Dynamic Entity Alignment with Attribute Integration

Anonymous ACL submission

Abstract

Entity alignment is a key technology for integrating knowledge graphs (KG). However, existing methods assume KG is static and overlook the fact that KG evolves over time. With the growth of KG, previous alignment results may require adjustments, and new equivalent entities need to be found for the newly added entities. Additionally, new entities often have fewer neighbors, which adds difficulty to their alignment. In this paper, we propose DEA-AttrAlign to address these challenges. The core idea is to quickly generate representations for entities based on their neighborhoods. In cases where neighborhood information for new entities is lacking, we propose utilizing entity attributes and semantic facts derived from triplets as additional alignment information. Extensive experiments show that our approach is more effective than methods based on retraining or inductive learning.

1 Introduction

Entity alignment(EA) is the task of finding identical entities across different knowledge graphs(KGs). It can facilitate the sharing and transfer of knowledge from multiple sources, providing better support for downstream applications such as question-answering systems(Yu et al., 2017), and search engine optimization(Xiong et al., 2017). Early research in entity alignment mainly relied on string matching to calculate entity similarity and used similarity propagation for inference during entity alignment(Melnik et al., 2002; Suchanek et al., 2011). The development of deep learning techniques has led to recent work representing entities as embeddings and using nearest neighbor search in vector space for entity alignment(Chen et al., 2017). Typically, embedded-based entity alignment models employ shallow or deep neural networks, such as TransE(Bordes et al., 2013) or Graph Convolutional Networks (GCN)(Kipf and Welling, 2017), to encode entities and utilize the vector representations to compute similarity between entities. These methods have achieved good alignment performance in the alignment reasoning stage, driving current research in entity alignment.

However, most current entity alignment methods assume that the knowledge graphs are static(Yan et al., 2021), ignoring the fact that KG grows over time. For example, the system of DBpedia(Lehmann et al., 2015) extracts approximately 21 billion new triples each month(Hofer et al., 2020), introducing new entities and providing clues for correcting previous alignments. However, most existing methods can only adapt to changes in the KG by retraining models on the grown KG. This will undoubtedly result in significant cost consumption.

In this paper, we focus on the problem of alignment after the growth of KG(referred to as dynamic KG entity alignment, Figure 1 provides an example of dynamic KG entity alignment.) which presents the following challenges:

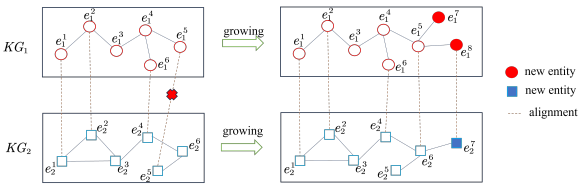

Figure 1: Explanation of dynamic KG entity alignment. Solid icons represent newly added entities. Dotted lines represent equivalent entities. The task of dynamic KG alignment is to find equivalent entities for the newly added ones and correct misalignments from the previous stage.

- When the KG grows, it inevitably introduces the problem of aligning new enti-

ties. The cost of retraining the model from scratch is high. Therefore, a key challenge is how to obtain embeddings for new entities quickly, accurately, and with minimal cost. ContEA(Wang et al., 2022) first uses the entity reconstruction function to infer the embeddings of new entities based on their neighborhoods. Then fine-tune the KG encoder to obtain more accurate embeddings. However, its entity reconstruction function only performs mean pooling on the neighbors surrounding the entity, without considering the varying influence of neighbors connected by different relations on the entity.

- The neighborhood information surrounding new entities is often insufficient, leading to the long-tail entity problem. This poses significant challenges for GCN-based EA methods. MCEA(Qi et al., 2022) proposes to enhance the embedding representation of long-tail entities by using second-order neighborhood embeddings during the aggregation of neighbors. This inevitably introduces some noise. It has been proven that attribute information plays a positive role in the entity alignment process and complements the KG structure(Zeng et al., 2019; Cheng et al., 2022). However, currently, no work has utilized attributes in dynamic entity alignment.

In response to these challenges, we propose DEA-AttrAlign, a framework for Dynamic Entity Alignment with Attribute Integration. For the challenge of obtaining embeddings for new entities: when a new entity is introduced, we utilize its neighborhood information to generate embeddings. Specifically, we don't simply average its neighbors, but instead use attention-based weighted averaging to address the varying influence of different neighbors. Then the encoder is fine-tuned to generate embedding representations that are more sensible.

To tackle the challenge of the long-tail entity problem, we propose integrating knowledge facts to enrich information by preserving the original triplet semantics for new entities with insufficient neighborhood information. In previous research on static entity alignment, attribute information has been proven to enhance performance(Yang et al., 2019; Lai et al., 2020; Shi et al., 2023). Therefore, we are attempting for the first time to utilize the attribute information to achieve better performance in dynamic EA.

Extensive experiments have been conducted, and our method has achieved optimal performance whether in the base KG or after KG growth.

2 Preliminary

2.1 Entity Alignment

Given two KGs $\mathcal{G}_1 = \{\mathcal{E}_1, \mathcal{R}_1, \mathcal{A}_1, \mathcal{T}_1^{\mathcal{R}}, \mathcal{T}_2^{\mathcal{A}}\}$ and $\mathcal{G}_2 = \{\mathcal{E}_2, \mathcal{R}_2, \mathcal{A}_2, \mathcal{T}_2^{\mathcal{R}}, \mathcal{T}_2^{\mathcal{A}}\}$, where $\mathcal{E}, \mathcal{R}, \mathcal{A}$ represent sets of entities, relations, and attributes respectively. $\mathcal{T}_{\mathcal{R}} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of relation triplets. $\mathcal{T}_{\mathcal{A}} \subset \mathcal{E} \times \mathcal{A} \times \mathcal{V}$ is the set of attribute triplets. Entity alignment aims to find the alignment set $\mathcal{A} = \{(e_1, e_2) \in \mathcal{E}_1 \times \mathcal{E}_2 \mid e_1 \equiv e_2\}$, where "$\equiv$" represents equivalence. In most cases, a small set of seed alignments $\mathcal{A}_{train}$ is provided as training data, and the task is to find the remaining alignments $\mathcal{A}_{res} = \{(e_1', e_2') \in \mathcal{E}_1' \times \mathcal{E}_2' \mid e_1' \equiv e_2'\}$, where e_1' and e_2' represent the sets of entities to be aligned in the two KGs.

