World Wide Web (2023) 26:4069–4092
https://doi.org/10.1007/s11280-023-01221-8

Entity alignment via graph neural networks: a component-level study

Yanfeng Shu[1] · Ji Zhang[2] · Guangyan Huang[3] · Chi-Hung Chi[4] · Jing He[5]

Received: 10 July 2023 / Revised: 3 November 2023 / Accepted: 15 November 2023 /
Published online: 29 November 2023

Abstract
Entity alignment plays an essential role in the integration of knowledge graphs (KGs) as it seeks to identify entities that refer to the same real-world objects across different KGs. Recent research has primarily centred on embedding-based approaches. Among these approaches, there is a growing interest in graph neural networks (GNNs) due to their ability to capture complex relationships and incorporate node attributes within KGs. Despite the presence of several surveys in this area, they often lack comprehensive investigations specifically targeting GNN-based approaches. Moreover, they tend to evaluate overall performance without analysing the impact of individual components and methods. To bridge these gaps, this paper presents a framework for GNN-based entity alignment that captures the key characteristics of these approaches. We conduct a fine-grained analysis of individual components and assess their influences on alignment results. Our findings highlight specific module options that significantly affect the alignment outcomes. By carefully selecting suitable methods for combination, even basic GNN networks can achieve competitive alignment results.

Keywords Knowledge graph · Entity alignment · Graph neural network · Experimental study

✉ Yanfeng Shu
Yanfeng.Shu@csiro.au

Ji Zhang
Ji.Zhang@usq.edu.au

Guangyan Huang
guangyan.huang@deakin.edu.au

Chi-Hung Chi
chihungchi@gmail.com

Jing He
jing.he@ndcn.ox.ac.uk

1 Data61, CSIRO, Hobart, Australia

2 University of Southern Queensland, Brisbane, Australia

3 Deakin University, Melbourne, Australia

4 Nanyang Technological University, Singapore, Singapore

5 Oxford University, Oxford, England

1 Introduction

Knowledge graphs (KGs) serve as structured, graph-based representations of knowledge, capturing real-world entities, their attributes, and the relationships between them. They are indispensable tools, facilitating sophisticated data analysis, inference, and decision-making processes. KGs come in various forms, including general KGs like DBpedia [1] and YAGO [2], as well as domain-specific KGs like BioKG [3] and FoodKG [4], catering to a wide range of applications. However, a common challenge with standalone KGs is their incompleteness, lacking comprehensive domain coverage. To overcome this limitation, KG integration becomes essential. By combining KGs from diverse sources, integration enables the presentation of different perspectives and complementary information. One crucial step in KG integration is entity alignment, which involves identifying entities across KGs that refer to the same real-world objects. Aligning entities allows the development of advanced applications that offer a holistic view of information, enhancing the quality of knowledge-based systems.

Recent research in entity alignment has primarily focused on embedding-based approaches. These approaches represent entities as low-dimensional vectors, capturing semantic relatedness by computing distances in the vector space. Among them, graph neural networks (GNNs) [5, 6] have gained popularity for embedding learning. GNNs effectively learn node representations by aggregating information from neighboring nodes recursively. The underlying assumption behind using GNNs for entity alignment is that similar entities tend to have similar neighborhoods, as supported by the expressiveness of GNNs in identifying isomorphic subgraphs, akin to the Weisfeiler-Lehman (WL) algorithms [7]. Moreover, GNNs naturally excel at handling complex graph structures and incorporating node attributes, making them promising for entity alignment tasks. However, the introduction of GNNs into entity alignment has led to more intricate embedding architectures, complicating the interpretation of an approach's effectiveness as it becomes hard to discern whether the effectiveness is due to the embedding itself or other components of the alignment process.

Despite several surveys on embedding-based entity alignment approaches [8–11], they often fail to specifically examine GNN-based approaches, overlooking key characteristics of GNNs that are crucial for entity alignment. Additionally, while these surveys assess the overall effectiveness of the approaches, they typically overlook the impact of individual components and methods on performance. To fill this gap, our work offers a fine-grained analysis of individual components and their impacts. We contribute to the field by providing:

- A general framework that encompasses the fundamenal components of GNN-based entity alignment approaches, along with a categorisation of these approaches based on the key characteristics associated with these components.
- A comprehensive component-level experimental study conducted on representative datasets, evaluating the impact of different components and their combinations on the overall performance.

Our analysis reveals that certain module options have a significant impact on performance, such as combining entity name initialisation with skip connections for embedding and employing iterative training with CSLS as the enhanced distance metric. We demonstrate that, by selecting suitable methods for combination, even basic GNN networks can achieve competitive results. This study provides valuable insights into the design and optimisation of GNN-based approaches for entity alignment, advancing the understanding and applicability of these methods in knowledge graph integration tasks.

The rest of the paper is organised as follows. Section 2 provides preliminaries, including problem definition and a summary of related work. Section 3 presents a general framework for GNN-based entity alignment approaches. Section 4 discusses the importance of component-level analysis and Section 5 reports analysis results. Finally, Section 6 concludes the paper.

2 Preliminaries

2.1 Problem definition

We define a KG as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{A}, \mathcal{V}, \mathcal{T})$, where $\mathcal{E}$, $\mathcal{R}$, $\mathcal{A}$, $\mathcal{V}$ and $\mathcal{T}$ are sets of entities, relations, attributes, values, and triples respectively. $\mathcal{T}$ consists of relation triples $\mathcal{T}^r$ and attribute triples $\mathcal{T}^a$, where $\mathcal{T}^r \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, and $\mathcal{T}^a \subseteq \mathcal{E} \times \mathcal{A} \times \mathcal{V}$. Given two KGs, $\mathcal{G}_1 = (\mathcal{E}_1, \mathcal{R}_1, \mathcal{A}_1, \mathcal{V}_1, \mathcal{T}_1)$ and $\mathcal{G}_2 = (\mathcal{E}_2, \mathcal{R}_2, \mathcal{A}_2, \mathcal{V}_2, \mathcal{T}_2)$, the goal of entity alignment is to find aligned entities $\Phi = \{(e_1, e_2)|e_1 \in \mathcal{E}_1, e_2 \in \mathcal{E}_2\}$, where e_1 and e_2 refer to the same real-world object. In many cases, a small subset of Φ, i.e., pre-aligned entities, is provided and used as training data for finding new alignments.

2.2 Related work

GNNs Many learning tasks involve complex relationships and dependencies within graph data, which cannot be effectively handled by standard neural networks like convolutional neural networks (CNNs) [12] and recurrent neural networks (RNNs) [13]. These networks are specifically designed for Euclidean domains like images and text, making them less effective in tackling the complexities of graph-based data. To address this, graph neural networks (GNNs) have emerged. Initially introduced in [14], GNNs learned node representations by iteratively exchanging information with neighbours until a stable fixed point was reached. Subsequent works on GNNs largely relax the fixed point assumption, employing stacked graph convolutional layers to extract higher-level node representations. Representative GNNs include graph convolutional network (GCN) [15], graph attention network (GAT) [16] and gated graph neural network (GGNN) [17]. For a detailed understanding of GNNs and taxonomies, interested readers can refer to recent surveys [5, 6].

Entity linking/matching Tasks similar to entity alignment have been addressed under different names depending on fields or applications. Entity linking or entity disambiguation aims to identify entity mentions in natural language text and map them to corresponding entries in a KG. Previous research [18–20] has predominantly utilised contextual information, including local contexts of entity mentions and document-level coherence of referenced entities, for disambiguation. On the other hand, entity matching, entity resolution, or record linkage involves matching records from different relational tables that refer to the same entities [21–24]. When applied within the same relational table, it is referred to as deduplication. The matching process involves comparing attribute values using specific similarity measures and aggregating comparison results across all attributes. To reduce the number of record pairs to compare, indexing or blocking techniques are commonly employed to filter out obvious non-matching pairs [22].

