homogeneous and heterogeneous entities are presented. Here, an E-GCN model is proposed, and a two-stage EA algorithm is designed. The *k* value determination involved in the experiment is also defined. In Section 4, the performance evaluation of the existing mainstream methods is described. Finally, in Section 7, conclusions are inferred, and future research directions are suggested.

2. RELATED WORKS

2.1 KGE Models

KGE models embed a graph structure into a low-dimensional vector space and the existing KGE models are divided into three main groups.

(1) Translational models (*e.g.*, TransE [7], TransH [8], TransR [9], and TransD [10]) were inspired by Word2Vec [11], which captures semantic information based on the prin-

ciple of translational invariance. Moreover, TransE regards entities in the KGs as translation vectors. Assume a triple (h, r, t), where h, r, and t represent head entities, relations, and tail entities, respectively. To make $h + r = t$, a scoring function is utilized to continuously optimize h, r, and t. However, these models have a low expressive ability without capturing semantic information.

(2) Semantic matching models (*e.g.*, DistMult [12], ComplEx [13], HolE [14], SimplE [15], RotatE [16], and TuckER [17]): Compared to translational models, semantic matching models exhibit a significant performance improvement by utilizing matrix multiplication. However, these models ignore the intrinsic relationships between triples in the KGs because they are not sufficient for capturing deeper semantics and complex relations.

(3) Deep learning models (*e.g.*, ProjE [18], ConvE [19], R-GCNs, ConvKB [20], and DSKG [21]) are primarily based on the type of entity or relation, path information, and structure information. The use of CNN and attention mechanisms results in a better embedding performance. Table 1 lists the weaknesses of the KGE models.

Table 1. KGE models.

Category	Weakness	Models
Translational Models	Low expressive power without capturing semantic information	TransE, TransH, TransR, TransD
Semantic Matching Models	Not Sufficient for capturing deeper semantics	DistMult, ComplEx, HolE, SimplE, RotatE, TuckER
Deep Learning Models	Low efficiency for large scale KGs	ProjE, ConvE, R-GCNs, ConvKB, DSKG

2.2 EA Models

Embedding-based models, such as JE [22], MTransE, JAPE [23], IPTransE, and BootEA, were used for entity representations based on prior alignments. These methods require less human involvement in feature construction and can be scaled to large KGs. However, they were constrained by the assumption $h + r \approx t$. This strong assumption makes it inefficient for the model to capture more complex relational information in multi-relational graphs. Zequn *et al.* [24] indicated that embedding-based methods cannot be directly applied to EA for two reasons. First, not all the entities follow the same schema. Second, the attribute alignment must be a 1-to-1 mapping. Considering the above issues, Zequn *et al.* [25] adopted the GCN and proposed a new direction to represent multiple KG entities. However, this approach is also unable to model relation information accurately. Regarding the R-GCN model, it performs message passing through a convolution opera-tion on multi-relational graphs. HGCNs are two-layered GCN with entity name initialization and highway gates. Subsequently, GCN-based models, such as RDGCN and entity alignment with multi-order graph convolutional network (EMGCN) [26], were proposed, and RDGCN achieved better entity representation by the attention interaction mechanism and dual graph. The EMGCN is an unsupervised KG entity alignment method that uses a late-fusion mechanism, and rich relational triples can be integrated. Table 2 lists the categories, advantages, and weaknesses of EA models.

Table 2. EA models.

Category	Advantages and disadvantages		Models
Embedding-based Models	Pros	– Requires less human involvement in feature construction – Can be scaled to large KGs	JE, MTransE, JAPE, IPTransE, BootEA
	Cons	– Not all entities follow the same schema – Not all attributes alignment meets 1-to-1 mapping	
GCN-based Models	Pros	– Better representations for complex edge structures	RDGCN, EMGCN, GCN-Align, HGCN-JE
	Cons	– Unable to model relation information accurately	

2.3 Semantic Distance Calculation Methods

The goal of embedded models is to perform EA based on vector distance after obtaining the semantic vectors. According to the vector category, they can be divided into numerical and Boolean vectors; among them, semantic vectors belong to numerical vectors. The most common distance calculation method is the Euclidean distance [27] which is simple and often used to calculate the semantic distance between low-dimensional vectors. The Manhattan Chebyshev distance [28] represents the sum of the absolute wheelbases of the entity vectors and the maximum difference between the entity vectors, respectively. The cosine distance [29] is not considered as a distance but a similarity. Other distances directly measure the distance between two entity vectors in high-dimensional space. Considering the distance as zero, two vectors are regarded as "same" and the result of the cosine is in [0, 1]. The Canberra distance [30] is a weighted version of the Manhattan distance used to measure the middle of the two vector spaces. Moreover, it has been used to compare ranking lists and intrusion detection in computer security. Table 3 lists different semantic distance methods.

Table 3. Semantic distance methods.

Methods	Formula	Methods	Formula
Canberra Distance	$\sum_{i=1}^{n} \frac{\lvert e_{1i}-e_{2i}\rvert}{\lvert e_{1i}\rvert+\lvert e_{2i}\rvert}$	Hamming Distance	$\frac{C_{01}-C_{10}}{n}$
Manhattan Distance	$\sum_{i=1}^{n} \lvert e_{1i}-e_{2i}\rvert$	Russell-Rao dissimilarity	$\frac{n-C_{TT}}{n}$
Cosine Distance	$\frac{\sum_{i=1}^{n} e_{1i}\times e_{2i}}{\sqrt{\sum_{i=1}^{n} e_{1i}^2}\times\sqrt{\sum_{i=1}^{n} e_{2i}^2}}$	Sokal-Michener dissimilarity	$\frac{2(C_{TF}+C_{FT})}{C_{FF}+C_{TT}}$
Euclidean Distance	$\sqrt{\sum_{i=1}^{n}(e_{1i}-e_{2i})^2}$	Yule dissimilarity	$\frac{2\times C_{TF}\times C_{FT}}{C_{FF}\times C_{TT}+C_{TF}\times C_{FT}}$
Chebyshev Distance	$\max_i(e_{1i}-e_{2i})$		

2.4 Activation Functions

The activation function adds non-linear factors to the model to make it more expressive. Common activation functions include sigmoid and rectified linear unit (ReLU). The

sigmoid function limits the value between (0, 1), which can be used as a probability to explain the results obtained by the algorithm and is suitable for binary classification problems. ReLU has no gradient saturation problem when the input is positive, and the training speed is much faster when compared to the sigmoid function. However, the sigmoid and ReLU functions have a common problem; if the input is negative, the gradient is completely zero. The leaky ReLU function was introduced to solve this problem, and its range was from negative infinity to positive infinity. ELU causes the normal gradient closer to the unit natural gradient by reducing the effect of the bias offset, thereby accelerating the learning of the mean towards zero. Swish used the same gating value to simplify the gating mechanism, called self-gating. This can prevent slow gradients from gradually approaching zero and causing saturation. Table 4 lists formulas of activation functions.

Table 4. Activation functions.