2.2 Dynamic KG Entity Alignment:

The dynamic KG $\mathcal{G}$ is defined as a series of snapshots $\mathcal{G} = (\mathcal{G}^0, \mathcal{G}^1, ..., \mathcal{G}^T)$, where the superscript number represents different timestamps. For any two consecutive timestamps $\mathcal{G}^t = \{\mathcal{E}^t, \mathcal{R}^t, \mathcal{T}_{\mathcal{R}}^t, \mathcal{T}_{\mathcal{A}}^t\}$ and $\mathcal{G}^{t+1} = \{\mathcal{E}^{t+1}, \mathcal{R}^{t+1}, \mathcal{T}_{\mathcal{R}}^{t+1}, \mathcal{T}_{\mathcal{A}}^{t+1}\}$, it holds that $\mathcal{E}^t \subseteq \mathcal{E}^{t+1}$, $\mathcal{R}^t = \mathcal{R}^{t+1}$, $\mathcal{T}_{\mathcal{R}}^t \subseteq \mathcal{T}_{\mathcal{R}}^{t+1}$, and $\mathcal{T}_{\mathcal{A}}^t \subseteq \mathcal{T}_{\mathcal{A}}^{t+1}$.

Given two growing KGs $\mathcal{G}_1$ and $\mathcal{G}_2$, and a seed entity alignment $\mathcal{A}_s$ at time $t = 0$. The goal of dynamic entity alignment is to predict new alignments and revise old alignments $\mathcal{A}_{new}^t = \{(e_1^t, e_2^t) \in \mathcal{E}_1^t \times \mathcal{E}_2^t \mid e_1^t \equiv e_2^t\}$ based on the current learned embeddings and alignment model at time t. Considering that acquiring seed entities is challenging, we do not assume that the growth of KGs will bring new training data, meaning that $\mathcal{A}_s$ remains the same at $t = 0$ and $t > 0$.

3 Method

In this section, we will introduce our dynamic entity alignment framework: DEA-AttrAlign. Figure 2 depicts its overall process. It mainly

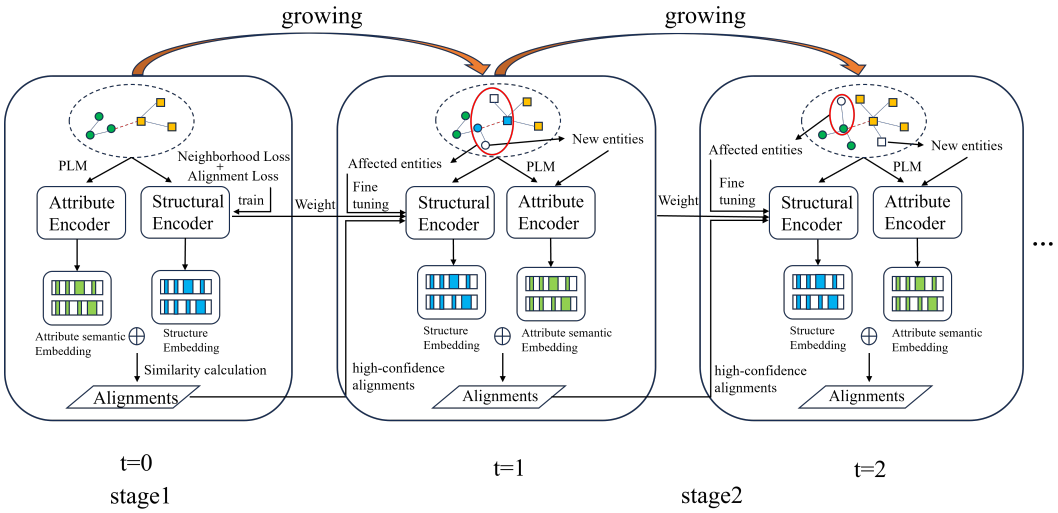

Figure 2: Overview of the proposed method DEA-AttrAlign

consists of two stages: the base KG alignment stage and the growth KG alignment stage.

3.1 Basic KG alignment stage

At this stage, we first represent entities as embedding vectors by capturing information from both the structure and attributes of the entities. Specifically, we train a GNN-based structural encoder to capture structural information and we utilize pre-trained language models to capture textual information associated with entity attributes. Subsequently, we combine specific similarity measures to find alignment results.

3.1.1 Structural Encoder

We adopt a GNN-based Dual-AMN(Mao et al., 2021) as the structural encoder to capture the structural information. It includes two modules, which capture structure information at different levels. It consists of an inner-graph module ($Aggregator_{inner}$) that captures structural information within a single KG and a cross-graph module ($Aggregator_{cross}$) that captures cross-graph matching information. The process can be expressed using the following formula.

$$\boldsymbol{h}_{\boldsymbol{e}_{\boldsymbol{inner}}} = Aggregator_{inner}(\sum_{e_j \in \mathcal{N}_{e_i}} \alpha_{ij}\mathbf{e}_j)) \quad (1)$$

$$\boldsymbol{h}_{\boldsymbol{e}_{\boldsymbol{struct}}} = Aggregator_{cross}(h_{e_{inner}}, \mathcal{E}_{proxy}) \quad (2)$$

Where, $\boldsymbol{h}_{\boldsymbol{e}_{\boldsymbol{struct}}}$ represents the structural information embedding of entity e. $\mathcal{N}_{e_i}$ represents the neighbors of entity e. $\mathcal{E}_{proxy}$ represents the proxy vector.

To train the Structural Encoder, we utilize two losses: alignment loss and neighborhood loss. The alignment loss adopts the loss proposed by Dual-AMN. The neighborhood loss, designed to address the issue of KG expansion, aims to quickly and reasonably generate initial embeddings for new entities.

Alignment Loss Given the embedding of the structural encoder, the objective of alignment learning is to minimize the distance between equivalent entities (positive samples) and maximize the distance between non-equivalent entities (negative samples). We adopted the loss function used in Dual-AMN as the alignment loss.

$$\mathcal{L}_{\text{align}} = \log\left[1 + \sum_{i \in \mathcal{A}_s} \sum_{j \in \mathcal{A}_{e_1}^{\text{neg}}} \exp\left(\gamma\left(\lambda + s_i - s_j\right)\right)\right] \quad (3)$$

$$s_j = \text{sim}\left(e_1, e_2'\right) - \text{sim}\left(e_1', e_2\right) \quad (4)$$

$$s_i = \text{sim}\left(e_1, e_2\right) \quad (5)$$

Where A_S is the seed alignment pair, e_1', e_2' are the negative samples. The specific sampling strategy is that for e_1, all other entities in the same training batch are used as negative samples. The negative samples for e_2 are obtained in the same way. γ is a scale factor, and λ is the margin for separating the similarities

of seed alignment pairs and negative pairs. The similarity calculation uses cosine similarity.