Entity alignment on KGs Conventional approaches for mapping entities between KGs include concept-level matching [25–27], instance-level matching [28, 29], or a combination

of both [30], depending on whether the entities being aligned are concepts or instances. Graph structures and entity properties, such as string representations, are commonly used for identifying alignments. Additionally, when KGs contain richer representations like RDFS or OWL, logic reasoning can be employed to deduce correspondences. Embedding-based approaches are generally classified as translation-based or GNN-based. Translation-based approaches, e.g.,[31–35], employ translational models such as TransE [36] to learn entity embeddings, treating relations as translations between entities. In contrast, GNN-based approaches, e.g.,[37–40], utilise GNN models for learning entity embeddings, as we will discuss in Section 3. Several studies [8–11] have conducted empirical evaluations on representative embedding-based approaches. These studies either introduce new benchmark datasets for evaluation or provide new implementations of approaches using specific libraries or toolkits developed for embedding-based entity alignment. Notably, one study [11] categorises approaches based on different settings, such as whether additional information beyond graph structure is used for alignment, and compares results within and across these categories. While these studies offer valuable insights into the overall effectiveness of these approaches, they lack a detailed analysis of individual components and their impact on performance. Our work complements these studies by conducting a thorough analysis at the component level, with a focus on GNN-based approaches.

3 A general framework

Many recent entity alignment approaches rely on graph neural networks (GNNs) as their underlying learning architecture. Figure 1 presents a general framework that encompasses GNN-based approaches, with optional components indicated by dashed lines. There are three main modules: an embedding module (GNNs), an alignment training module, and an alignment inference module. The embedding module and the training module jointly constitute the embedding learning modules for entity alignment. The framework takes as input two knowledge graphs and learns embeddings for entities. Based on these embeddings, it generates alignments between the entities in the two graphs. If pre-aligned entities are provided, they serve as seed alignments to guide the learning process. Furthermore, the alignment results generated by the inference module can be leveraged to expand these seed alignments. Table 1 provides a categorisation of representative GNN-based approaches based on their key characteristics associated with the three modules.

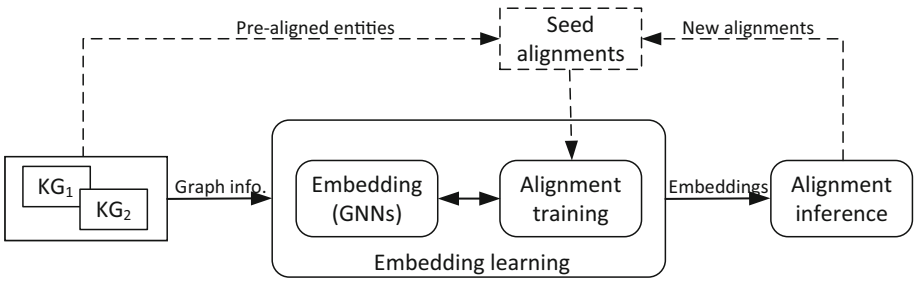

Figure 1 A GNN-based entity alignment framework

Table 1 Categorisation of representative GNN-based entity alignment approaches

Approach	Embedding (GNNs)											Alignment training			Alignment inference		
	Information used								Structure embedding			Strategy	Cost func.	Negative sampling	Strategy	Dist. meas.	CSLS
	TC	RT	RN	EN	ED	IM	AT	AV	Entity init.	Nbr. aggr.	Skip conn.						
GCN-Align [39]	√						√		Random	GCN		Superv.	Triplet	Uniform	NN	L_1	
AVG-GCN [47]	√	√							Random	GCN		Superv.	Triplet	Uniform	NN	L_1	
HMAN [48]	√	√			√		√		Random	GCN		Superv.	Triplet	Uniform	NN	L_1	
KECG [49]	√	√							Random	GAT		Superv.	Triplet	Nearest	NN	L_2	
MUGNN [37]	√	√							Random	GAT		Superv.	Triplet	Nearest	NN	L_2	
AliNet [38]	√	√							Random	Hybrid	Concat.	Superv.	Constr.	Nearest	NN	Cos.	√
CEA [50]	√			√					Random	GCN		Superv.	Triplet	Uniform	SM	L_1	
AttrGNN [51]	√			√			√	√	Random	GCN		Superv.	Triplet	Nearest	NN	Cos.	
MRAEA [52]	√	√							Random	GAT*	Concat.	Semi-	Triplet	Nearest	NN	Cos.	√
RREA [53]	√	√							Random	GAT*	Concat.	Semi-	Triplet	Nearest	NN	Cos.	√
Dual-AMN [54]	√	√							Random	GAT*	Highway	Semi-	NHSM	LogSumExp	NN	Cos.	√
KE-GCN [55]	√	√							Random	GCN*		Superv.	Triplet	Uniform	NN	L_1	
EVA [56]	√	√				√	√		Random	GCN		Semi-	NCA	Nearest	NN	Cos.	√
MCLEA [57]	√	√		√		√	√		Random	GAT		Semi-	ICL,IAL	Nearest	NN	Cos.	√
GMNN [58]	√			√					Entity name	GCN		Superv.	CE	Nearest	NN	Prob.	
HGCN [40]	√	√		√					Entity name	GCN	Highway	Superv.	Triplet	Nearest	NN	L_1	
RDGCN [59]	√	√		√					Entity name	Hybrid	Highway	Superv.	Triplet	Nearest	NN	L_1	
NMN [60]	√			√					Entity name	GCN	Highway	Superv.	Triplet	Nearest	NN	L_1	
RNM [61]	√	√		√					Entity name	GCN	Highway	Superv.	Triplet	Nearest	IM	L_1	
RAGA [62]	√	√		√					Entity name	Hybrid	Highway	Superv.	Triplet	Nearest	SM	L_1	
SelfKG [63]	√			√					Entity Name	GAT		UnSup.	NCE	Self-	NN	L_2	
ICLEA [64]	√	√	√	√	√				Entity Name/Des.	GAT		UnSup.	NCE	Self-	NN	L_2	

(TC, RT, RN, EN, ED, IM, AT and AV mean topological connections, relation types, relation names, entity names, entity descriptions, images, attribute types, and attribute values respectively; blank cells mean "N/A"; GCN*, GAT* represent GCN and GAT variants respectively)

3.1 Embedding (GNNs)

The embedding module aims to embed a KG into a vector space, representing entities as embeddings. Different types of KG information can be used by the embedding module, including graph structure (topological connections), relations (relation types and names), attributes (attribute types and names), and values (entity names, descriptions and images are considered as special cases of attributes and values and are treated differently). Among these, graph structure is the most basic one. Accordingly, structure embedding, which focuses on embedding the structure information of entities, forms the core part of the embedding module. Other types of information, such as relations, attributes, and values, can be incorporated into structure embedding to provide a more comprehensive representation of entities.

As observed from Table 1, all approaches use graph structure for embedding. Many approaches also incorporate relation types or entity names. On the other hand, attributes and values are explored to varying degrees. While GCN_Align, HMAN, EVA and MCLEA use attribute types, AttrGNN uses both attribute types and values. Additionally, HMAN incorporates entity descriptions, EVA and MCLEA leverage images, and ICLEA uses relation names in addition to relation types, entity names, and descriptions. To learn from different information types, separate channels can be employed. For instance, AttrGNN uses four channels for learning representations of graph structure, entity name, literal attribute, and digital attribute, respectively. Alternatively, certain information like entity names or relations can be incorporated directly into structure embedding, as we will discuss. In terms of structure embedding, existing approaches differ in, among other things, how they initialise entity representations, how they aggregate neighbours' information and how they obtain the final entity representations.

Entity initialisation While some approaches like AliNet and MRAEA initialise entities randomly, others such as HGCN and SelfKG initialise entities with specific feature vectors. In the latter case, entity names are commonly used to derive the initial features of entities, often by leveraging pre-trained language models. ICLEA goes a step further by incorporating entity descriptions. It obtains an entity's initial embedding by concatenating its name embedding and description embedding to create a comprehensive representation.