Name	Formula	Name	Formula
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	Leaky ReLU	$f(x)=\lambda\begin{cases}x, & x>0\\ \alpha x, & x\le 0\end{cases}$
Swish	$f(x) = x * sigmoid(\beta x)$	SELU	$f(x)=\lambda\begin{cases}x, & x>0\\ \alpha e^{x}-\alpha, & x\le 0\end{cases}$
ReLU	$f(x)=\lambda\begin{cases}1, & x>0\\ 0, & x\le 0\end{cases}$	ELU	$f(x)=\begin{cases}x, & x>0\\ \alpha(e^{x}-1), & x\le 0\end{cases}$

3. TWO-STAGE ENTITY ALIGNMENT METHOD

3.1 Two-stage EA Architecture

Fig. 2 depicts the proposed two-stage EA architecture. The entire architecture consists of preprocessing (entity extraction, embedding, and semantic clustering), EA (first and second stages), and optimization (*k* values adjustment). The main purpose of preprocessing is to replace the entities with vectors and match clusters distributed in different KGs, acc-

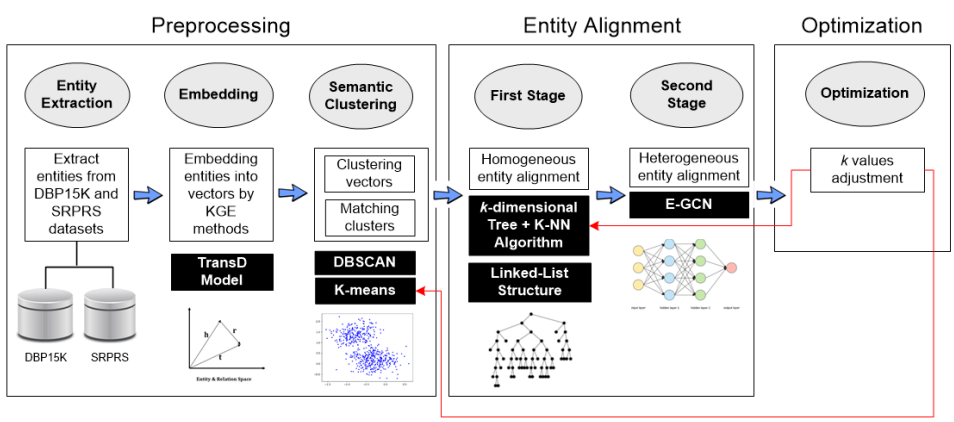


Fig. 2. Proposed two-stage EA architecture.

cording to the centroid distance. The EA is divided into the first and second stages, which process the alignment of homogeneous and heterogeneous entities, respectively. Mainly, the optimization is for the selection of *k* values that appear in the K-means and K-nearest neighbor (K-NN) algorithms. The definitions of KGs and EA are as follows:

Definition 1: KGs are represented by G = (E, R, A, AV, T), where E, R, A, AV, and T represent entities, relations, attributes, attribute values, and triples, respectively. Given two KGs $G_1 = (E_1, R_1, A_1, AV_1, T_1)$ and $G_2 = (E_2, R_2, A_2, AV_2, T_2)$, the EA aims to identify entity pairs (e_1, e_2), $e_1 \in E_1$, $e_2 \in E_2$, where e_1 and e_2 represent the same real-world entity.

Table 5. Statistics of benchmark datasets.

Dataset	Language	Entities #	Rel #	Rel.triples #	Attr.triples #
DBP15K	Chinese	19,388	2,830	153,929	379,684
Zh-En	English	19,572	2,317	237,674	567,755
DBP15K	Japanese	19,814	2,043	164,373	354,619
Ja-En	English	19,780	2,096	233,319	497,230
DBP15K	French	19,661	1,379	192,191	528,665
Fr-En	English	19,993	2,209	278,590	576,543
SRPRS	French	15,000	177	33,532	53,045
Fr-En	English	15,000	221	36,508	60,800
SRPRS	German	15,000	222	38,363	55,580
De-En	English	15,000	120	37,377	73,753
SRPRS	DBpedia	15,000	253	38,421	64,021
DBP-Wd	Wikidata	15,000	144	40,159	133,371
SRPRS	DBpedia	15,000	323	33,748	58,853
DBP-Yg	Yago 3	15,000	30	36,569	18,241

[*] Entities #, Rel #, Rel.triples #, and Attr.triles indicate the number of entities, relations, relation triples, and attribute triples in each KG, respectively.

(1) Entity extraction: In our experiments, we have used the DBP15K datasets [31]. DBP15K contains four language-specific KGs extracted from English (En), Chinese (Zh), French (Fr), and Japanese (Ja) DBpedia, each of which contains approximately 65,000–106,000 entities. Three DBP15K sets were constructed to align entities between the other three languages (Zh, Fr, and Ja) and En. For example, we merge two different Zh and En KGs to create a Zh-En dataset. Table 5 presents the statistics for the DBP15K and SRPRS datasets. The main task of the entity extraction step is to extract entities from the DBP15K datasets using Apache Jena [32]. Lingbing *et al.* [33] indicated that the KGs of traditional EA datasets (*e.g.*, DBP15K or DWY100K) are highly dense, and the degree distributions deviate from the real-life KGs. Therefore, they established a new EA benchmark (*i.e.*, SRPRS) that follows a real-life distribution using reference links in DBpedia. The SRPRS evaluation benchmark consists of cross-lingual (Fr-En and De-En) and mono-lingual KG pairs (DBP-Wd and DBP-Yg), where De, Wd, and Yg represent German, Wikidata, and Yago 3, respec-tively.

(2) Embedding: The main purpose of this step is to embed the entities into vectors using the KGE method. The KGE models can be classified as translational, semantic matching, and deep-learning models. Representative embedded models were evaluated to determine the best performance model. For example, TransD and TransR were select-

ed for translation models, ComplEx and DistMult for semantic matching models, and ConvE for deep learning models. All embedding experiments were conducted using the OpenKE platform [34]. The experimental results reveal that the performance of translational models is better than that of others; in particular, TransD performs the best. We embed the 7 datasets in Table 5 using the TransD model.

(3) Semantic clustering: There are two tasks in this step: clustering vectors and matching clusters distributed in different KGs according to the centroid distance. Density-based spatial clustering of applications with noise (DBSCAN) and K-means clustering can be used to cluster vectors. DBSCAN has the advantage of clustering arbitrary-shaped clusters. However, this requires considerable time and memory in the actual experiments which correspond to MinPts (the minimum number of samples) and Eps (the maximum distance between two samples). The number of iterations was negatively related to the above parameters. Therefore, the K-means clustering method was adopted instead of DBSCAN. To achieve the optimal clustering effect, the number of clusters in K-means clustering can be determined according to the visualization of triples, unlike DBSCAN, where MinPts and Eps must be fine-tuned repeatedly. We calculated the centroids of clusters distributed in two KGs and pairs of clusters based on the centroid distance.

(4) First-stage: In the semantic clustering process, the clusters were paired in the two KGs. In the first stage, homogeneous EA on the paired clusters was performed. Owing to the large size difference between clusters, we need to traverse repeatedly to find homogenous entities. Therefore, a *k*-dimensional tree for large clusters and Linked-Lists for small clusters are initialized to improve the alignment efficiency. A *k*-dimensional tree combined with the K-NN algorithm can effectively reduce the number of iterations and is suitable for large clusters. Moreover, the Linked-List structure is simple, flexible, and suitable for small clusters. An entity is considered homogenous if the distance between the two entities is the closest and the process is called the homogeneous entity alignment. In actual experiments, there is a phenomenon that one entity corresponds to multiple entities. Therefore, an appropriate *k* value for the K-NN algorithm must be selected to solve this problem. A TMR equation was proposed to select the appropriate *k* value, as presented in Eq. (1). The numerator represents successfully matched entities (SME), and the denominator represents time cost (TC). The purpose of this equation is to determine the optimal *k* value for efficiency and effectiveness.

$$\arg\max f(K) = \frac{SME_K}{TC_K} \tag{1}$$

(5) Second-stage: First, the homogeneous entities from the dataset are marked to improve the matching accuracy for GCN-based models. The specific approach is to reuse Linked-Lists and *k*-dimensional tree to filter out homogeneous entities. A *k*-dimensional tree has two functions: a homogeneous EA and a filter for heterogeneous entities. Because of the low proportion of homogeneous entities in the total EA, they are stored in Linked-Lists, and the corresponding KG is stored in a *k*-dimensional tree. In the traversal process, if entities are similar, they are removed from the KG. Finally, the filtered KG is trained using the E-GCN model.

(6) Optimization: In this step, two variables *k* for K-means and K-NN were adjusted to achieve optimal EA performance. Fig. 3 depicts a visual representation of the implementation process of the proposed EA architecture.