Neighborhood Loss As the KG grows, a well-trained GNN structural encoder may encounter new entities. The significant challenge is how to incorporate the newly added entities into the encoder. ContEA(Wang et al., 2022) adopts averaging neighbors to initialize embeddings for new entities. However, this approach overlooks the varying impact of different neighbors on the new entity. Therefore, we utilize an attention mechanism to represent the new entity through its neighbors, thus emphasizing the distinct roles of different neighbors. For this purpose, we have designed a neighborhood loss function:

$$\mathcal{L}_{\text{nei}} = \sum_{e_i \in \mathcal{E}} \left\| \mathbf{e_i} - \sum_{e_j \in \mathcal{N}_{e_i}} \alpha_{ij} \mathbf{e_j} \right\|_2^2 \tag{6}$$

Where, N_{e_i} represents the set of one-hop neighbors of entity e_i. The calculation of α_{ij} is as follows:

$$\alpha_{ij} = \frac{\exp\left(\boldsymbol{v}^T \boldsymbol{h}_{\boldsymbol{r_k}}\right)}{\sum_{e'_j \in \mathcal{N}_{e_i}} \sum_{r_{k'} \in \mathcal{R}_{ij'}} \exp\left(\boldsymbol{v}^T \boldsymbol{h}_{r_{k'}}\right)} \tag{7}$$

Where, $\boldsymbol{v}^T$ is an attention vector, $\boldsymbol{h}_{\boldsymbol{r_k}}$ represents the embedding vector of the relationship r_k between entities e_i and e_j.

Adding the neighborhood loss and alignment loss with weights gives the final loss:

$$\mathcal{L}_{base} = \mathcal{L}_{align} + \beta \cdot \mathcal{L}_{nei} \tag{8}$$

3.1.2 Attribute Semantic Encoder

The structural encoder can effectively capture the structural information of KGs. Previous studies have shown that structure and textual representations are fundamentally two complementary views of entity alignment. Therefore, We also introduce attribute features in dynamic EA. Firstly, we collected the attribute information of entities from the DBP15K dataset. As KG grows, we use entity names to query related attributes and attribute values in Dbpedia. Then we used LaBSE(Feng et al., 2022) as the text encoder to capture entity semantic information. The specific approach is as follows:

For entity names, we directly feed them into the encoder to form embeddings. For entity attribute information, we concatenate all attribute triples related to e into a single string, denoted as e_{attr} (in the form of $a_1 v_1 a_2 v_2 ...$). The order of attribute triples depends on their frequency in the knowledge graph. Finally, we merge the entity's structural and semantic information to form the final embedding representation:

$$\boldsymbol{h}_{\boldsymbol{e_i}} = \boldsymbol{h}_{\boldsymbol{e_{struct}}} \oplus \boldsymbol{h}_{\boldsymbol{e_{attr}}} \tag{9}$$

where, $\boldsymbol{h}_{\boldsymbol{e_{attr}}}$ represents the attribute information embedding of entity e using LaBSE. $\oplus$ denotes the concatenation operation. Entity name embedding is used as the initialization of the structural encoder.

3.1.3 Bidirectional Alignment Selection

After passing through two encoders, we obtain the joint embedding representation of the entities. Then, we use CSLS to compute the similarity matrix and select alignment results. To address the one-to-many issue inherent in nearest neighbor search, we implement bidirectional nearest neighbor selection. This method ensures that (e_1, e_2) is considered equivalent only if e_1 's nearest neighbor in KG_2 is e_2, and e_2 's nearest neighbor in KG_1 is e_1.

3.2 Growth KG Alignment Stage

Figure 3 illustrates the main process of the growth KG alignment stage. When $t > 0$, the structure of KGs changes with the emergence of new triples.

The task in this stage is to capture the structural changes while generating embeddings for new entities. To tackle this challenge, we first generate embeddings for the newly added entities using their neighborhoods with attention mechanisms and relevant fact triples. Then, we fine-tune the structural encoder. We also captured the semantic information brought by the attributes of the new entities. After fine-tuning, we combine the structural and semantic embeddings to form the final embeddings which are used for new alignment predictions. Heuristic strategies are employed to merge the new predicted alignments with the old alignments discovered at $t-1$.

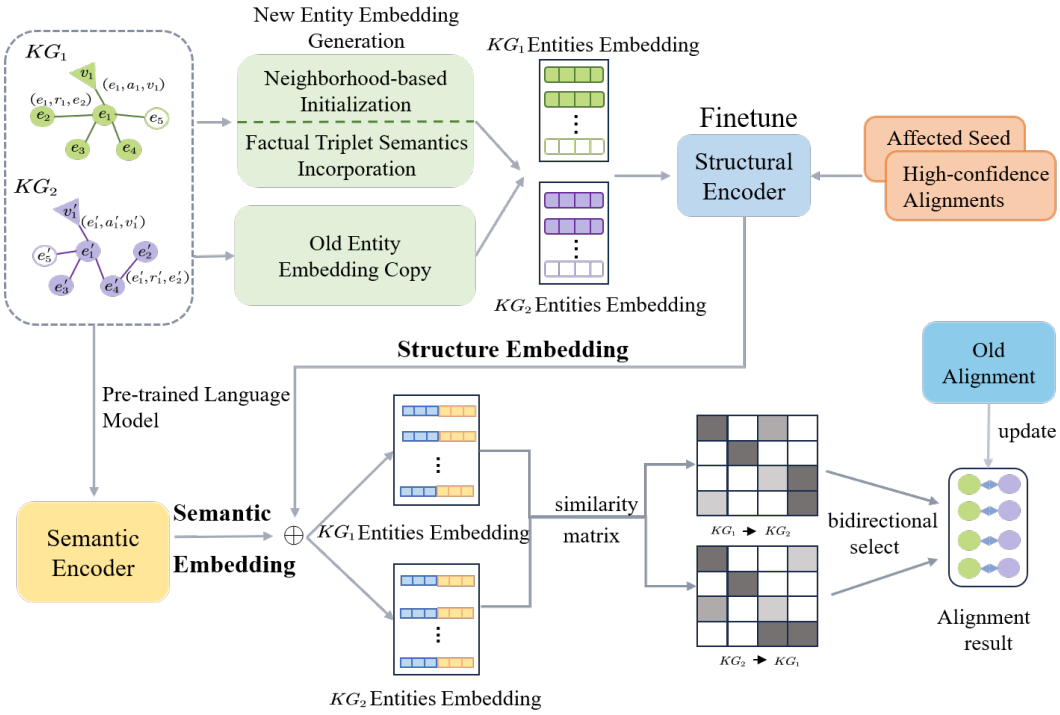

Figure 3: Growth KG alignment stage

3.2.1 New Entity Embedding Generation

The embedding generation of the new entity is divided into two steps: neighborhood-based initialization and factual triplet semantics incorporation.