Neighbourhood aggregation A key feature of GNN-based embedding is that an entity's representation is updated by recursively aggregating the entity's neighbourhood information. At each GNN layer, the following updates are typically performed [41]:

$$m_{e_i}^{l+1} \leftarrow Aggregate(\{\mathbf{h}_{e_j}^{l}, \forall e_j \in N_{e_i}\}) \tag{1}$$

$$\mathbf{h}_{e_i}^{l+1} \leftarrow \sigma(\mathbf{W}^{l} m_{e_i}^{l+1}) \tag{2}$$

where $\mathbf{h}_{e_i}^{l}$ represents the embedding of e_i at layer l, N_{e_i} the set of immediate neighbors of e_i (including e_i), $m_{e_i}^{l+1}$ the aggregated representation of N_{e_i}, $\mathbf{W}^{l}$ the transformation matrix, and $\sigma(\cdot)$ an activation function. Equation 1 is responsible for aggregating information from immediate neighbours, while Equation 2 is for transformation (typically non-linear) of the aggregated information. GCN and GAT are two basic GNN models, and their main difference is in the way they aggregate neighbours' information. While GCN performs neighborhood aggregation by normalised mean pooling [15]:

$$\mathbf{h}_{e_i}^{l+1} = \sigma\Big(\sum_{e_j \in N_{e_i}} \frac{1}{\sqrt{d_{e_i} d_{e_j}}} \mathbf{W}^{l} \mathbf{h}_{e_j}^{l} \Big) \tag{3}$$

where d_{e_i} represents the degree of e_i, GAT accomplishes aggregation through attentional weighted summation [16]:

$$\mathbf{h}_{e_i}^{l+1} = \sigma\Big(\sum_{e_j \in N_{e_i}} a_{ij}^{l} \mathbf{W}^{l} \mathbf{h}_{e_j}^{l} \Big) \tag{4}$$

$$a_{ij}^{l} = \frac{exp(LeakyReLu(\vec{v}^{T}[\mathbf{W}^{l}\mathbf{h}_{e_i}^{l} \| \mathbf{W}^{l}\mathbf{h}_{e_j}^{l}]))}{\sum_{e_k \in N_{e_i}} exp(LeakyReLu(v^{T}[\mathbf{W}^{l}\mathbf{h}_{e_i}^{l} \| \mathbf{W}^{l}\mathbf{h}_{e_k}^{l}]))} \tag{5}$$

where a_{ij}^{l} is the attention coefficient at layer l, $\vec{v}$ is the weight vector, $\cdot^{T}$ represents transposition and $\|$ is the concatenation operation. In practice, GAT often employs multi-head attention to stabilise the learning process, by using K independent attention mechanisms, where K represents the number of attention heads, and merging their outputs through concatenation or averaging. Table 1 shows that all approaches are based on GCN, or GAT, or their variants or hybrids for neighbourhood aggregation. Note that some approaches, e.g., MRAEA and KE-GCN, include in the aggregation not only neighbouring entities, but also neighbouring relations, to make entity representations relation-aware.

Skip connection Stacking multiple GNN layers enables each entity to aggregate more information from further reaches of the graph. This, however, could also cause noisy information to propagate through layers. To mitigate this issue, some approaches, such as AliNet and RDGCN, use skip connections to bypass some layers and feed the output of one layer as the input to the next layers (instead of only the next layer). One commonly used skip connection method is concatenation, which is often accomplished by concatenating the outputs of all layers. As a result, final representations of entities involve their respective representations at all layers, instead of only the final layer. Another commonly used method is highway networks [42], which introduces gates at each layer and sums the output of a layer with its input with gating weights:

$$\begin{aligned} g(\mathbf{h}_{e_i}^{l}) &= \sigma(\mathbf{W}^{l}\mathbf{h}_{e_i}^{l} + \mathbf{b}^{l}) \\ \mathbf{h}_{e_i}^{l+1} &= g(\mathbf{h}_{e_i}^{l}) \cdot \mathbf{h}_{e_i}^{l+1} + (1 - g(\mathbf{h}_{e_i}^{l})) \cdot \mathbf{h}_{e_i}^{l} \end{aligned} \tag{6}$$

By using skip connections, entity representations are made more robust and more (neural network) structure-aware.

3.2 Alignment training

Given entity embeddings of two KGs, the training module aims to unify them into the same vector space so that aligned entities can be identified. As shown in Table 1, most approaches are supervised, that is, they rely on the supervision provided by pre-aligned entities. In supervised approaches (e.g., GCN-Align and GMNN), pre-aligned entities are used as labelled data to guide the training process, which pulls aligned entities close in the space. As pre-aligned entities are often limited, some approaches (e.g., MRAEA and EVA) also explore unlabelled data in training. These approaches are referred to as semi-supervised approaches. A common strategy is to iteratively label likely entity pairs from the alignment results generated by the inference module as the training data. The decision on which entity pairs are considered likely differs. In MRAEA, RREA, and Dual-AMN, two entities in the results are newly aligned if and only if they are mutual nearest neighbours. In EVA and MCLEA, a similar decision is made but enities are required to remain mutual nearest neighbours after a probation phase.

With pre-aligned or newly aligned entities as seed alignments, embeddings of entities are trained by minimising a loss function. The most commonly used loss function is the triplet loss:

$$L = \sum_{(e_i,e_j)\in\mathcal{S}} \sum_{(e'_i,e'_j)\in\mathcal{S}'} max(0, d(e_i, e_j) - d(e'_i, e'_j) + \gamma) \tag{7}$$

where $\mathcal{S}$ is the set of positive pairs (seed alignments), $\mathcal{S}'$ is the set of negative pairs, $d(\cdot)$ is a distance function (e.g., Manhattan distance) and $\gamma > 0$ is a margin hyper-parameter. Negative pairs are obtained by corrupting positive pairs, i.e., replacing entities in positive pairs with negative samples. Two strategies are generally used for generating negative samples: uniform sampling where negative samples are randomly selected from all entities, and nearest sampling where negative samples are selected from the positive sample's nearest neighbours. With the triplet loss, positive pairs are expected to have smaller distances than negative pairs, and also, a margin is expected to exist between the distances of positive and negative pairs.

Several other loss functions are also used for (semi-)supervised training. AliNet uses the contrastive alignment loss instead of the triplet loss to ensure that positive pairs have absolutely small distances. GM-Align formulates entity alignment as a graph matching problem and uses the cross entropy (CE) loss to maximise the matching probability of seed alignments. Dual-AMN uses the normalised hard sample mining(NHSM) loss to tackle the inefficiency issue in nearest sampling and leverages the LogSumExp operation [43] for generating high-quality negative samples. EVA employs a Neighbourhood Component Analysis (NCA) [44] based loss to mitigate the hubness problem in the embedding space. MCLEA uses intra-modal contrastive loss (ICL) and inter-modal algnment loss (IAL) for modelling both intra-modal and inter-modal interactions.

In addition to supervised and semi-supervised approaches, there are unsupervised approaches that do not require labelled entity pairs to align entities. SelfKG and ICLEA are two such approaches. While SelfKG focuses only on pushing negative pairs away than pulling positive pairs close, ICLEA emphasises both with the support of cross KG interaction through pseudo-aligned entity pairs. Both approaches adapt the noise contrastive estimation (NCS) loss for self-supervised settings and sample negative pairs from the same KGs (called self-negative sampling). Note that while certain approaches like MRAEA and EVA claim to support unsupervised training, they actually dependent on preprocessing to create initial alignments based on similarities of entity names or images. Since their embedding modules still require supervision, we classify them as (semi-)supervised in this paper.

3.3 Alignment inference

Given entity embeddings in the same space, the inference module aims to find alignments between two KGs. Without loss of generality, we refer to one KG as the source and the other as the target, and the inference module is to determine the most likely target entity for each source entity. The most common strategy used for inference is nearest neighbor (NN) search. For each source entity, NN search calculates the entity's distances to all target entities and then chooses the nearest target entity as the alignment. Commonly used distance measures include the Mahatten distance (L_1), the Euclidean distance (L_2) and the cosine similarity. Some approaches, e.g., AliNet and RREA, additionally employ cross-domain similarity local scaling (CSLS) [45] as an improved measure, to normalise the distance

between a source entity and a target entity based on the density of their neighbours. Suppose the cosine similarity is used, we have:

$$CSLS(e_i, e_j) = 2\,cos(e_i, e_j) - \frac{1}{m} \sum_{e'_i \in N(e_i)} cos(e_i, e'_i) - \frac{1}{m} \sum_{e'_j \in N(e_j)} cos(e'_j, e_j) \quad (8)$$

where $N(e_i)$ is the set of m nearest neighbors of e_i in the embedding space. As GM-Align aims to solve a graph matching problem for entity alignment, the matching probability is used as the distance measure, and the target entity with the highest matching probability is chosen as the alignment.