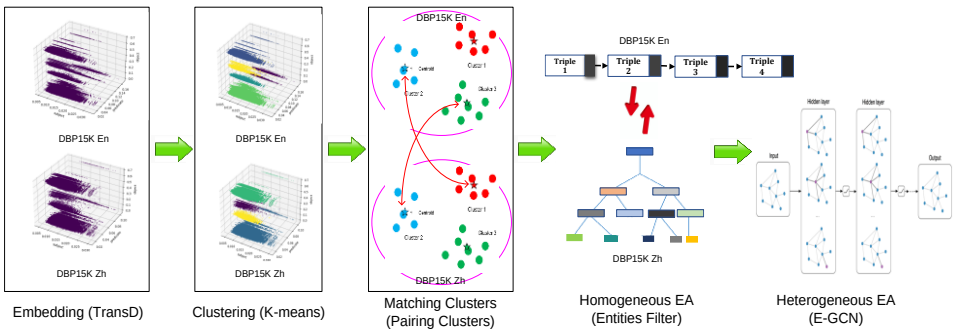


Fig. 3. Implementing process of the proposed EA architecture.

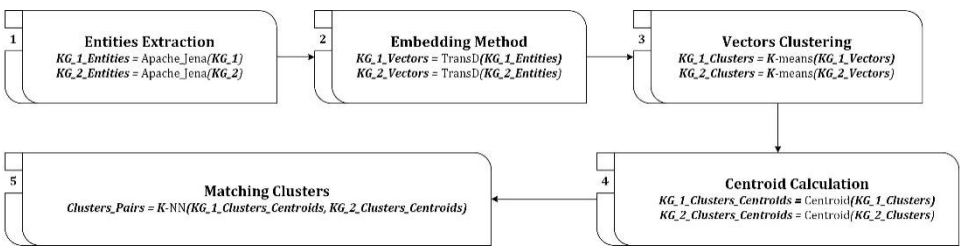


Fig. 4. Preprocessing process for KG_1 and KG_2.

3.2 Two-Stage EA Algorithms

The entire algorithm is divided into preprocessing, first-stage, and second-stage. Furthermore, preprocessing is subdivided into entity extraction, embedding method, vector clustering, and matching clusters. Fig. 4 presents the preprocessing process for KG_1 and KG_2. First, Apache Jena was adopted to extract the entities from the two KGs (Step 1) and then the embedding method was performed based on the TransD model for extracted entities (Step 2). Subsequently, the vectors of the entities are obtained. The next step is to perform clustering on the vectors and considering the efficiency, the K-means clustering algorithm was chosen (Step 3). Later, the centroid point of each cluster was determined (Step 4). Finally, the clusters were paired using the K-NN algorithm according to the centroid distance (Step 5). Eq. (2) presents the calculation method for the centroid points.

$$Centroid = \left(\frac{\sum_{i=0}^{n} S_i}{n}, \frac{\sum_{i=0}^{n} P_i}{n}, \frac{\sum_{i=0}^{n} O_i}{n} \right) \tag{2}$$

where *S*, *P*, *O*, and *n* represent the subject, predicate, object, and total number of entities in each cluster, respectively. The biggest impact on the experimental results is the choice

of *k* value for K-means and K-NN. The determination of the optimization *k* value is introduced in Section 3.4.

3.2.1 First-stage EA algorithm

Algorithm 1 presents the first-stage EA algorithm for homogeneous entities. Assuming X as a successfully paired cluster in KG_1_Clusters_Pairs, there must be a corresponding cluster Y in the KG_2_Clusters_Pairs. Our main task was to traverse the paired clusters (lines 1 and 2). The larger cluster is initialized in a *k*-dimensional tree and the smaller cluster is Linked-Lists (lines 3 to 8) to improve the matching speed. Finally, the semantic similarity between E1 and E2 was calculated using the Euclidean distance (ED), as presented in Eq. (3). If the ED is the shortest, then it has a homogeneous relationship (lines 9 to 13).

$$ED = \sqrt{(S_1 - S_2)^2 + (P_1 - P_2)^2 + (O_1 - O_2)^2} \tag{3}$$

Algorithm 1: First-stage EA Algorithm

```
1  For each cluster X in KG_1_Clusters_Pairs:
2      For each cluster Y in KG_2_Clusters_Pairs:
3          If size of X > size of Y
4                  KT = k-dimensional Tree(X)
5                  LL = Linked-List(Y)
6          Else
7                  KT = k-dimensional Tree(Y)
8                  LL = Linked-List(X)
9          For each triple E1 in LL:
10             If ED of E1 and E2 in KT is the nearest
11                 Then Add (E1, E2) to
                   First-stage(EA_Results)
12             End If
13         End For
14     End For
15 End For
```

Algorithm 2: Second-stage EA Algorithm

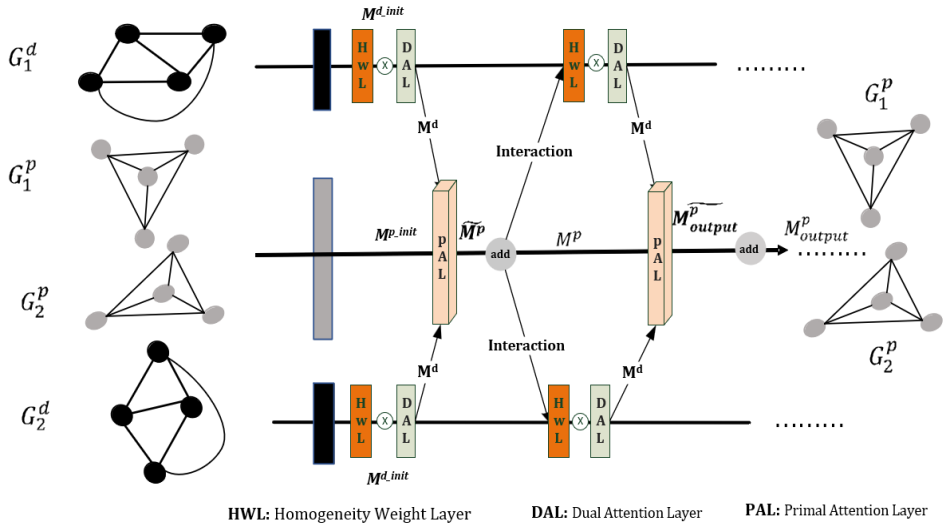
```
1   K_KG1 = k-dimensional Tree (KG1)
2     K_KG2 = k-dimensional Tre (KG2)
3     For E1, E2 in First-stage(EA_Results)
4           If  E1 in K_KG1
5               Mark E1 in K_ KG1
6           End If
7           If  E2 in K_ KG2
8               Mark E2 in K_ KG2
9           End If
10    End For
11  Second-stage(EA_Results) =
                        E-GCN(K_KG1, K_KG2)
12    Final(EA_Results) = First-stage(EA_Results)
        +
      Second-stage(EA_Results)
```

3.2.2 Second-stage EA algorithm

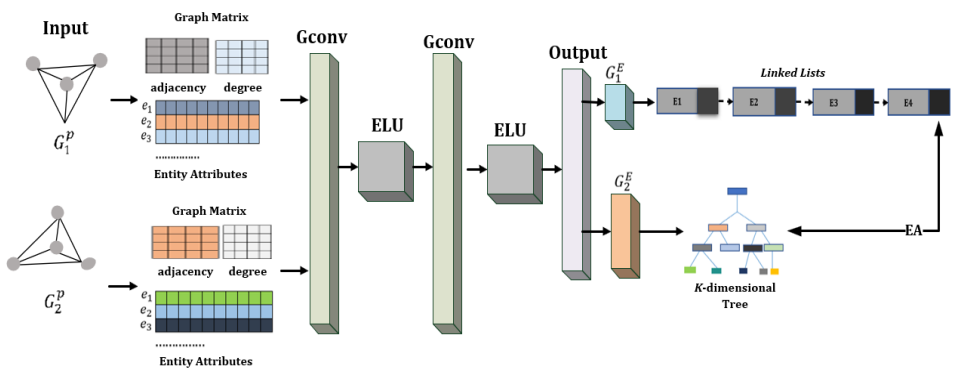
Algorithm 2 presents the second-stage EA algorithm for heterogeneous entities. First, KG_1 and KG_2 were initialized to a *k*-dimensional tree, which not only improves the efficiency of traversal but also reuses the previous index structure to reduce the complexity of the EA system. The main purpose of lines 3 to 10 is to mark homogeneous entities in KG_1 and KG_2 to enrich the relationship expression in the homogeneous weight layer. Thereafter, K_KG_1 and K_KG_2 were trained using the E-GCN model for the second-stage EA. Finally, the EA result is composed of the first and second-stage results (line 12).