Neighborhood-based Initialization

Thanks to the optimization objective of neighborhood loss, we can represent a new entity using its neighborhood. The specific approach is shown in Equations 1 and 2.

Factual Triplet Semantics Incorporation

However, the neighbors of new entities are often sparse, resulting in a long-tail entity problem. Inspired by KAGNN(Huang et al., 2022), we used an approach that combines the factual triplets of new entities to enhance the embedding representation of new entities. More details can be found in Appendix A.1.

3.2.2 Fine-tuning Structural Encoder

The new entity embedding representation obtained through the aforementioned process is not adequate. In addition, as new entities appear, the embedding representations of entities connected to the new entities will also be affected and need to be updated. To address the above issues, we choose to fine-tune the structural encoder. Specifically, according to Wang et al. (2022), we only update the parameters of the second module of the structural encoder, named the cross-layer aggregator $Aggregator_{cross}$.

For the training data for fine-tuning, since alignment mostly occurs around anchor entities, we only select anchor entities that appear in new triplets (referred to as affected entities) as seed alignments. However, the neighbors of new entities are often sparse, so there are not many anchor entities. It is far from sufficient to fine-tune the model with only these data. Therefore, combining the results of the basic KG alignment stage, we select m pairs of highly confident alignment results by setting a threshold, thereby providing more comprehensive data for fine-tuning. We use two parts of data for fine-tuning. The loss function for fine-tuning is:

$$\begin{aligned}\mathcal{L}_{update} =& \mathcal{L}_{\text{align}}\left(High-Confidence\right) \\ &+ \alpha \cdot \mathcal{L}_{\text{align}}\left(Affectd\right) + \beta \cdot \mathcal{L}_{\text{nei}}\end{aligned} \tag{10}$$

The alignment loss of affected entity pairs, denoted as $\mathcal{L}_{\text{align}}\left(Affectd\right)$, and the alignment loss of high-confidence entity pairs, denoted

as $\mathcal{L}_{\text{align}}$ ($High-Confidence$), are calculated. The fine-tuning loss function is then obtained by taking a weighted sum of these three losses. α and β are hyper parameters used to balance their importance.

3.2.3 Alignment updates

After obtaining the joint embeddings for the structure and attributes of the new entities, we can compute a similarity matrix for the entities, which allows us to make a new set of alignment predictions. Inevitably, conflicts may arise between these new alignment predictions and those generated in the previous step. To address this, we use a simple yet effective solution. For a given prediction (e_1, e_2), when both e_1 and e_2 are new entities, we keep it as an accurate alignment prediction. However, when either e_1 or e_2 is an old entity, conflicts occur between the alignment predictions and the ones given in the previous step. Because e_1 or e_2 may align with a new entity in the current stage, while e_1 or e_2 has already aligned with the old entity in the previous stage. In this case, we determine which pair of predictions to keep based on the principle of similarity, retaining the pair with higher similarity.

4 Experiment

4.1 Datasets

Wang et al. (2022) propose a dataset for dynamic KG entity alignment, which is constructed based on the widely used DBP15K dataset. Specifically, it includes 5 snapshots to simulate the growth of KGs. The detailed statistical information of the dataset is shown in Appendix B.1.

4.2 Baselines

We compared our model with the following methods and conducted experiments on the datasets.

4.2.1 Retraining baselines

Since most existing embedding-based EA methods are designed for static KGs, they need to be retrained whenever new triplets appear.

Here, we select the representative translation-based method MTransE(Chen et al., 2016), as well as several GNN-based methods, including GCN-Align(Wang et al., 2018), KEGCN(Yu et al., 2021), and Dual-AMN(Mao et al., 2021) as our baselines.

4.2.2 Inductive baselines

There are limited entity alignment methods focusing on KG growth, such as DINGAL-O(Yan et al., 2021) and ContEA(Wang et al., 2022). Additionally, since there are some inductive Knowledge Graph Embedding (KGE) methods that can generate embeddings for new entities, we selected two representative inductive KGE methods, MEAN(Hamaguchi et al., 2017) and LAN(Wang et al., 2020), as the entity representation layer, and integrated them with Dual-AMN, referring to Wang et al. (2022). We denote them as MEAN+ and LAN+ for our two baselines.

4.3 Experiment settings

4.3.1 Evaluation metrics

Dao et al. (2023) noted that rank-based metrics like Mean Reciprocal Rank (MRR) and Hits@K, though common in information retrieval, are unsuitable for EA matching tasks since only Hits@1 aligns with the recall of greedy matching algorithms. Therefore, we follow their approach and use precision, recall, and F1 score for evaluation. All values are presented as percentages from 0 to 100, with an F1 score of 100 indicating a perfect match.

4.4 Results

4.4.1 Main results

We conducted experiments on the dataset and the experimental results are shown in Table 1. To further validate the impact of attributes on dynamic entity alignment, we apply our attribute fusion method to five GCN-based baseline models. The experimental results are shown in Appendix B.3