However, NN search fails to consider the interdependency between different alignment decisions. As such, a source entity may be aligned to a target entity that is more likely to be the alignment of another source entity based on their distance. To address this, CEA and RAGA formulate alignment inference as the stable matching (SM) problem and solve it by using the deferred acceptance algorithm [46]: the input of the algorithm is a matrix where rows represent source entities, columns represent target entities and entries represent preferences calculated based on a distance measure, and the output is a set of alignments where no pairs of entities prefer each other than their current aligned ones. RNM instead explores the interactions between entity alignments and relation alignments and employs an iterative matching (IM) strategy for inference, which iteratively updates the distance between two entities based on the mapping properties of the connected relations.

4 Discussion

Comparing approaches in their entirety is a common practice, but it can pose challenges in achieving a fair and meaningful evaluation of their performance. One significant factor is the diversity in the types of graph information used as input features within their embedding modules. As shown in Table 1, some approaches solely consider topological connections and relation types from relation triples, while others explore additional information, such as attributes or relation names. Incorporating more information during the embedding process can potentially improve entity representations, leading to better alignment results.

Furthermore, the methods used in embedding, training, and inference modules can vary across approaches, even when the input graph information remains the same. These methods are not necessarily specific to any one approach and can be applied universally. For example, instead of initialising entity embeddings randomly, the embedding module might use word embeddings derived from entity names for more informed initialisation. Training strategies can range from unsupervised to supervised, depending on the availability of labelled data. Training can also be conducted in a single pass or through iterative processes. Additionally, the inference module, which operates independently from embedding and training modules, may offer options for employing different distance metrics and search strategies while keeping the embedding and alignment modules unchanged. Each decision made regarding these modules can significantly impact the alignment results.

While existing studies offer insights into the overall effectiveness of entity alignment approaches through direct comparisons on benchmark datasets, a comprehensive understanding of their strengths and weaknesses necessitates examining individual components.

Specifically, it is crucial to investigate how the methods employed within each component influence the overall performance. By dissecting and evaluating these components individually, we can gain unique insights into their contributions, fostering opportunities for innovation and optimisation. Although there are ablation studies for individual approaches, they tend to focus only on the methods employed within each specific approach and lack a systematic analysis that goes beyond these methods. A comprehensive analysis would not only explore the methods within each approach but also consider alternative methods that have the potential to enhance the overall performance. This level of analysis will offer flexibility and adaptability to researchers and practitioners. By experimenting with different combinations of methods and components, they can tailor their approach to the specific needs and characteristics of the datasets they are working with. This adaptability enables the exploration of various techniques, leading to a better understanding of their impact on the overall performance. It promotes the discovery of novel combinations and fine-tuned strategies, enhancing the effectiveness and efficiency of entity alignment approaches.

However, conducting such an analysis for each approach, let alone comparing between approaches to identify specific components or methods contributing to superior performance, would be infeasible. Nevertheless, it is possible to focus on representative methods within each component and evaluate the effects of individual methods and potential combinations, as demonstrated in the next section.

5 Comparative analysis

5.1 Experiment settings

Datasets We conduct our analysis using two representative datasets: DBP15K [32] and SRPRS [65]. DBP15K is a widely used dataset for entity alignment, consisting of three subsets sampled from DBPedia: $\text{DBP}_{\text{ZH-EN}}$ (Chinese-English), $\text{DBP}_{\text{JA-EN}}$ (Japanese-English) and $\text{DBP}_{\text{FR-EN}}$ (French-English). Each subset contains 15,000 pre-aligned entity pairs, which are used for training and testing. SRPRS is another dataset sampled from DBPedia and Wikidata. Compared to DBP15K, SRPRS is sparser and has much fewer relations and triples. We specifically use two cross-lingual subsets of SRPRS: $\text{SRPRS}_{\text{EN-FR}}$ (English-French) and $\text{SRPRS}_{\text{EN-DE}}$ (English-German). Similar to DBP15K, each subset of SRPRS also contains 15,000 pre-aligned enity pairs. Table 2 provides the statistics of these datasets, where 'avg. deg.' denotes the average number of relation triples in which an entity is involved. Following the conventions of existing studies, in our experiments, we use 30% of the pre-aligned entity pairs for training and 70% of them for testing.

Evaluation metrics We report our results using standard evaluation metrics, specifically $H@k$ (where $k = 1, 10$). The $H@k$ metric measures the percentage of correctly aligned entities among the top-k nearest target entities, with $H@1$ representing the accuracy of alignment results. Higher $H@k$ values indicate better performance. Additionally, we employ the mean reciprocal rank (MRR), which evaluates alignment results by averaging the reciprocal ranks of correctly aligned entities. Both the $H@k$ and MRR metrics assess alignment quality by considering the position or rank of correct matches. Due to space constraints, we omit the MRR results in this paper. To ensure reliable and robust measurements, we report the performance based on the average of five independent runs.

Table 2 Dataset statistics

Dataset	KG	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	$\|\mathcal{T}^r\|$	avg. deg.
$\text{DBP}_{\text{ZH-EN}}$	Chinese	19,388	1,701	70,414	7.25
	English	19,572	1,323	95,142	9.71
$\text{DBP}_{\text{JA-EN}}$	Japanese	19,814	1,299	77,214	7.79
	English	19,780	1,153	93,484	9.44
$\text{DBP}_{\text{FR-EN}}$	French	19,661	903	105,998	10.77
	English	19,993	1,208	115,722	11.56
$\text{SRPRS}_{\text{EN-DE}}$	English	15,000	222	38,363	5.11
	German	15,000	120	37,377	4.98
$\text{SRPRS}_{\text{EN-FR}}$	English	15,000	221	36,508	4.86
	French	15,000	177	33,532	4.47

Methodologies and implementation datails To gain insights into the impact of individual components on alignment performance, we conduct component-level comparisons. In these comparisons, we vary one component while keeping the other components fixed. To represent each module, we select representative methods and organise our experiments accordingly. In addition, we assess the performance of selected combinations of components. Throughout the evaluation process, we fix the neighbourhood aggregation methods to be GCN or GAT, conducting the same sets of experiments for each method. This allows us to observe how changes in one or more components affect the performance of these two basic GNN models for entity alignment. To maintain consistency with existing approaches, we fix the loss function to be the triplet loss, which is commonly employed by various alignment methods. Table 3 presents the evaluated representative methods, with the default choices being underlined. These choices provide a solid foundation for our evaluation, allowing us to examine and compare the performance of different combinations in a systematic manner.

We adopt a typical network configuration for entity alignment, consisting of 2 layers with a dimension of 300 for each layer. We employ the Adam optimiser [66] and train our models for up to 2000 epochs. Negative samples are updated every 10 epochs. For each combination of neighbourhood aggregation and skip connection options, we tune the following parameters to find their optimal values: the learning rate in {0.0005, 0.001, 0.005, 0.01}, the margin for the triplet loss in {1.0, 2.0, 3.0, 4.0}, the dropout rate in {0.1, 0.2, 0.3, 0.4}, the number of negative samples in{15, 20, 25, 30, 35} and the number of attention heads (GAT) in {1, 2}.