3.3 E-GCN Model

The proposed E-GCN model is divided into three stages: relationship-aware, GCN, and similarity computation, as depicted in Fig. 5.



(a) Relationship-aware stage.



(b) GCN and similarity computation stage.

Fig. 5. E-GCN model for heterogeneous entities alignment.

3.3.1 Interaction between dual graphs and homogeneity weight layer

A primal graph is depicted in a set of vertices connected by edges. Let us assume a primal graph $G^p = (V^p, E^p)$, where the vertex set $V^p = Entity_1 \cup Entity_2$ and the edge set $E^p = Triple_1 \cup Triple_2$. Given a primal graph G^p, the dual-relation graph $G^d = (V^d, E^d)$ can be constructed. If two relations r_m and r_n share similar heads or tails, the Jaccard coefficient of each edge W^d_{mn} in G^d will be assigned a greater weight θ. If r_m and r_n share homogeneous heads or tails, W^d_{mn} will be assigned a weight γ as computed in Eqs. (4)-(6). The value of γ is greater than θ but less than 1.

$$W^d_{mn} = Head(r_m, r_n) + Tail(r_m, r_n) \tag{4}$$

$$Head(r_m, r_n) = \frac{Head_m \cap Head_n}{Head_m \cup Head_n} \tag{5}$$

$$Tail(r_m, r_n) = \frac{Tail_m \cap Tail_n}{Tail_m \cup Tail_n} \tag{6}$$

where $Head_m$ and $Tail_m$ are the sets of head and tail entities for relation r_m. The main task of homogeneity weight layer (HWL) is to assign different weight for each edge in G^d by judging whether entities are homogeneous or similar in G^P. If there is a homogeneous relationship between $Head_1$ or $Tail_1$ in G_1^p and $Head_2$ or $Tail_2$ in G_2^p then the corresponding edge will be given a great weight γ in G^d. Our goal of introducing HWL in dual graphs is to better incorporate relation information into G^P.

3.3.2 Interactions between dual and primal graphs

In the dual attention layer, the graph attention mechanism (GAM) was used to assign different weights to the vertices. The GAM consists of an attention coefficient in Eqs. (7)-(8) and aggregate in Eq. (9). h_m is the relation representation for relation r_m in the primary attention layer. For any vertex v_m, it and its neighbors ($n \in N_m^d$) are concatenated by $[h_m \| h_n]$ and assign weight W_{mn}^d, and β projects the concatenated multidimensional vertex matrix to a scalar. Finally, the attention coefficient a_{mn}^d was calculated using a softmax function with the activation function ELU in Eq. (8).

$$D_{mn}^d = \beta^d (W_{mn}^d [h_m \| h_n]), n \in N_m^d \tag{7}$$

$$a_{mn}^d = \frac{\exp(ELU(D_{mn}^d))}{\sum_{k \in N_m^d} \exp(ELU(D_{mk}^d))} \tag{8}$$

According to the attention coefficient a_{mn}^d, the vertices were aggregated using Eq. (9). M_m^d represents a new dual representation of vertex v_m. N_m^d is the neighbor vertex in the dual graph and M_n^d denotes the dual representation of vertex v_n.

$$M_m^d = ELU\left(\sum_{n \in N_m^d} a_{mn}^d M_n^d\right) \tag{9}$$

In the primal attention layer, the primal attention coefficient a_{ij}^p was computed using the dual vertex representation M_{ij}^d. M_{ij}^d denotes the dual representation of r_{ij} (the relation between entities e_i and e_j). FL is a fully connected layer that maps the input into a scalar. For entity e_i in the primal graph G^p, its representation $\tilde{M}_i^p$ can be computed using Eq. (11). The relationship-aware entity representation $\tilde{M}_i^p$ can be better identified by the interaction between the dual and primary graphs. Fig. 5 (a) presents the flow of these two interactions.

$$a_{ij}^p = \frac{\exp(ELU(FL^p M_{ij}^d))}{\sum_{k \in N_i^p} \exp(ELU(FL^p M_{ik}^d))} \tag{10}$$

$$\tilde{M}_i^p = ELU\left(\sum_{j \in N_i^p} a_{ij}^p M_j^p\right) \tag{11}$$

The initial representation matrix for the primal vertices M^{p_init} is initialized using the

TransD model. The primal graph vectors were updated by mixing the initial representations with the output of the primal attention layer M^p, where M^P_{output} denotes the final output representation of G_1^p and G_2^p by updating with the second interaction result $\tilde{M}^p_{output}$.

$$M^p = \tilde{M}_i^p + M^{p\_init} \tag{12}$$

$$M^P_{output} = M^p + \tilde{M}^p_{output} \tag{13}$$

3.3.3 Model training and similarity computation

The input layer is composed of G_1 and G_2, and each graph is divided into graph matrices (adjacency and degree matrices) and identity matrix (entity attributes). The output layer is a new vector for each entity, where G_1^E and G_2^E represent the entity vector sets in G_1 and G_2, respectively. The $(l + 1)$th layer of the E-GCN model is based on the output of the previous layer. The convolutional computation is as follows.

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{14}$$

where A is an adjacency matrix, $\hat{A} = A + I$, where I is an identity matrix, $\hat{D}$ is the degree matrix of $\hat{A}$, and $H^{(l)}$ is a hidden state that is input to the lth layer. $W^{(l)}$ is a weight matrix for the lth layer and σ is a non-linear activation function ELU.

The gated skip connection creates a new hidden state by calculating the correlation coefficient between the hidden state $H^{(l-1)}$ that does not pass through the hidden layer, and the batch-normalized hidden state $\tilde{H}^{(l)}$ which passes the hidden layer.

$$T = gate(H^{(l-1)}, \tilde{H}^{(l)}) = sigmoid(U_1 H^{(l-1)} + U_2 \tilde{H}^{(l)} + b) \tag{15}$$

where U_1 and U_2 are the linear hidden layers and b is the bias. The hidden state of the l layer is updated based on the gate function.

$$H^{(l)} = sigmoid(T * H^{(l)} + (1 - T) * \tilde{H}^{(l-1)}) \tag{16}$$

To improve the efficiency of EA, the index structures in the first stage are reused, where entities with a small amount are stored in Linked-Lists and entities with a large amount are stored in a k-dimensional tree. Canberra Distance (CD) is used to calculate the semantic similarity between two n-dimensional vectors $e_1(e_{11}, e_{12}, \ldots, e_{1k})$ and $e_2(e_{21}, e_{22}, \ldots, e_{2k})$, as presented in Eq. (17).

$$CD(e_1, e_2) = \sum_{k=1}^{n} \frac{|e_{1k} - e_{2k}|}{|e_{1k}| + |e_{2k}|} \tag{17}$$

To minimize the distance between the entities of the equivalent entity pair and optimize the loss function of the maximum margin, we use Eq. (18), where $\varnothing$ is a marginal hyperparameter greater than zero. L is our alignment seed and L' is the set of negative instances. In the EA process, the same index structure as in the first stage is used to accelerate the training model.

$$L = \sum_{(i,j)\in L} \sum_{(i',j')\in L'} \max\{0, CD(i,j) - CD(i',j') + \varnothing\} \tag{18}$$

3.4 *k* value Determination for K-means and K-NN

The *k* value selection had a significant impact on K-means. Overall, the *k* value is inversely proportional to the accuracy and directly proportional to efficiency. To determine the appropriate *k* value, the silhouette coefficient (SC) is adopted, which is an evaluation method of the clustering effect combining cohesion (C) and separation (S). SC is calculated using Eq. (19), where C represents the mean distance between a sample and all other points in the same class and S represents the mean distance between a sample and all other points in the next nearest cluster. SC ranges from −1 to 1. Furthermore, the larger the value, the better the clustering effect.

$$SC = \frac{S - C}{\max(C, S)} \tag{19}$$

Fig. 6 depicts the *k* value determination for the K-means based on the SC. Various embedding methods such as TransD and TransR were applied to determine the *k* value. The TransD-based clustering performance was the best from the SC perspective. Hence, the best semantic clustering effect was observed in translational models followed by deep learning models. However, semantic matching models exhibited the worst effect, which indirectly proved the semantic representation ability of various KGE models. Mostly, SC displays a normal distribution, and KGE models with poor semantic representation ability have a low SC. Therefore, the optimal KGE model and the appropriate *k* value can be selected based on the SC. Fig. 7 presents the *k* value determination for the K-NN algorithm based on TMR. The x- and y-axis represent the *k* value and number of entities paired per second, respectively. The EA performance was best when the *k* value was 1, and TransD was particularly prominent for all the models. According to the performance comparison presented in Figs. 6 and 7, the TransD model exhibits distinct performance advantages.