Overall performance Our model achieved optimal performance across all snapshots. Compared to the best baseline, ContEA, our model's F1 scores improved significantly on various datasets. For the DBP_{ZH-EN} snapshots, the F1 scores increased by 9.2% ($t = 0$), 11.9% ($t = 1$), 10% ($t = 2$) and 8.9% ($t = 3$). Similarly, for DBP_{JA-EN}, the improvements were 6.9%, 7%, 5.9% and 4% across the respective snapshots. On the DBP_{FR-EN} snapshots, our

			t = 0			t = 1			t = 2			t = 3		
			P	R	F1	P	R	F1	P	R	F1	P	R	F1
DBP_{ZH-EN}	Retraining	MTransE	55.2	17.8	26.9	24.2	11.1	15.2	15.9	7.8	10.5	9.4	5.4	6.8
		GCN-Align	56.5	21.2	30.8	27.5	17.0	21.0	17.3	14.0	15.5	12.3	11.7	12.0
		KEGCN	56.9	22.1	31.9	33.8	18.0	23.5	27.3	25.3	19.6	23.0	12.8	16.5
		Dual-AMN	83.3	60.3	69.9	50.6	51.4	51.0	38.5	45.6	41.7	28.5	28.6	28.6
	Induct	MEAN$^+$	82.8	57.6	67.9	48.3	42.2	45.0	35.7	34.1	34.9	26.7	26.4	26.5
		LAN$^+$ *	82.7	57.6	67.9	48.8	42.6	45.5	36.0	34.5	35.2	27.4	27.1	27.2
		DINGAL-O	49.7	19.5	28.0	37.0	15.8	22.2	31.5	13.5	18.9	25.1	11.1	15.4
		ContEA	84.5	61.1	70.9	54.1	53.1	53.6	43.1	46.8	44.9	36.4	41.6	38.8
	DEA-AttrAlign		**91.1**	**71.5**	**80.1**	**66.4**	**64.7**	**65.5**	**52.4**	**57.5**	**54.9**	**45.1**	**50.6**	**47.7**
DBP_{JA-EN}	Retraining	MTransE	59.9	20.0	29.9	29.3	12.1	17.2	21.3	8.2	11.8	15.1	6.1	8.7
		GCN-Align	62.1	24.4	35.0	31.9	19.3	24.1	21.6	15.7	18.2	16.2	12.9	14.4
		KEGCN	58.4	22.7	32.7	36.2	18.2	24.3	29.3	14.9	19.8	25.7	12.4	16.7
		Dual-AMN	84.4	59.0	69.5	54.4	50.6	52.5	42.2	43.0	42.7	34.8	29.2	31.8
	Induct	MEAN$^+$	84.7	57.1	68.2	52.8	42.0	46.8	40.7	33.0	36.5	33.0	26.1	29.2
		LAN$^+$	84.5	57.5	68.4	52.8	42.4	47.0	41.0	33.3	36.8	33.5	26.5	29.2
		DINGAL-O	54.0	22.7	32.0	39.1	17.4	24.1	32.8	13.7	19.4	27.1	11.3	15.9
		ContEA	85.5	60.5	70.9	57.7	52.2	54.8	46.5	44.3	45.4	39.8	37.9	38.9
	DEA-AttrAlign		**90.2**	**68.5**	**77.8**	**62.4**	**61.3**	**61.8**	**50.1**	**52.6**	**51.3**	**41.7**	**44.2**	**42.9**
DBP_{FR-EN}	Retraining	MTransE	57.0	18.8	28.3	24.6	10.0	14.2	14.5	6.2	8.7	10.8	4.0	5.9
		GCN-Align	58.3	22.3	32.3	28.5	17.2	21.5	18.4	13.1	15.3	13.8	10.3	11.8
		KEGCN	58.7	23.3	33.4	35.0	17.8	23.6	27.4	13.3	17.9	23.2	10.4	14.4
		Dual-AMN	85.1	61.8	71.6	52.1	49.8	51.0	40.7	40.4	40.6	35.1	27.3	30.7
	Induct	MEAN$^+$	84.0	58.5	69.0	51.4	41.5	45.9	38.7	30.5	34.1	31.4	23.5	26.9
		LAN$^+$	84.5	59.4	69.7	50.6	41.0	45.3	37.9	30.0	33.5	30.4	22.7	26.0
		DINGAL-O	54.0	22.4	31.7	38.1	16.5	23.1	32.9	12.4	18.0	25.8	9.2	13.6
		ContEA	86.8	63.5	73.3	56.6	52.4	54.4	45.2	42.6	43.8	38.7	35.4	37.0
	DEA-AttrAlign		**91.8**	**75.4**	**82.8**	**59.2**	**64.2**	**61.6**	**46.6**	**51.3**	**48.8**	**39.0**	**41.5**	**40.2**

Table 1: Main results of entity alignment on three datasets

model achieved gains of 9.5%, 7.2%, 5%, and 3.2% at the corresponding time points.

Compared with retraining baselines Compared to the best retraining baseline: Dual-AMN, our model's F1 scores show significant improvements across all datasets and time points. For DBP_{ZH-EN}, the F1 scores increase by 10.2%, 14.5%, 13.2%, and 19.1% at $t = 0$, $t = 1$, $t = 2$, and $t = 3$, respectively. In the DBP_{JA-EN} dataset, the scores increase by 8.3%, 9.3%, 8.6%, and 11.1% at the same time points. For DBP_{FR-EN}, the respective increases are 11.2%, 10.6%, 8.2%, and 9.5% at each time point. DEA-AttrAlign achieves good performance by considering the entire process as continuous, effectively utilizing the outputs of previous stages (such as alignment prediction results and model parameters) and integrating their alignment predictions through heuristic strategies(Wang et al., 2022).

Compared with inductive baselines As for the inductive baselines, DINGAL-O, a model specifically designed for dynamic entity alignment tasks, does not perform well either as it does not make any adjustments to the model parameters and simply inducts new entities based on existing model parameters, which is unreasonable. Although ContEA achieves good results compared to other inference models, it only relies on neighborhoods for new entity generation, and we know that the neighborhoods of new entities are often very sparse. Our method not only utilizes neighborhood information with attention mechanisms to generate embeddings for new entities but also incorporates textual information derived from the semantic facts of new entities and the attribute triples associated with them. Various types of information are used to make the embeddings of new entities more reasonable, leading to performance improvements. Furthermore, due to the incorporation of entity attribute triples, our method also achieves performance improvements of 9.2% on DBP_{ZH-EN}, 6.9% on DBP_{JA-EN}, 9.5% on DBP_{FR-EN} during the basic KG alignment stage ($t = 0$).