For entity initialisation, we use Glorot initialisation [67] to generate random embeddings for entities. Alternatively, when entities are initialised with names, we utilise pre-trained fasttext embeddings [68, 69] as name embeddings. Following typical implementations, for DBP15K, we employ Google Translate to translate entity names to English and then use the pretrained wiki word vectors[1] to derive embeddings; for SRPRS, we directly use entity names without translation and derive the embeddings via aligned word vectors[2]. For semi-supervision, we implement the bi-direction iterative method [52] and set the maximum iteration number to 3. Nearest sampling is limited to the training data, while uniform sampling involves selecting samples from all entities. To facilitate stable matching, we utilise the

[1] https://fasttext.cc/docs/en/pretrained-vectors.html

[2] http://fasttext.cc/docs/en/aligned-vectors.html

Table 3 Component-level experiments: options and default choices (underlined)

Structure embedding	**Entity initialisation**: random, entity name embedding
	Neighbourhood aggregation: GCN, GAT
	Skip connection: none, concatenation, highway gates
Alignment training	**Training strategy**: supervision, semi-supervision
	Cost function: triplet loss
	Negative sampling: uniform, nearest
Alignment inference	**Inference strategy**: nearest neighbour(NN), stable matching(SM)
	Distance Measure: Manhattan(L1), Euclidean(L2), cosine
	CSLS: no, yes

deferred acceptance algorithm [46]. When employing the CSLS method, we fix the number of nearest neighbors, denoted as 'm' in the CSLS computation (Equation 8), to 1. We find that larger values do not significantly improve performance. All experiments are conducted on a workstation with 2 Intel(R) Xeon(R) Gold 5118 CPUs, 128GB memory and a Nvidia Quadro P5000 GPU. The code and parameter settings are available online[3].

5.2 Results and analyses

Experiment 1: effect of structure embedding options. Table 4 shows the performance of different entity initialisation, neighbourhood aggregation, and skip connection strategies. On the DBP15K dataset, initialising entity embeddings with name features leads to a significant improvement compared to randomly initialised embeddings. The gain is more pronounced when skip connections are also used. Take GCN for example, using name initialisation alone leads to a gain of about 9%-13% in $H@1$ compared to random initialisation, while combining name initialisation and highway gates results in an even greater gain of about 27%-42% in $H@1$. This enhancement can be attributed to two factors. First, name initialisation allows the network to capture additional information about entities beyond the graph structure. Second, skip connections ensure that the network can effectively extract relevant information from name embeddings, while ignoring any noisy or irrelevant signals. However, we notice that using skip connections with randomly initialised entity embeddings usually leads to ineffective results. This ineffectiveness likely arises from the network's difficulty in discerning meaningful patterns amid the noise present in random initialisations. Skip connections, in such cases, introduce unnecessary complexities, potentially hindering the embedding performance. We also find that, in general, the utilisation of highway gates leads to superior performance compared to concatenation. As for the effect of neighbourhood aggregation options, the difference in performance between GCN and GAT on DBP15K is not apparent.

[3] https://github.com/YF-SHU/EvalFramework

Table 4 Experimental results on the effect of structure embedding options

Method	$\text{DBP}_{\text{ZH-EN}}$		$\text{DBP}_{\text{JA-EN}}$		$\text{DBP}_{\text{FR-EN}}$		$\text{SRPRS}_{\text{EN-DE}}$		$\text{SRPRS}_{\text{EN-FR}}$	
	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10
With random initialisation										
GCN	**45.47**	**81.28**	**48.16**	**83.18**	47.49	**84.13**	**42.66**	**69.08**	**27.80**	**59.74**
GCN+Highway	43.23	77.35	46.80	80.27	**48.00**	80.43	41.02	68.24	26.59	59.32
GCN+Concat	41.15	74.15	43.76	76.18	44.91	77.83	31.05	61.25	22.53	52.57
GAT	44.56	**80.54**	46.10	**82.16**	45.80	**83.39**	**42.03**	**68.48**	**26.84**	58.56
GAT+Highway	**45.18**	78.63	**48.11**	81.45	**49.17**	82.18	40.76	68.45	26.13	**58.79**
GAT+Concat	42.84	76.28	44.42	78.59	44.57	79.80	35.10	60.72	22.39	50.49
With name initialisation										
GCN	56.51	88.65	57.68	90.36	60.91	90.42	50.28	76.83	32.50	67.07
GCN+Highway	**72.96**	**89.01**	**79.75**	**93.40**	**90.55**	**97.33**	**61.20**	**84.51**	**47.05**	**79.11**
GCN+Concat	70.21	83.25	76.38	88.15	89.49	95.74	40.11	61.31	38.74	56.04
GAT	55.18	86.96	56.23	89.23	62.59	91.35	38.78	68.48	24.12	56.12
GAT+Highway	72.28	**89.13**	**78.54**	**92.78**	**90.81**	**97.44**	**53.08**	**81.09**	38.96	**71.17**
GAT+Concat	**72.39**	85.82	78.53	89.99	90.66	96.82	50.82	78.51	**39.65**	64.09

On the SRPRS dataset, the overall performance is worse than on DBP15K, as entities in SRPRS are involved in fewer relations, resulting in the network captures less contextual information. However, we consistently observe that combining name intialisation with highway gates achieves better performance than using name initialisation alone, and using highway gates generally outperforms concatenation. Furthermore, a notable difference in performance between GAT and GCN networks on SRPRS arises when using name features without skip connections. GAT performs much worse than GCN, even worse than GAT with random initialisation. We attribute this to GAT's heightened sensitivity to the noise introduced by name embeddings. Unlike GCN, which performs neighborhood aggregation based on node degrees, GAT aggregates information using attentional weights computed through similarity computations of embeddings. This reliance on attentional weights makes GAT more susceptible to the noise present in name embeddings. The reason we do not observe the same performance degradation on DBP15K is that entity names are translated to English first, which effectively reduces noise in the embeddings.

To sum up, the effect of structure embedding options is influenced by the degree distribution of datasets. Our experiments consistently demonstrate that the results on the denser DBP15K dataset outperform those on SRPRS. Furthermore, initialising entity embeddings with name features generally enhances performance; however, its effectiveness depends on the network's ability to handle noise introduced by name embeddings, and incorporating highway gates effectively reduces noise within the network. Combining name initialisation with highway gates consistently leads to significant performance improvements on both DBP15K and SRPRS datasets. Conversely, when entities are randomly initialised, the use of skip connections tends to be ineffective.

Experiment 2: effect of training options Table 5 presents the results of different negative sampling and training strategies. As shown in the table, under supervised training, using nearest sampling achieves better performance than using uniform sampling. Figure 2 further illustrates the training epochs required for the GCN network to converge on both $\texttt{DBP}_{\text{ZH-EN}}$ and $\texttt{SRPRS}_{\text{EN-DE}}$ datasets. Clearly, the network with uniform sampling takes much longer to converge compared to nearest sampling: with nearest sampling, only about 500 epochs are needed, while with uniform sampling, the network has not yet converged even at 1500 epochs. Similar results are observed for the GAT network and other datasets.

As the training shifts from supervised to semi-supervised by labelling likely aligned entity pairs as new training data, $H@1$ consistently improves on both GCN and GAT networks, regardless of the negative sampling strategy used. Figure 3 shows the precision and recall of the semi-supervision strategy when used with the GCN network on both $\texttt{DBP}_{\text{ZH-EN}}$ and $\texttt{SRPRS}_{\text{EN-DE}}$ datasets. Here, precision and recall respectively denote the percentages of truely aligned pairs discovered over the total number of discovered pairs and over the total number of truely aligned pairs. With more iteration rounds, the precision decreases, while the recall increases, indicating that more errorneous pairs are included in the training over time. Similar trends are observed for the GAT network and other datasets. The inclusion of errorneous pairs potentially explains the degradation in $H@10$ of the GAT network as it is more sensitive to noise in the network.