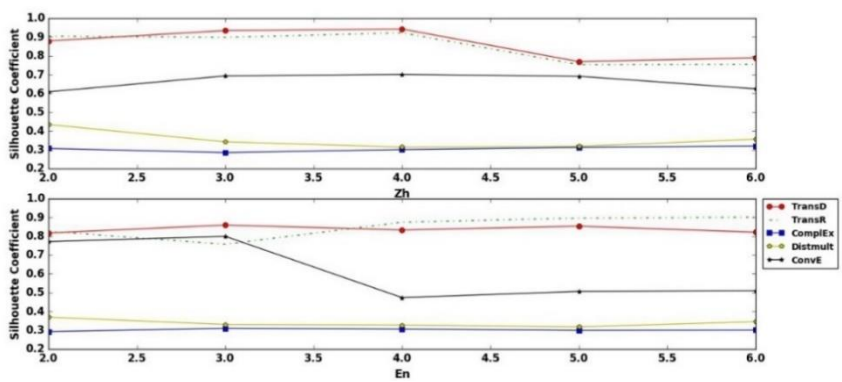


Fig. 6. *k* value determination for K-means based on SC.

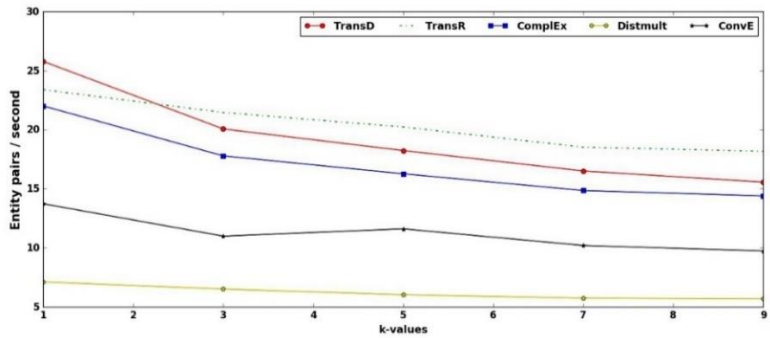


Fig. 7. *k* value determination for K-NN algorithm based on TMR.

4. PERFORMANCE EVALUATION

We adopted Hits@k as the evaluation benchmark. A Hits@k score is computed by measuring the proportion of correctly aligned entities ranked in the top *k* list. Table 6 lists a performance comparison of our two-stage EA method (referred as TwoStage) with the existing popular EA models. Due to the length of tables, we simply write H@k instead of Hits@k in the table. MRR (Mean Reciprocal Rank) and NDCG@k (Normalized Discounted Cumulative Gain) are also important rank-aware evaluation metrics. Compared to NDCG@k, however, MRR is simple to compute and widely used to evaluate the entity alignment. In order to make the evaluation more convincing, we decided to use MRR as well. JE, MTransE, JAPE, IPTransE, and BootEA are embedding-based EA methods, whereas GCN, R-GCNs, HGCNs, RDGCN, EMGCN, and TwoStage are GCN-based EA models, DuGa-DIT [35], REA [36], DAEA [37] and RALG [38] are GNN-based EA approaches. The experimental results verify that the proposed two-stage EA method (*i.e.*, TwoStage) has a certain effect, especially between Zh and En (Zh-En). DuGa-DIT's tiny win in Ja-En and Fr-En is mainly because of its intra-KG attention and cross-KG attention

Table 6. Performance comparison with other EA models for DBP15K datasets.

Models	Zh-En				Ja-En				Fr-En			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
JE	21.27	28.98	42.77	-	18.92	23.31	39.97	-	15.38	25.41	38.84	-
MTransE	30.83	46.71	61.41	0.364	27.86	41.85	57.45	0.349	24.41	41.57	55.55	0.335
JAPE	41.18	57.63	74.46	0.49	36.25	56.84	68.5	0.476	32.39	52.32	66.68	0.43
IPTransE	40.59	32	73.47	0.516	36.69	55.37	69.26	0.474	33.3	52.47	68.54	0.451
BootEA	62.94	75.41	84.75	0.703	62.23	77.89	85.39	0.701	65.30	73.52	87.44	0.731
GCN	41.25	58.62	74.38	0.61	39.91	60.05	74.46	0.59	37.29	59.86	74.49	0.64
GCNs	50.82	63.58	79.15	0.63	53.09	68.63	82.96	0.76	54.49	70.68	84.73	0.67
R-GCNs	46.57	60.37	74.29	0.651	48.68	68.35	77.82	0.74	51.11	72.65	80.07	0.66
HGCNs	72.1	77.25	85.7	0.76	76.6	84.21	89.7	0.81	89.2	92.32	96.1	0.91
RDGCN	69.7	76.34	84.2	0.75	76.3	80.65	89.7	0.81	87.3	92.54	95.12	0.901
EMGCN	86.25	91.27	94.62	0.79	86.63	93.47	95.19	0.87	93.95	95.51	98.89	0.87
DuGa-DIT	80.73	85.33	88.17	0.83	**91.44**	**93.68**	95.21	0.928	**98.17**	**98.68**	99.22	**0.985**
REA	28.9	48.6	67.5	0.61	29.3	49.8	75.1	0.602	30.4	53.9	80.3	0.72
DAEA	56.76	63.41	88.3	0.68	57.59	68.85	89.23	0.683	58.04	79.68	91.16	0.695
MRAEA	75.7	85.32	92.98	0.83	75.78	82.12	93.38	0.826	78.04	84.56	94.81	0.849
RALG	83.54	90.2	94.75	0.876	87.23	90.56	96.58	0.906	94.83	96.44	98.91	0.964
TwoStage	**88.58**	**92.42**	**95.88**	**0.917**	87.85	91.57	**96.63**	**0.93**	95.42	97.29	**99.42**	0.974

* We shaded the best-performing results for Hits@k.

mechanism. It can dynamically update cross-KG attention score matrices, which enables model to capture more cross-KG information. The MRAEA [39] model directly models cross-language entity embedding by focusing on meta-semantics of the incoming and outgoing neighbors of nodes and their connection relations. Its biggest highlight is the introduction of bi-directional iterative strategy for newly aligned seeds selection. However, there is no great alignment performance improvement in the benchmark evaluation. Comparatively, the accuracy of embedding-based models is lower than GCN-based and GNN-based models.

Table 7. Performance comparison with other EA models for SRPRS datasets.