Additionally, it can be observed that the performance of all models decreases as the graph grows. This is because the search space of candidate entity pairs expand, the ratio of seed alignment to entities to be aligned decreases and the number of entities to be aligned becomes very large, resulting in difficulties in correctly matching entities.

		t = 0				t = 1				t = 2				t = 3			
		P	R	F1		P	R	F1		P	R	F1		P	R	F1	
ZH − EN	**DEA-AttrAlign**	**91.1**	**71.5**	**80.1**		**66.4**	**64.7**	**65.5**		**52.4**	**57.5**	**54.9**		**45.1**	**50.6**	**47.7**	
	w/o Nei	80.3	53.6	64.3	15.8↓	52.8	45.1	48.7	16.8↓	39.3	38.1	38.7	16.2↓	32.2	32.4	32.3	15.4↓
	w/o Attr	87.3	65.7	75.0	5.1↓	60.4	57.8	59.1	6.4↓	49.1	50.5	49.8	5.1↓	42.1	43.8	42.9	4.8↓
	w/o Attr&Name	83.7	60.1	69.9	10.2↓	57.2	52.4	54.7	10.8↓	46.5	46.5	46.5	8.4↓	38.5	40.6	39.5	8.2↓
	w/o Fact	-	-	-	-	60.9	56.4	58.6	6.9↓	47.7	50.3	49.0	5.9↓	39.4	44.5	41.8	5.9↓
JA − EN	**DEA-AttrAlign**	**90.2**	**68.5**	**77.8**		**62.4**	**61.3**	**61.8**		**50.1**	**52.6**	**51.3**		**41.7**	**44.2**	**42.9**	
	w/o Nei	82.1	55.2	66.0	11.8↓	53.6	44.3	48.5	13.3↓	41.3	35.3	38.0	13.3↓	33.0	28.1	30.4	12.5↓
	w/o Attr	88.6	66.1	75.7	2.1↓	61.3	55.9	58.5	3.3↓	50.4	46.5	48.4	2.9↓	43.7	38.7	41.0	1.9↓
	w/o Attr&Name	85.2	60.4	70.7	7.1↓	59.6	51.8	55.4	6.4↓	49.4	43.9	46.5	4.8↓	43.2	36.9	39.8	3.1↓
	w/o Fact	-	-	-	-	58.8	54.5	56.6	5.2↓	46.4	47.1	46.7	4.6↓	38.7	40.2	39.5	3.4↓
FR − EN	**DEA-AttrAlign**	**91.8**	**75.4**	**82.8**		**59.2**	**64.2**	**61.6**		**46.6**	**51.3**	**48.8**		**39.0**	**41.5**	**40.2**	
	w/o Nei	83.4	58.0	68.5	14.3↓	53.3	44.4	48.5	13.1↓	39.3	32.8	35.8	13.0↓	30.7	25.0	27.5	12.7↓
	w/o Attr	85.7	62.1	72.0	10.8↓	55.5	49.7	52.5	9.1↓	46.0	39.6	42.6	6.2↓	41.3	31.8	35.9	4.3↓
	w/o Attr & Name	84.7	60.8	70.8	12.0↓	57.2	49.4	53.0	8.6↓	47.5	39.4	43.0	5.8↓	42.0	31.6	36.1	4.1↓
	w/o Fact	-	-	-	-	54.9	53.5	54.2	7.4↓	42.3	43.3	42.8	6.0↓	35.6	36.0	35.8	4.4↓

Table 2: Ablation study results on three datasets

4.4.2 Ablation study

We design four sets of ablation experiments. The experimental results are shown in Table 2.

DEA-AttrAlign w/o Nei: When training the structural encoder, neighborhood loss is not used.

DEA-AttrAlign w/o Attr: At all stages, for an entity e, we only use GCN to capture structural information.

DEA-AttrAlign w/o Attr&name: In the basic KG alignment stage, we do not use the textual representation vector of the entity name as initialization for entity e. Instead, we randomly initialize the embedding.

DEA-AttrAlign w/o Fact: In the growth KG alignment stage, we no longer use TransE to retain the semantic information of the factual triplets. We only rely on GCN to aggregate neighborhood information in generating embeddings.

After removing the neighborhood loss, there is a significant decrease in model performance by over 10% on each snapshot, which also proves the effectiveness of the neighborhood loss we designed. The structural encoder trained with the neighborhood loss enables us to initialize embeddings for new entities using their neighbors. The performance decreases when the attribute information is removed. On the ZH-EN dataset, performance decreased by up to 6.1% ($t = 1$); on the JA-EN dataset, performance decreased by up to 3.3% ($t = 1$); on the FR-EN dataset, performance decreased by up to 10.8% ($t = 0$) and there is even greater performance decrease when the entity name is also removed. On the ZH-EN dataset, performance decreased by up to 10.8% ($t = 1$); on the JA-EN dataset, performance decreased by up to 7.1% ($t = 0$); on the FR-EN dataset, performance decreased by up to 12.0% ($t = 0$). This also demonstrates the effectiveness of our method in utilizing both attribute information and entity names. The performance also decreases when the semantic information of the factual triplets is removed. On the ZH-EN dataset, performance decreased by up to 6.9% ($t = 1$); on the JA-EN dataset, performance decreased by up to 5.2% ($t = 1$); on the FR-EN dataset, performance decreased by up to 7.4% ($t = 1$), which further validates the effectiveness of this module.

5 Conclusion

In this paper, we propose a method called DEA-AttrAlign to address the more realistic scenario of entity alignment, which is dynamic entity alignment task. This method leverages entity neighborhood structure and fact triples to quickly generate embeddings for new entities. Additionally, we are the first to incorporate attribute information into the dynamic entity alignment task, using attributes to enhance alignment when entity neighborhood information is insufficient. Extensive experimental results demonstrate the effectiveness of our method.

6 Limitations

Although the effectiveness of our method, DEA-AttrAlign, has been demonstrated, there are still issues that need to be explored in the future: (1) The current datasets simulate KG growth by assuming only the appearance of new entities. More complex scenarios, such as

the addition or deletion of relationships, need to be explored in future research. (2) Our current work incorporates only two modalities, namely structure and text. Future work could investigate the impact of incorporating additional modalities, such as images, on dynamic entity alignment.

References

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2016. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *International Joint Conference on Artificial Intelligence*.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1511–1517.

Bo Cheng, Jia Zhu, and Meimei Guo. 2022. Multijaf: Multi-modal joint entity alignment framework for multi-modal knowledge graph. *Neurocomputing*, 500:581–591.

Nhat Minh Dao, Thai V. Hoang, and Zonghua Zhang. 2023. A benchmarking study of matching algorithms for knowledge graph entity alignment. *ArXiv*, abs/2308.03961.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 878–891, Dublin, Ireland. Association for Computational Linguistics.

Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1802–1808. AAAI Press.

Marvin Hofer, Sebastian Hellmann, Milan Dojchinovski, and Johannes Frey. 2020. The new dbpedia release cycle: Increasing agility and efficiency in knowledge extraction workflows. *Semantic Systems. In the Era of Knowledge Graphs*, 12378:1 – 18.