Comparing the results between DBP15K and SRPRS, we observe that the effect of different negative sampling or training strategies on performance is more apparent on DBP15K than on SRPRS. For instance, the GCN network achieves a gain of about 8% in $H@1$ with semi-supervision on DBP15K compared to supervision, while there is only about 3% improvement on SRPRS, when nearest sampling is used. This illustrates that the degree distribution of

Table 5 Experimental results on the effect of training options

Method	DBP_{ZH-EN}		DBP_{JA-EN}		DBP_{FR-EN}		$SRPRS_{EN-DE}$		$SRPRS_{EN-FR}$	
	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10
With uniform sampling										
GCN+superv.	40.31	77.71	41.64	78.96	40.26	79.86	40.00	67.31	25.61	57.65
GCN+semi	**49.11**	**82.93**	**51.20**	**83.76**	**50.12**	**85.19**	**44.18**	**69.71**	**29.35**	**59.73**
GAT+superv.	38.50	76.43	40.31	78.41	37.07	76.79	40.20	67.23	25.61	**57.46**
GAT+semi	**45.26**	**79.11**	**48.51**	**81.15**	**44.30**	**80.36**	**44.31**	**67.46**	**29.34**	56.49
With nearest sampling										
GCN+superv.	45.47	81.28	48.16	83.18	47.49	84.13	42.66	69.08	27.80	59.74
GCN+semi	**53.76**	**84.86**	**56.32**	**85.86**	**55.96**	**87.59**	**45.87**	**70.43**	**30.98**	**61.24**
GAT+superv.	44.56	80.54	46.10	**82.16**	45.80	**83.39**	42.03	**68.48**	26.84	**58.56**
GAT+semi	**51.54**	**80.85**	**52.74**	80.99	**51.98**	80.11	**45.84**	66.81	**30.00**	57.03

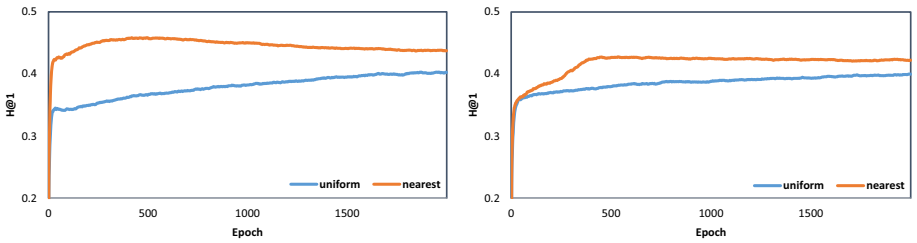

Figure 2 Covergence of uniform and nearest sampling strategies

datasets also affects the quality of samples and aligned pairs discovered, leading to variations in the impact of different strategies on different datasets.

Overall, the effect of training options is influenced by the degree distribution of datasets. Our findings consistently demonstrate that different training options yield superior results on the DBP15K dataset compared to the SRPRS dataset. Additionally, we find that the network using uniform sampling exhibits slower convergence and produces inferior results compared to the network utilising nearest sampling. This highlights that the quality of negative samples significantly affects training efficiency and effectiveness. Moreover, the utilisation of semi-supervision enhances alignment performance, particularly in terms of $H@1$. However, it is important to note that the quality of the chosen strategy plays a crucial role in the overall performance of semi-supervised learning.

Experiment 3: effect of inference options Table 6 presents the results of different distance metrics and inference strategies. Among L1, L2, and cosine similarity, no single metric clearly outperforms the others across all networks or datasets. However, combining these metrics with CSLS in nearest neighbor (NN) search consistently yields improved results. CSLS enhances these metrics by normalising the distance between two entities based on the density of their neighbours in the embedding space. Entities that frequently appear as nearest neighbors of others receive more significant distance penalisation. Notably, the improvement on DBP15K is more noticeable than on SRPRS. For example, using cosine similarity with CSLS on DBP15K leads to about 5% improvement in $H@1$, while on SRPRS, only about 2% improvement is achieved. The relative ineffectivess of CSLS on SRPRS is mainly due to sparse KGs having fewer hub entities (entities that appear more than once as nearest neighbours) in the vector space compared to dense KGs when considering only the structural information, as is the case here. This is confirmed by Figure 4 which displays proportions

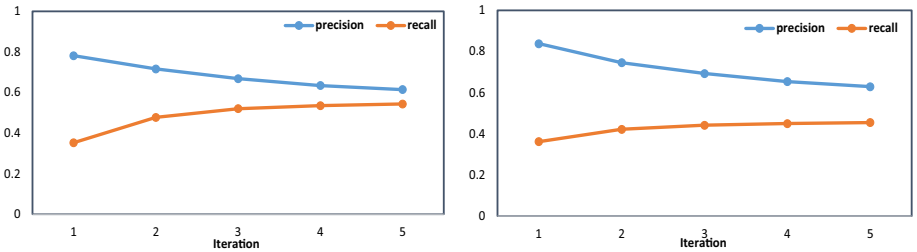

Figure 3 Precision and recall of the semi-supervised strategy employed

Table 6 Experimental results on the effect of inference strategies

Method	DBP_{ZH-EN}		DBP_{JA-EN}		DBP_{FR-EN}		$SRPRS_{EN-DE}$		$SRPRS_{EN-FR}$	
	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10
With L1										
GCN+NN	46.36	79.49	48.70	81.30	46.60	81.78	44.91	69.61	29.91	60.28
GCN+NN,CSLS	51.51	**83.11**	52.99	**84.63**	51.37	**85.42**	46.91	**70.84**	31.90	**61.72**
GCN+SM	51.82		53.47		51.52		**48.13**		32.24	
GCN+SM,CSLS	**52.24**		**53.62**		**51.79**		47.94		**32.87**	
GAT+NN	44.10	77.88	45.78	80.82	44.82	80.58	42.58	67.40	26.71	57.07
GAT+NN,CSLS	49.11	**82.23**	51.23	**84.40**	49.29	**84.27**	45.21	**69.41**	29.72	**59.43**
GAT+SM	50.24		51.84		49.49		**48.99**		**33.77**	
GAT+SM,CSLS	**50.42**		**52.06**		**49.80**		47.81		32.41	
With L2										
GCN+NN	46.57	79.91	49.13	81.73	48.33	83.00	45.11	69.42	30.06	60.61
GCN+NN,CSLS	52,02	**83.69**	53.67	**85.06**	53.11	**86.91**	47.03	**70.94**	32.18	**62.26**
GCN+SM	52.94		54.49		53.34		**48.50**		32.75	
GCN+SM,CSLS	**53.06**		**54.99**		**53.49**		48.05		**33.21**	
GAT+NN	44.14	78.25	45.88	80.65	44.88	80.71	42.53	66.99	26.71	56.39
GAT+NN,CSLS	49.51	**82.72**	51.24	**84.60**	49.58	**85.12**	45.45	**69.49**	29.53	**59.03**
GAT+SM	51.52		52.40		50.04		**50.97**		**34.99**	
GAT+SM,CSLS	**52.51**		**52.60**		**50.52**		48.79		33.89	

Table 6 continued

Method	DBP_{ZH-EN}		DBP_{JA-EN}		DBP_{FR-EN}		$SRPRS_{EN-DE}$		$SRPRS_{EN-FR}$	
	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10
With cosine										
GCN+NN	45.47	81.28	48.16	83.18	47.49	84.13	42.66	69.08	27.80	59.74
GCN+NN,CSLS	50.31	**85.23**	53.17	**85.71**	52.71	**87.30**	44.63	**69.98**	29.91	**61.15**
GCN+SM	51.97		53.04		53.39		47.26		31.05	
GCN+SM,CSLS	**53.04**		**55.10**		**54.26**		**47.63**		**31.72**	
GAT+NN	44.56	80.54	46.10	82.16	45.80	83.39	42.03	68.48	26.84	58.56
GAT+NN,CSLS	50.08	**83.42**	52.00	**85.08**	51.30	**86.23**	44.75	**69.54**	29.35	**59.72**
GAT+SM	51.50		53.92		52.73		**48.80**		32.54	
GAT+SM,CSLS	**52.33**		**54.22**		**53.36**		48.65		**33.56**	

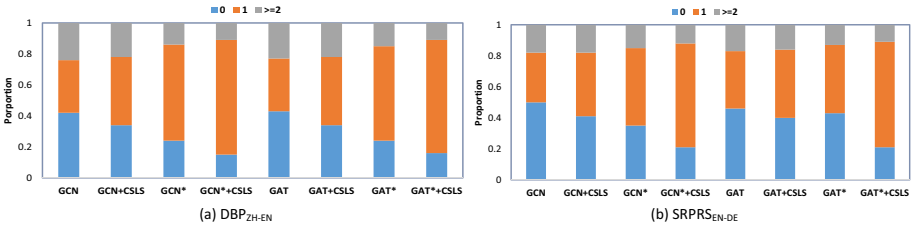

Figure 4 Proportions of target entities that appear 0, 1 and more times as nearest neighbours (GCN*, GAT* represent GCN and GAT networks with entity name initialisation and highway gates in Experiment 4)

of target entities that appear 0, 1 and more times as nearest neighbours on $\text{DBP}_{\text{ZH-EN}}$ and $\text{SRPRS}_{\text{EN-DE}}$ datasets (other datasets exhibit similar results).