Models	Fr-En				De-En				DBP-Wd				DBP-Yg			
	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR	H@1	H@5	H@10	MRR
JE	11.7	16.25	22.77	–	13.5	17.5	29.97	–	13.58	18.63	31.42	–	18.14	24.14	33.58	–
MTransE	21.3	32.41	44.7	0.29	10.7	15.6	24.8	0.16	23.8	41.54	50.7	0.33	19.6	28.63	40.1	0.29
JAPE	24.1	41.36	54.4	0.34	26.8	39.98	54.7	0.36	21.2	39.21	50.2	0.31	19.3	39.32	50.2	0.3
IPTransE	12.4	21.56	30.1	0.18	13.5	23.67	31.6	0.2	10.1	20.54	26.2	0.16	10.3	19.54	26.6	0.16
BootEA	36.5	52.31	64.9	0.46	50.3	60.85	73.2	0.58	38.4	48.63	66.7	0.48	38.1	57.64	65.1	0.45
GCN	24.3	39.23	52.2	0.34	38.5	51.87	60.8	0.46	29.1	43.54	55.6	0.38	31.9	44.65	58.6	0.41
GCNs	27.82	43.54	59.15	0.42	43.09	50.36	62.96	0.58	34.49	48.47	64.73	0.45	43.58	57.36	66.39	0.52
R-GCNs	26.5	32.56	44.69	0.38	38.47	46.52	58.6	0.49	31.67	47.23	58.32	0.39	40.23	52.74	59.24	0.44
HGCNs	67.28	72.66	77.45	0.69	76.54	83.4	86.87	0.8	88.09	98.36	99.27	0.93	91.1	91.86	93.7	0.91
RDGCN	67.2	70.52	76.7	0.71	77.9	84.6	88.6	0.82	97.4	98.2	99.4	0.98	93.5	93.65	94.7	0.93
EMGCN	65.26	75.36	85.52	0.77	66.73	70.84	75.49	0.66	88.65	89.21	89.65	0.88	84.36	85.31	86.54	0.85
DuGa-DIT	92.12	93.83	95.34	0.93	90.54	93.78	92.52	0.93	**98.88**	98.31	99.21	0.99	93.54	94.67	96.52	0.94
REA	45.65	74.54	89.85	0.74	51.33	63.54	78.65	0.67	68.57	75.36	86.32	0.75	71.56	80.65	88.63	0.76
DAEA	52.31	77.87	91.52	0.79	62.21	74.52	87.64	0.74	63.54	73.65	84.21	0.72	69.57	77.68	85.64	0.71
MRAEA	46.02	63.53	76.8	0.559	59.43	74.21	81.52	0.664	59.63	64.32	77.84	0.68	61.32	76.32	82.45	0.74
RALG	91.52	94.56	96.32	0.934	93.45	95.4	96.31	0.941	98.85	**99.12**	**99.96**	**0.993**	**99.25**	**99.54**	**99.89**	**0.994**
TwoStage	**92.25**	**95.33**	**97.12**	**0.94**	**94.89**	**96.41**	**98.36**	**0.95**	86.4	87.59	89.64	0.88	95.13	97.25	99.6	0.96

Table 7 presents a performance comparison with other EA models for the SRPRS datasets. Notably, the overall EA performance of SRPRS is lower compared to DBP15K, which indicates that these models might not perform well on relatively sparse KGs. Overall, the performance of GCN-based and GNN-based models was much better than that of embedded-based models, irrespective of the datasets DBP15K or SRPRS. Our method did not best performance exceptionally on the DBP-Wd and DBP-Yg dataset. Here, a rule is derived from the experimental analysis. The homogeneity rate derived in Eq. (20) is correlated positively to the TwoStage method accuracy. Table 8 lists the relationship between the homogeneity rate and EA accuracy with Hits@1 based on the TwoStage method. Overall, the higher the homogeneity rate, the higher the matching accuracy for proposed TwoStage method. This is also verified by RALG model. However, the RALG model is an exception, unlike other GNN-based models, it is a relation-aware line graph neural network. It learns entity-independent relational representations for heterogeneous line graphs. The constructed heterogeneous line graph can explicitly capture the correlation of relationships and a new aggregation way is designed in the form of triples to strengthen the correlation and correspondence between entities. The utilization of the above two methods (*i.e.*, TwoStage and RALG) greatly reduces the influence of homogeneous entities on entire EA process.

$$Homogeneity\ Rate = \frac{homogeneous\ entities}{homogeneour\ entities + heterogenerous\ entitites} \tag{20}$$

Table 8. Relationship between homogeneity rate and EA accuracy.

	DBP15K (Zh-En)	DBP15K (Ja-En)	DBP15K (Fr-En)	SRPRS (Fr-En)	SRPRS (De-En)	SRPRS (DBP-Wd)	SRPRS (DBP-Yg)
Homogeneity Rate	0.0352	0.026	0.501	0.39	0.6418	0	0.714
EA Accuracy	88.58	87.85	95.42	92.25	94.89	86.4	95.13

Table 9 presents the EA performance of the TwoStage method under different activation functions. Overall, the training efficiency of ReLU and its variants was the highest. Except for ReLU, which exhibited the best performance between Zh and En (Zh-En), ELU activation function exhibits the best alignment performance in more than 50% of all cases. In terms of training time, ELU performs the best for over 80% of all cases. Therefore, considering the efficiency and performance, ELU is selected as the activation function.

Table 9. EA performance of TwoStage method for DBP15K datasets.

Activation Functions	Zh-En		Ja-En		Fr-En	
	H@1	Training Time	H@1	Training Time	H@1	Training Time
Sigmoid	69.78	23,846	76.77	25,124	91.24	23,506
Swish	77.49	29,515	83.24	23,260	91.68	25,588
ReLU	**88.95**	45,512	85.32	41,809	91.2	36,707
Leaky ReLU	78.4	24,538	84.62	24,294	91.92	25,606
SELU	77.7	24,539	84.64	23,204	91.57	22,256
ELU	88.58	**21,985**	**87.85**	**23,073**	**95.42**	**22,233**

* The unit of training time is seconds.

Table 10. EA performance of TwoStage method for SRPRS datasets.

Activation Functions	Fr-En		De-En		DBP-Wd		DBP-Yg	
	H@1	Training Time	H@1	Training Time	H@1	Training Time	H@1	Training Time
Sigmoid	92.24	13,369	93.87	21,282	59.84	11,475	90.06	12,513
Swish	92.13	12,932	94.19	16,465	79.95	22,419	91.71	13,273
ReLU	92.05	45,653	94.03	26,588	86.24	34,309	95.11	34,317
Leaky ReLU	96.67	16,464	94.86	16,454	85.27	**11,459**	92.92	12,412
SELU	**96.74**	16,622	93.57	16,622	85.87	22,727	**95.67**	13,244
ELU	92.25	**12,853**	**94.89**	**16,190**	**86.4**	22,760	95.13	**12,259**

According to the training results in Tables 11 and 12, the training time of Canberra was the shortest for more than 85% of all cases. Canberra exhibited excellent performance in DBP15K(Ja-En), DBP15K(Fr-En), SRPRS(De-En), and SRPRS(DBP-Wd) datasets. Therefore, the Canberra calculation method was adopted as the evaluation benchmark of EA.

Table 11. Alignment performance under different distance methods for DBP15K datasets.

Distance Method	Zh-En		Ja-En		Fr-En	
	H@1	Training Time	H@1	Training Time	H@1	Training Time
Euclidean	79.31	23,457	85.75	23,678	91.44	22,904
Manhattan	**91.89**	22,233	87.29	33,073	91.89	26,707
Cosine	77.07	**19,783**	84.13	23,340	92.73	22,778
Chebyshev	42.07	23,876	51.44	30,377	59.67	26,030
Canberra	88.58	21,985	**87.85**	**23,073**	**95.42**	**22,233**

Table 12. Alignment performance under different distance methods for SRPRS datasets.

Distance Method	Fr-En		De-En		DBP-Wd		DBP-Yg	
	H@1	Training Time	H@1	Training Time	H@1	Training Time	H@1	Training Time
Euclidean	96.25	15,725	94.44	16,114	83.33	17,754	92.19	15,977
Manhattan	91.11	45,653	94.03	26,588	86.24	34,309	**95.17**	34,317
Cosine	**96.9**	15,734	94.56	16,104	80.5	18,209	92.44	15,678
Chebyshev	73.62	16,038	77.97	16,549	44.98	18,531	73.49	16,376
Canberra	92.25	**12,853**	**94.89**	**15,590**	**86.4**	**15,760**	95.13	**12,259**

We adopted recall (*R*), precision (*P*), and *F* score as the evaluation metrics of the TwoStage method. Fig. 8 presents the training process of the TwoStage method for the DBP15K and SRPRS datasets. The recall is computed as the number of correct matches (*C*) divided by the number of correctly matched entities and missing entities (*M*). Precision was computed as *C* divided by the number of all corrected matches and irrelevant matches (*I*). The F-score is the harmonic mean between recall and precision, as presented in Eq. (21).

$$R = \frac{C}{C+M} \quad P = \frac{C}{C+I} \quad \text{and} \quad F = \frac{2P*R}{P+R} \tag{21}$$

Our method does not overfit according to the training accuracy in Fig. 8 (a) and test accuracy (Fig. 8 (b)) of the TwoStage method. Combining Table 8 and Fig. 8 (b) (test accuracy), we can conclude that with an increase in homogeneity rate, the initial accuracy increases. For example, the initial accuracies of DBP15K (Fr-En), SRPRS (Fr-En), SRPRS (De-En), and SRPRS (DBP-Yg) exceeded 87%. According to the *F* score, the method reached its peak performance when the epoch was 100.