Zhichao Huang, Xutao Li, Yunming Ye, Baoquan Zhang, Guangning Xu, and Wensheng Gan. 2022. Multi-view knowledge graph fusion via knowledge-aware attentional graph neural network. *Applied Intelligence*, 53:3652–3671.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *Preprint*, arXiv:1609.02907.

Kwei-Herng Lai, Daochen Zha, Yuening Li, and Xia Hu. 2020. Bert-int: A bert-based interaction model for knowledge graph alignment. In *International Joint Conference on Artificial Intelligence*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, S. Auer, and Christian Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

Xixun Lin, Hong Yang, Jia Wu, Chuan Zhou, and Bin Wang. 2019. Guiding cross-lingual entity alignment via adversarial knowledge embedding. *2019 IEEE International Conference on Data Mining (ICDM)*, pages 429–438.

Xin Mao, Wenting Wang, Yuanbin Wu, and Man Lan. 2021. Boosting the speed of entity alignment 10 ×: Dual attention matching network with normalized hard sample mining. *Proceedings of the Web Conference 2021*.

Xin Mao, Wenting Wang, Huimin Xu, Man Lan, and Yuanbin Wu. 2020. Mraea: An efficient and robust entity alignment approach for cross-lingual knowledge graph. *Proceedings of the 13th International Conference on Web Search and Data Mining*.

S. Melnik, H. Garcia-Molina, and E. Rahm. 2002. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th International Conference on Data Engineering*, pages 117–128.

Shichao Pei, Lu Yu, and Xiangliang Zhang. 2019. Improving cross-lingual entity alignment via optimal transport. In *International Joint Conference on Artificial Intelligence*.

Donglin Qi, Shudong Chen, Xiao Sun, Ruipeng Luan, and Da Tong. 2022. A multiscale convolutional gragh network using only structural information for entity alignment. *Applied Intelligence*, 53:7455–7465.

Xinchen Shi, Bin Li, Ling Chen, and Chao Yang. 2023. Bi-neighborhood graph neural network for cross-lingual entity alignment. *Knowl. Based Syst.*, 277:110841.

Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. 2011. Paris: Probabilistic alignment of relations, instances, and schema. *Preprint*, arXiv:1111.7164.

Zequn Sun, Jiacheng Huang, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. 2019a. Transedge: Translating relation-contextualized embeddings for knowledge graphs. *ArXiv*, abs/2004.13579.

Zequn Sun, Chengming Wang, Wei Hu, Muhao Chen, Jian Dai, Wei Zhang, and Yuzhong Qu. 2019b. Knowledge graph alignment network with gated multi-hop neighborhood aggregation. In *AAAI Conference on Artificial Intelligence*.

Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2020. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. *Preprint*, arXiv:1811.01399.

Yuxin Wang, Yuanning Cui, Wenqiang Liu, Zequn Sun, Yiqiao Jiang, Kexin Han, and Wei Hu. 2022. Facing changes: Continual entity alignment for growing knowledge graphs. *ArXiv*, abs/2207.11436.

Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual knowledge graph alignment via graph convolutional networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 349–357, Brussels, Belgium. Association for Computational Linguistics.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. *ArXiv*, abs/1908.08210.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1271–1279, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *AAAI Conference on Artificial Intelligence*.

Hsiu-Wei Yang, Yanyan Zou, Peng Shi, Wei Lu, Jimmy Lin, and Xu Sun. 2019. Aligning cross-lingual entities with multi-aspect information. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4431–4441, Hong Kong, China. Association for Computational Linguistics.

Donghan Yu, Yiming Yang, Ruohong Zhang, and Yuexin Wu. 2021. Knowledge embedding based graph convolutional network. In *Proceedings of the Web Conference 2021*, WWW '21, page 1619–1628, New York, NY, USA. Association for Computing Machinery.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 571–581, Vancouver, Canada. Association for Computational Linguistics.

Weixin Zeng, Jiuyang Tang, and Xiang Zhao. 2019. Iterative representation learning for entity alignment leveraging textual information. In *PKDD/ECML Workshops*.

Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2017. Iterative entity alignment via joint knowledge embeddings. In *International Joint Conference on Artificial Intelligence*.

A Method Appendix

A.1 Detail of factual triplet semantics incorporation

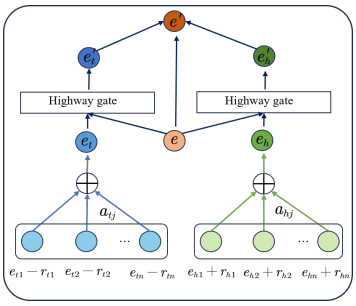

Figure 4: Combine the factual triplets

The detail of factual triplet semantics incorporation is shown in Figure 4. For a given entity e, it may act as either a head or tail entity in a factual triplet (head entity, relation, tail entity). In response to these two different cases, the following processing methods are used. Specifically, we employ the notion of $TransE$ to represent triplets, which assumes $h + r \approx t$ for a triplet. When the entity e serves as the head entity in a triplet, it has corresponding neighbors $\{(r_{t1}, e_{t1}), (r_{t2}, e_{t2}), \ldots, (r_{tn}, e_{tn})\}$, where n represents the number of neighbors. Therefore, we can represent the entity e using its neighbors as $\{\boldsymbol{e}_{t1}\boldsymbol{r}_{t1}, \ldots, \boldsymbol{e}_{tn}\boldsymbol{r}_{tn}\}$. When e

	DBP_{ZH-EN}					DBP_{JA-EN}					DBP_{FR-EN}				
	$\|\mathcal{T}\|_{ZH}$	$\|\mathcal{T}\|_{EN}$	$\|\mathcal{A}_s\|$	$\|\mathcal{A}_v\|$	$\|\mathcal{A}_p\|$	$\|\mathcal{T}\|_{JA}$	$\|\mathcal{T}\|_{EN}$	$\|\mathcal{A}_s\|$	$\|\mathcal{A}_v\|$	$\|\mathcal{A}_p\|$	$\|\mathcal{T}\|_{FR}$	$\|\mathcal{T}\|_{EN}$	$\|\mathcal{A}_s\|$	$\|\mathcal{A}_v\|$	$\|\mathcal{A}_p\|$
t=0	70,414	95,142	3,623	1,811	12,682	77,214	93,484	3,750	1,875	13,127	105,998	115,722	3,727	1,863	13,048
t=1	103,982	154,833	3,623	1,811	14,213	112,268	150,636	3,750	1,875	15,079	148,274	184,132	3,727	1,863	15,875
t=2	137,280	213,405	3,623	1,811	16,296	147,097	207,056	3,750	1,875	18,092	191,697	251,591	3,727	1,863	20,481
t=3	173,740	278,076	3,623	1,811	18,716	185,398	270,469	3,750	1,875	21,690	239,861	326,689	3,727	1,863	25,753
t=4	213,814	351,659	3,623	1,811	21,473	227,852	341,432	3,750	1,875	25,656	293,376	411,528	3,727	1,863	31,564
t=5	258,311	434,683	3,623	1,811	24,678	274,884	421,971	3,750	1,875	29,782	352,886	507,793	3,727	1,863	37,592

Table 3: Statistics of the three datasets

serves as the tail entity, the neighbors are $\{(r_{h1}, e_{h1}), (r_{h2}, e_{h2}), \ldots, (r_{hn}, e_{hn})\}$. The embedding table becomes: $\{\boldsymbol{e}_{h1} + \boldsymbol{r}_{h1}, \ldots, \boldsymbol{e}_{hn} + \boldsymbol{r}_{hn}\}$.