Furthermore, using stable matching instead of NN search further improves $H@1$. However, we do not observe significant improvement when CSLS is also used. Additionally, the improvement achieved on DBP15K and on SRPRS with stable matching compared to NN search is similar. This suggests that stable matching is less affected by the choice of distance metric and the degree distribution of datasets. It is important to note that while stable matching enhances $H@1$, it comes at the cost of significantly increased running time compared to NN search. For instance, on DBP15K, without CSLS being used, NN search takes about 10s to produce results, whereas stable matching requires about 27s.

Experiment 4: effect of selected combinations Finally, we compare the overall performance of two groups of combinations. In the first group, we assume no name information is available, and entities are randomly initialised without skip connections. In the second group, we assume entity names are available, and entities are initialised with name embeddings, while highway gates are used. For both groups, we employ nearest sampling and cosine similarity-based NN search, and combine these strategies with CSLS or semi-supervision. The results are shown in Table 7. As with previous observations, incorporating name initialisation and skip connections (highway gates) significantly improves the performance of GCN and GAT networks compared to random initialisation and no skip connections. Using CSLS or semi-supervision further enhances $H@1$, and the combination of both brings the most substantial improvement. This is because the entity pairs labelled as new training data in each iteration are more accurate due to the use of CSLS. Interestingly, on SRPRS, the effect of using CSLS on performance is more evident for the second group than for the first group. Figure 4 shows that when entity names are considered, the proportions of target entities appearing only once as nearest neighbours of source entities increase significantly, indicating better entity embeddings are learned. By using CSLS, the proportions of both isolated and hub entities further decrease, and this decrease is more significant than when no name information is considered. Moreover, on SRPRS, while there is a substantial performance gap between GCN and GAT networks when no CSLS or semi-supervision is used for the second group, the use of these two strategies bridges the gap.

It is noteworthy that the performance of these two groups of combinations, namely the combination of random enity initialisation, CSLS, and semi-supervision, and the combination of name initialisation, highway gates, CSLS, and semi-supervision, is comparable or even superior to many existing approaches that incorporate more graph information as input or employ more complicated embedding methods. This demonstrates that by selecting suitable methods for combination, even basic GNN networks can achieve competitive results.

Table 7 Experimental results of selected combinations

Method	DBP_{ZH-EN}		DBP_{JA-EN}		DBP_{FR-EN}		$SRPRS_{EN-DE}$		$SRPRS_{EN-FR}$	
	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10	H@1	H@10
With random initialisation, no skip connection										
GCN	45.47	81.28	48.16	83.18	47.49	84.13	42.66	69.08	27.80	59.74
GCN+semi	53.76	84.86	56.32	85.86	55.96	87.59	45.87	70.43	30.98	61.24
GCN+CSLS	50.31	85.23	53.17	85.71	52.71	87.30	44.63	69.98	29.91	61.15
GCN+CSLS+semi	**55.66**	**86.33**	**58.06**	**87.32**	**58.51**	**89.74**	**46.94**	**70.41**	**32.37**	**61.97**
GAT	44.56	80.54	46.10	82.16	45.80	83.39	42.03	68.48	26.84	58.56
GAT+semi	51.54	80.85	52.74	80.99	51.98	80.11	45.84	66.81	30.00	57.03
GAT+CSLS	50.08	**83.42**	52.00	**85.08**	51.30	**86.23**	44.75	**69.54**	29.35	**59.72**
GAT+CSLS+semi	**53.98**	82.40	**56.36**	83.60	**54.50**	82.69	**46.80**	67.20	**31.32**	57.29
With name initialisation, highway gates										
GCN	72.96	89.01	79.75	93.40	90.55	97.33	61.20	84.51	47.05	79.11
GCN+semi	81.69	93.45	87.33	96.57	94.72	98.68	74.07	88.02	59.06	82.52
GCN+CSLS	80.73	92.99	87.41	96.83	93.89	97.97	74.91	89.50	63.69	84.26
GCN+CSLS+semi	**85.93**	**96.31**	**91.16**	**98.41**	**96.11**	**99.52**	**81.14**	**91.85**	**69.12**	**85.08**
GAT	72.28	89.13	78.54	92.78	90.81	97.44	53.08	81.09	38.96	71.17
GAT+semi	78.44	89.92	84.57	93.76	94.15	98.02	69.66	86.86	50.16	77.10
GAT+CSLS	79.84	92.83	86.36	96.35	94.06	98.93	76.27	93.12	59.04	83.65
GAT+CSLS+semi	**83.38**	**94.16**	**89.71**	**97.51**	**96.00**	**99.45**	**83.21**	**94.91**	**68.61**	**88.32**

6 Conclusion

This paper delves into the critical role of entity alignment in knowledge graph (KG) integration, focusing specifically on exploring Graph Neural Network (GNN)-based approaches. Our investigation has led us to develop a framework that captures the essential features of existing GNN-based entity alignment methods. Through a detailed analysis, we have shed light on the significant impact that individual components and methods have on performance, highlighting specific module options that notably influence alignment results. Additionally, we have learned that the degree distribution of the dataset plays a pivotal role in shaping alignment outcomes.

Our research has shown that by carefully selecting suitable methods for combination, competitive results can be achieved even with basic GNN networks. However, it's important to note that our analysis has limitations. We have not fully explored the impact of various graph information types beyond graph structures and entity names. Our experiments have revealed a performance gap between dense and sparse datasets. Recent advancements, such as incorporating multi-modal information [56] or exploring associations between attributes and relations [70, 71] for long-tail entity alignment, present opportunities to address this challenge. Furthermore, we have yet to explore the impact of self-supervised training strategies on performance and their applicability. Despite these limitations, our work lays the foundation for tailored approaches considering specific needs and dataset characteristics. Researchers can drive the field forward by experimenting with diverse combinations of components and methods, advancing the state-of-the-art in entity alignment, and enhancing knowledge graph integration techniques. These efforts will open new possibilities for leveraging knowledge graphs across diverse applications.

Author Contributions Yanfeng Shu wrote the main manuscript text. All authors reviewed the manuscript.

Funding Open access funding provided by CSIRO Library Services. The work was supported by CSIRO, Australia.

Availability of data and materials Yes

Code Availibility Yes.