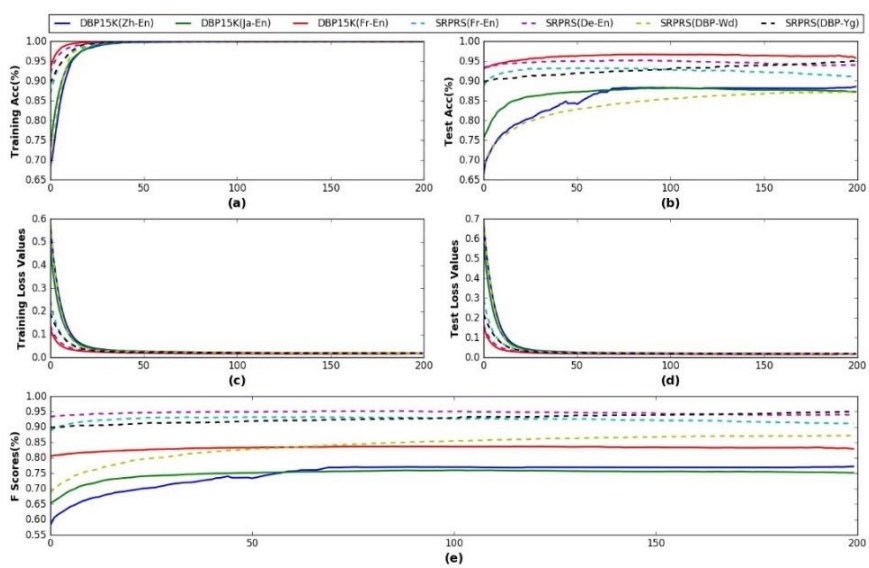


Fig. 8. Training process based on TwoStage for DBP15K and SRPRS datasets: (a) Training accuracy; (b) Test accuracy; (c) Training loss values; (d) Test loss values; and (e) *F* scores.

Fig. 9 presents the computation time comparison between GCN- and embedding-based models. The time cost of embedding-based models (*e.g.*, MtransE and JAPE) exceeds GCN-based models (*e.g.*, RDGCN and EMGCN). However, our proposed TwoStage method has the highest efficiency among all the GCN-based models. Compared to similar GCN-based models, the efficiency of the TwoStage method is greatly improved because of the addition of a *k*-dimensional tree in our model.

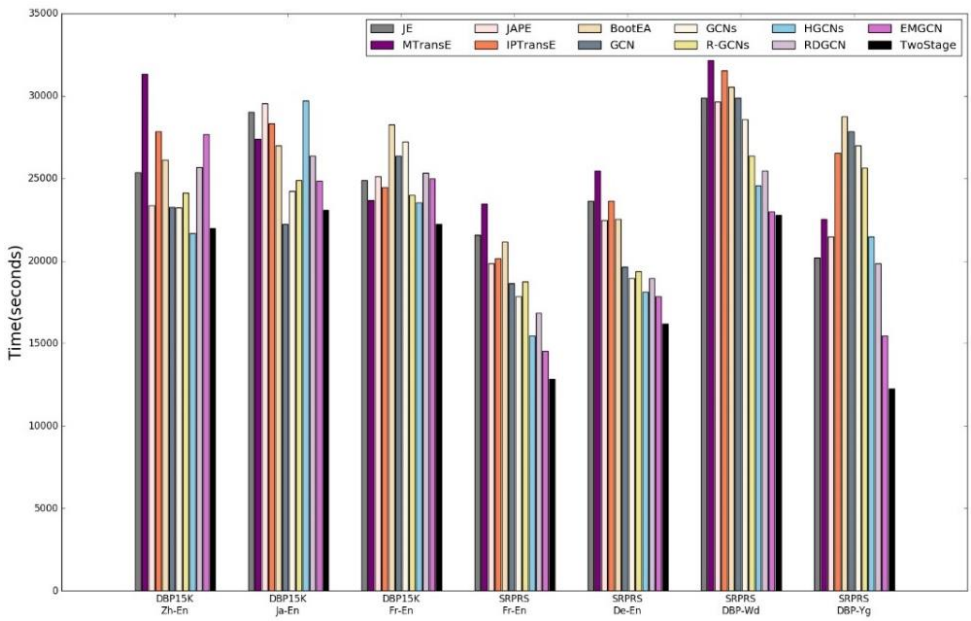


Fig. 9. Computational time comparisons for DBP15K dataset.

Table 13 lists the subset size and memory usage before and after removing homogeneous entities. Theoretically, TwoStage will consume more memory because the homogeneity weight layer is added on the basis of RDGCN. However, the peak memory usage is reduced, because homogenous entities are removed from the subset to reduce computation especially in DBP-Yg.

Table 13. Subset size and memory usage before and after removing homogeneous entities.

		DBP15K Zh-En	DBP15K Ja-En	DBP15K Fr-En	SRPRS Fr-En	SRPRS De-En	SRPRS DBP-Wd	SRPRS DBP-Yg
Size (MB)	Before	167	169	166	125	153	153	129
	After	164	159	145	113	125	153	88
	Reduce (%)	1.8	5.9	12.7	9.6	18.3	0	31.8
Memory usage (MB)	RDGCN	15118.0	15252.5	16562.2	8131.6	8534.0	7369.1	10057.0
	TwoStage	15086.2	14962.7	15117.4	7513.6	7463.8	7488.5	7402.4
	Reduce (%)	0.21	1.9	8.74	7.6	12.54	-0.16	26.4

Table 14 shows alignment performance before and after noise label correction. Inspired by REA model, we consider the impact of annotation errors to TwoStage model.

We adopted a simple dictionary correction method provided by Google Translate APIs to correct entity labels. In theory, it can increase the proportion of homogeneous entities to improve EA alignment performance, but not obvious for TwoStage. The main reason is that the dictionary correction method is too naive at some specialized entity labels. In future research, we will focus on more efficient noisy entity labels correction.

Table 14. Alignment performance before and after noise label correction.

	DBP15K Zh-En	DBP15K Ja-En	DBP15K Fr-En	SRPRS Fr-En	SRPRS De-En	SRPRS DBP-Wd	SRPRS DBP-Yg
	H@1	H@1	H@1	H@1	H@1	H@1	H@1
Before	88.58	87.85	95.42	92.25	94.89	86.4	95.13
After	90.4	89.05	96.22	93.75	97.59	88.8	96.93
Improve (%)	1.6	1.2	0.8	1.5	2.7	2.4	1.8

5. CONCLUSION AND FUTURE RESEARCH

In this study, an embedding-based Two-Stage EA strategy is proposed to divide the entity relationships into homogeneous and heterogeneous. Regarding the homogeneous entities, semantic clustering and pairing based on the centroid distance are performed and then combined with a *k*-dimensional tree and Linked-Lists to match homogeneous entities. In the clustering process and alignment, SC and TMR were used to determine the *k* value in the K-means and K-NN algorithms. For the heterogeneous entities, the E-GCN model with a graph attention mechanism is adopted to enhance the features on nodes, where the index structures of the first stage are reused to enhance the matching speed. The experimental results revealed that GCN-based models exhibited better EA performance than embedded-based models. Particularly, the proposed TwoStage method significantly improves the EA performance compared to state-of-the-art EA models. The training time of the TwoStage model was reduced by 43.2%. The alignment accuracies were 88.58, 87.85, and 95.42% for Zh-En, Ja-En, and Fr-En of the DBP15K datasets, respectively, and 92.25, 94.89, 86.4, and 95.13% for Fr-En, De-En, DBP-Wd, and DBP-Yg of the SRPRS datasets, respectively. Moreover, the mainstream semantic distance method was also evaluated, and the experimental results presented that the Canberra distance was the most appropriate. Additionally, the influence of different activation functions on the model accuracy was explored. Consequently, ELU was selected based on the accuracy and training time efficiency. However, some problems were encountered during the experiments. For example, the semantic clustering results were different each time owing to the characteristics of the K-means clustering algorithm. Our attempt to use density-based clustering algorithms failed because of their low efficiency. Therefore, in future research, we will optimize the semantic clustering algorithm to find a balance between its efficiency and effectiveness. Existing studies mainly adopt different attention mechanisms to improve relation representation and have achieved success for low homogenous rate. However, these methods place great demands on computing power. We believe that reducing entity annotation errors can broadly improve matching accuracy, which is also reflected in Table 14. Therefore, using a more intelligent approach to reduce entity annotation errors is the most effective way to solve EA. We have a plan to conduct case studies with concrete examples to demonstrate the effectiveness of considering homogeneous entities.