Declarations

Competing interests The authors declare that they have no competing interests.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Yes.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: ISWC (2007)
2. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A core of semantic knowledge unifying wordnet and wikipedia. In: WWW (2007)
3. Walsh, B., Mohamed, S.K., Nováček, V.: Biokg: A knowledge graph for relational learning on biological data. In: CIKM (2020)
4. Haussmann, S., Seneviratne, O., Chen, Y., Ne'eman, Y., Codella, J., Chen, C.-H., McGuinness, D.L., Zaki, M.J.: Foodkg: A semantics-driven knowledge graph for food recommendation. In: ISWC (2019)
5. J.Zhou, Cui, G., Zhange, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open. **1**, 57–81 (2020)
6. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2021)
7. Morris, C., Ritzert, M., Fey, M., Hamilton, W.L., Lenssen, J.E., Rattan, G., Grohe, M.: Weisfeiler and leman go neural: Higher-order graph neural networks. In: AAAI (2019)
8. Sun, Z., Zhang, Q., Wu, W., Wang, C., Chen, M., Akrami, F., Li, C.: A benchmarking study of embedding-based entity alignment for knowledge graphs. VLDB. **13**(12), 2326–2340 (2020)
9. Zeng, K., Li, C., Hou, L., Li, J., Feng, L.: A comprehensive survey of entity alignment for knowledge graphs. AI Open. **2**, 1–13 (2021)
10. Zhang, Z., Chen, J., Chen, X., Liu, H., Xiang, Y., Liu, B., Zheng, Y.: An industry evaluation of embedding-based entity alignment. In: COLING (2020)
11. Zhao, X., Zeng, W., Tang, J., Wang, W., Suchanek, F.M.: An experimental study of state-of-the-art entity alignment approaches. IEEE Trans. Knowl. Data Eng. **34**(6), 2610–2625 (2022)
12. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech and time series. In: The Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, MA, USA (1998)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
14. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2009)
15. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
16. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
17. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. In: ICLR (2016)
18. Ganea, O.-E., Hofmann, T.: Deep joint entity disambiguation with local neural attention. In: EMNLP (2017)
19. Le, P., Titov, I.: Improving entity linking by modeling latent relations between mentions. In: ACL (2018)
20. Zhou, X., Miao, Y., Wang, W., Qin, J.: A recurrent model for collective entity linking with adaptive features. In: AAAI (2020)
21. Barlaug, N., Gulla, J.A.: Neural networks for entity matching: A survey. ACM Trans. Knowl. Discov. Data **15**(3), 1–37 (2021)
22. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. **24**(9), 1537–1555 (2011)
23. Fu, C., Han, X., Sun, L., Chen, B., Zhang, W., Wu, S., Kong, H.: End-to-end multi-perspective matching for entity resolution. In: IJCAI (2019)
24. Koudus, N., Sarawag, S., Srivastava, D.: Record linkage: Similarity measures and algorithms. In: SIGMOD (2006)
25. Jiménez-Ruiz, E., Grau, B.C.: Logmap: Logic-based and scalable ontology matching. In: ISWC (2011)
26. Megdiche, I., Teste, O., Trojahn, C.: An extensible linear approach for holistic ontology matching. In: ISWC (2016)
27. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1), 158–176 (2013)
28. Lacoste-Julien, S., Palla, K., Davies, A., Kasneci, G., Graepel, T., Ghahramani, Z.: Sigma: Simple greedy matching for aligning large knowledge bases. In: SIGKDD (2013)
29. Niu, X., Rong, S., Wang, H., Yu, Y.: An effective rule miner for instance matching in a web of data. In: CIKM (2012)
30. Suchanek, F.M., Abiteboul, S., Senellart, P.: Paris: Probabilistic alignment of relations, instances, and schema. In: VLDB (2012)
31. Chen, M., Tian, Y., Yang, M., Zaniolo, C.: Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In: IJCAI (2016)

32. Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: ISWC (2017)
33. Sun, Z., Hu, W., Zhang, Q., Qu, Y.: Bootstrapping entity alignment with knowledge graph embedding. In: IJCAI (2018)
34. Sun, Z., Huang, J., Hu, W., Chen, M., Guo, L., Qu, Y.: Transedge: Translating relation-contextualized embeddings for knowledge graphs. In: ISWC (2019)
35. Zhu, H., Xie, R., Liu, Z., Sun, M.: Iterative entity alignment via joint knowledge embeddingss. In: IJCAI (2017)
36. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS (2013)
37. Cao, Y., Liu, Z., Li, C., Liu, Z., Li, J., Chua, T.-S.: Multi-channel graph neural network for entity alignment. In: ACL (2019)
38. Sun, Z., Wang, C., Hu, W., Chen, M., Dai, J., Zhang, W., Qu, Y.: Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In: AAAI (2020)
39. Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convoluational networks. In: EMNLP (2018)
40. Wu, Y., Liu, X., Feng, Y., Wang, Z., Zhao, D.: Jointly learning entity and relation representations for entity alignment. In: EMNLP (2019)
41. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS (2017)
42. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway netowrks. arXiv:1505.00387 (2015)
43. Sun, Y., Cheng, C., Zhang, Y., Zhang, C., Zheng, L., Wang, Z., Wei, Y.: Circle loss: A unified perspective of pair similarity optimization. In: CVPR (2020)
44. Goldberger, J., Hinton, G.E., Roweis, S.T., Salakhutdinov, R.R.: Neighbourhood components analysis. In: NeurIPS (2005)
45. Conneau, A., Lample, G., Ranzato, M., Denoyer, L., Jégou, H.: Word translation without parallel data. In: ICLR (2018)
46. Roth, A.E.: Deferred acceptance algorithms: History, theory, practice and open questions. Int. J. Game Theory **36**(3), 537–569 (2008)
47. Ye, R., Li, X., Fang, Y., Zang, H., Wang, M.: A vectorized relational graph convolutional network for multi-relational network alignmnet. In: IJCAI (2019)
48. Yang, H.-W., Zou, Y., Shi, P., Lu, W., Lin, J., Xun, X.: Aligning cross-lingual entities with multi-aspect information. In: EMNLP (2019)
49. Li, C., Cao, Y., Hou, L., Shi, J., Li, J., Chua, T.-S.: Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model. In: EMNLP (2019)
50. Zeng, W., Zhao, X., Tang, J., Lin, X.: Collective entity alignment via adaptive features. In: ICDE (2020)
51. Liu, Z., Cao, Y., Pan, L., Li, J., Liu, Z., Chua, T.-S.: Exploring and evaluating attributes, values, and structures for entity alignment. In: EMNLP (2020)
52. Mao, X., Wang, W., Xu, H., Lan, M., Wu, Y.: Mraea: An efficient and robust entity alignment approach for cross-lingual knowledge graph. In: WSDM (2020)
53. Mao, X., Wang, W., Xu, H., Wu, Y., Lan, M.: Relational reflection entity alignment. In: CIKM (2020)
54. Mao, X., Wang, W., Wu, Y., Lan, M.: Boosting the speed of entity alignment 10×: Dural attention matching network with normalized hard sample mining. In: WWW (2021)
55. Yu, D., Yang, Y., Zhang, R., Wu, Y.: Knowledge embedding based graph convolutional network. In: WWW (2021)
56. Liu, F., Chen, M., Roth, D., Collier, N.: Visual pivoting for (unsupervised) entity alignment. In: AAAI (2021)
57. Lin, Z., Zhang, Z., Wang, M., Shi, Y., Wu, X., Zheng, Y.: Multi-modal contrastive representation learning for entity alignment. In: COLING (2022)
58. Xu, K., Wang, L., Yu, M., Feng, Y., Song, Y., Wang, Z., Yu, D.: Cross-lingual knowledge graphs alignment via graph matching neural network. In: ACL (2019)
59. Wu, Y., Liu, X., Feng, Y., Wang, Z., Yan, R., Zhao, D.: Relation-aware entity alignment for heterogeneous knowledge graphs. In: IJCAI (2019)
60. Wu, Y., Liu, X., Feng, Y., Wang, Z., Zhao, D.: Neighborhood matching nework for entigy alignment. In: ACL (2020)
61. Zhu, Y., Liu, H., Wu, Z., Du, Y.: Relation-aware neighborhood matching model for entity alignment. In: AAAI (2021)
62. Zhu, R., Ma, M., Wang, P.: Raga: Relation-aware graph attention networks for global entity alignment. In: PAKDD (2021)
63. Liu, X., Hong, H., Wang, X., Chen, Z., Kharlamov, E., Dong, Y., Tang, J.: Selfkg: Self-supervised entity alignment in knowledge graphs. In: WWW (2022)

64. Zeng, K., Dong, Z., Hou, L., Cao, Y., Hu, M., Yu, J., Lv, X., Li, J., Feng, L.: Iclea: Interactive contrastive learning for self-supervised entity alignment. In: ACL (2022)
65. Guo, L., Sun, Z., Hu, W.: Learning to exploit long-term relational dependencies in knowledge graphs. In: ICML (2019)
66. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: ICLR (2015)
67. Guo, L., Sun, Z., Hu, W.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS (2010)
68. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. In: ACL (2017)
69. Joulin, A., Bojanowski, P., Mikolov, T., Jégou, H., Grave, E.: Loss in translation: Learning bilingual word mapping with a retrieval criterion. In: EMNLP (2018)
70. Sun, Z., Hu, W., Wang, C., Wang, Y., Qu, Y.: Revisiting embedding-based entity alignment: A robust and adaptive method. IEEE Trans. Knowl. Data Eng. **35**(8), 8461–8475 (2023)
71. Zhong, Z., Zhang, M., Fan, J., Dou, C.: Semantics driven embedding learning for effective entity alignment. In: ICDE (2022)