REFERENCES

1. M. Chen, Y. Tao, M. Yang, and C. Zaniolo, "Multilingual knowledge graph embeddings for cross-lingual knowledge alignment," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2017, pp. 1511-1517.
2. H. Zhu, R. Xin, Z. Liu, and M. Sun, "Iterative entity alignment via joint knowledge embeddings," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 4258-4261.
3. Z. Sun, W. Hu, Q. Zhang, and Y, Qu, "Boot-strapping entity alignment with knowledge graph embedding," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2018, pp. 4396-4402.
4. M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Bery, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of European Semantic Web Conference*, LNCS Vol. 10843, 2018, pp. 593-607.
5. F. Monti, O. Shchur, A. Bojchevski, O. Litany, S. Günnemann, and M. M. Bronstein, "Dual-primal graph convolutional networks," *arXiv Preprint*, 2018, arXiv:1806.00770, pp. 1-11.
6. Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao. "Relation-aware entity alignment for heterogeneous knowledge graphs," in *Proceedings of International Joint Conference on Artificial Intelligence*, 2019, pp. 5278-5284.
7. A. Bordes, N. Usunier, and A. Garcia-duran, "Translating embeddings for modeling multi-relational data," in *Proceedings of International Conference on Neural Information Processing Systems*, 2013, pp. 2787-2795.
8. Z. Wang, J. Zhang, J. Feng, and Z. Chen. "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of National Conference on Artificial Intelligence*, 2014, pp. 1112-1119.
9. Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of National Conference on Artificial Intelligence*, 2015, pp. 2181-2187.
10. G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of Asian Federation of Natural Language Processing*, 2015, pp. 687-696.
11. T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of International Conference on Learning Representations*, 2013, pp. 1-12.
12. B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of International Conference on Learning Representations*, 2015, pp. 1-12.
13. T. Trouillon, J. Welbl, S. Riedel. E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of International Conference on Machine Learning*, 2016, pp. 3021-3032.
14. M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 1955-1961.
15. S. Kazemi and D. Poole, "SimplE embedding for link prediction in knowledge graphs," *Advances in Neural Information Processing Systems*, 2016, pp. 1955-1961.

16. Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *Proceedings of the 7th International Conference on Learning Representations*, 2019, pp. 1-18.
17. I. Balazevic, C. Allen, and T. M. Hospedales, "Rotate: Tucker: Tensor factorization for knowledge graph completion," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2020, pp. 5185-5194.
18. B. Shi and T. Weninger, "ProjE: Embedding projection for knowledge graph completion," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 1236-1242.
19. T. Dettmers, "Convolutional 2D knowledge graph embeddings," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 1811-1818.
20. D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A novel embedding model for knowledge base completion based on convolutional neural network," in *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 327-333.
21. L. Guo, Q. Zhang, W. Ge, W. Hu, and Y. Qu, "DSKG: A deep sequential model for knowledge graph completion," *Communications in Computer and Information Science*, Vol. 957, 2019, pp. 65-77.
22. H. Ee, R. Chen, M. Song, P. Zhu, and Z. Wang, "A joint embedding method of relations and attributes for entity alignment," *International Journal of Machine Learning and Computing*, Vol. 10, 2020, pp. 605-611.
23. Z. Sun, W. Hu, and C. Li, "Cross-lingual entity alignment via joint attribute-preserving embedding," *Lecture Notes in Computer Science*, Vol. 10587, 2017, pp. 628-644.
24. Z. Sun, Q. Zhang, W. Hu, and C. Wang, "A benchmarking study of embedding-based entity alignment for knowledge graphs," in *Proceedings of the VLDB Endowment*, 2020, pp. 2326-2340.
25. Z. Wang, Q. Lv, X. Lan, and Y. Zhang, "Cross-lingual knowledge graph alignment via graph convolutional networks," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 349-357.
26. N. T. Tam, H. T. Trung, H. Yin, T. V. Vinh, D. Sakong, B. Zheng, and N. Q. V. Hung, "Entity alignment for knowledge graphs with multi-order convolutional networks," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4347, 2020, pp. 1-14.
27. I. Dokmanic, R. Parhizkar, J. Ranieri, and M. Vetterli, "Euclidean distance matrices: Essential theory, algorithms, and applications," *IEEE Signal Processing Magazine*, Vol. 32, 2015, pp. 12-30.
28. Y. Sari, P. Prakoso, and A. Baskara1, "Evaluation the influence of distance-based K-means method for detecting moving vehicles," in *Proceedings of International Conference on Science in Engineering and Technology*, 2022, pp. 1-6.
29. A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *Proceedings of the 4th International Conference on Cyber and IT Service Management*, 2016, pp. 1-6.
30. A. Pulungana and M. Zarlis, "Analysis of braycurtis, canberra and euclidean distance in KNN algorithm," *Journal Publications & Informatics Engineering Research*, Vol. 4, 2019, pp. 74-77.
31. K. Zeng, C. Li, L. Hou, J. Li, and L. Feng, "A comprehensive survey of entity alignment for knowledge graphs," *AI Open*, Vol. 2, 2021, pp. 1-13.

32. A. Ameen, K. U. R. Khan, and B. P. Rani, "Reasoning in semantic web using jena," *Computer Engineering and Intelligent Systems*, Vol. 5, 2014, pp. 39-47.
33. L. Guo, Z. Sun, and W. Hu, "Learning to exploit long-term relational dependencies in knowledge graphs," in *Proceedings of International Conference on Machine Learning*, 2019, pp. 2505-2514.
34. X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li, "OpenKE: An open toolkit for knowledge embedding," in *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 139-144.
35. Z. Xie, R. Zhu, K. Zhao, J. Liu, G. Zhou, and J. Huang, "Dual gated graph attention networks with dynamic iterative training for cross-lingual entity alignment," *ACM Transactions on Information Systems*, Vol. 40, 2022, pp. 1-30.
36. S. Pei, L. Yu, G. Yu, and X. Zhang, "REA: Robust cross-lingual entity alignment between knowledge graphs," in *Proceedings of International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 2175-2184.
37. J. Sun, Y. Zhou, and C. Zong, "Dual attention network for cross-lingual entity alignment," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 3190-3201.
38. Y. Zhang, J. Wu, K. Yu, and X. Wu, "Independent relation representation with line graph for cross-lingual entity alignment," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 14, 2022, pp. 1-30.
39. X. Mao, W. Wang, H. Xu, M. Lan, and Y. Wu, "MRAEA: An efficient and robust entity alignment approach for cross-lingual knowledge graph," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 420-428.

Yuxiang Sun is a Postdoctoral Researcher at the Institute of Software Technology, Kyungpook National University, South Korea. He graduated from Kyungpook National University with a master's degree and a doctorate degree in 2019 and 2022. His research interests include semantic retrieval, semantic entity matching, and databases.

Yongju Lee received the Ph.D. degree in Information and Communication Engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 1997. He is currently a Professor at the School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea. He has published more than 150 papers in domestic and international conferences and journals. His current research interests include semantic web technology, artificial intelligence, knowledge graph, and big data.