For an entity e, we have multiple embedding representations: $\{\boldsymbol{e}_{t1} - \boldsymbol{r}_{t1}, \ldots, \boldsymbol{e}_{tn} - \boldsymbol{r}_{tn}\}$, $\{\boldsymbol{e}_{h1} + \boldsymbol{r}_{h1}, \ldots, \boldsymbol{e}_{hn} + \boldsymbol{r}_{hn}\}$. To better utilize these embeddings, we use an attention mechanism to combine them.

B Experiment Appendix

B.1 Dataset

The detail of the dataset is shown in Table 3. It includes 5 snapshots to simulate the growth of KGs. The two knowledge graphs from the cross-lingual entity alignment dataset DBP15K serve as the first snapshot. Each subsequent snapshot grows by adding 20% more triplets based on the previous snapshot.

B.2 Implementation details

We set the embedding dimensions for entities and relations to be 100. For attributes and entity names, after encoding with LaBSE, we use PCA to reduce their dimensions to 100. The embedding similarity metric used is CSLS. As for the hyperparameters, we set $\alpha = 0.1$, $\beta = 0.3$, $m = 1000$, $l_r = 0.005$, and dropout_rate $= 0.3$.

B.3 Integration of attributes

The experimental results are shown in Table 4. It can be seen that attribute information has brought significant performance improvements across all snapshots on the three datasets. Specifically, for GCN-Align, KEGCN, and DINGAL-O, which are relatively simple GCN models unable to capture structural information effectively, utilizing the text features of attributes can lead to substantial performance gains. While Dual-AMN, with its excellent graph encoder, captures KG structural information well, experimental results indicate that textual features of attributes can further enhance its performance, showing the positive effect of attributes on entity alignment tasks. Even ContEA, specifically designed for dynamic EA task, would yield a performance improvement of 6.8% ($t = 1$), 4.9% ($t = 2$) and 4% ($t = 2$) on DBP_{ZH-EN} when using merged attribute triplets. Although combining attributes can enhance the performance of various models to some extent, our model still achieves the best performance on each snapshot of every dataset. Even when removing attribute information, our method outperforms models that incorporate attribute information (such as Dual-AMN and DINGAL-O). This further confirms the effectiveness of our approach.

It is worth noting that for the DINGAL-O model on the FR-EN dataset, utilizing attributes may lead to a decrease in precision but a significant improvement in recall. Higher recall indicates the discovery of more correct potential alignments, consistent with Hit@1. This aligns with our expectations as we first need to ensure a high recall to find as many alignments as possible.

Furthermore, the effect of attributes is more pronounced on the ZH-EN dataset. This is because Chinese and English exhibit significant differences, which are mitigated by the LaBSE cross-lingual encoder.

C Related Work Appendix

C.1 Static entity alignment

Currently, most methods consider entity alignment in static scenarios, with the mainstream approach being embedding-based entity alignment methods. The approach involves using a KG encoder to represent entities as vectors and then finding equivalent entities based on specific similarity measures. Based on different KG encoders, they can be classified into two categories: translation-based(Lin et al., 2019; Pei et al., 2019; Sun et al., 2019a; Zhu et al.,

2017) and GNN-based(Mao et al., 2021, 2020; Sun et al., 2019b; Wang et al., 2018; Wu et al., 2019). Translation-based models mainly use KG embedding (KGE) techniques(Wu et al., 2019) to embed entities into vectors and map the embedded representations of entities to the same vector space based on pre-aligned entities, which are the training data. GNN-based models, on the other hand, utilize the idea of twin GNN, where two KG encoders share parameters to encode entities from two KGs into vectors without the need to map pre-aligned entities to the same vector space. Due to their simplicity and high performance, GNN-based models are currently the mainstream entity alignment models.

C.2 Dynamic entity alignment

Few works focus on dynamic entity alignment. DINGA(Yan et al., 2021) first proposed the task of dynamic entity alignment. They believed that the difficulty of this task lies in the challenging update of embeddings, which is highly coupled with the KG encoder and the topology structure of the graph. Therefore, they proposed to address the dynamic entity alignment task by reducing the coupling. DINGAL has a variant called DINGAL-O, which uses previously learned model parameters to perform alignment predictions for new entities. ContEA(Wang et al., 2022) proposed a bidirectional matching strategy to select high-confidence alignment predictions to cope with the growth of the knowledge graph when searching for new alignments. By using high-confidence predictions and affected entities to fine-tune the previous model parameters, they achieved better results. They also introduced a heuristic strategy to resolve alignment conflicts during the stage.

		t = 0			t = 1			t = 2			t = 3		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
DBP_{ZH-EN}	GCN-Align	56.5	21.2	30.8	27.5	17.0	21.0	17.3	14.0	15.5	12.3	11.7	12.0
	GCN-Align$^{\diamond}$	**81.3**	**50.7**	**62.5**	**55.5**	**47.5**	**51.2**	**42.1**	**42.3**	**42.2**	**33.1**	**37.3**	**35.1**
	KEGCN	56.9	22.1	31.9	33.8	18.0	23.5	27.3	25.3	19.6	23.0	12.8	16.5
	KEGCN $^{\diamond}$	**82.4**	**53.7**	**65.0**	**56.6**	**49.9**	**53.0**	**43.3**	**44.3**	**43.8**	**35.2**	**39.0**	**37.0**
	Dual-AMN	83.3	60.3	69.9	50.6	51.4	51.0	38.5	45.6	41.7	28.5	28.6	28.6
	Dual-AMN $^{\diamond}$	**90.0**	**68.8**	**78.0**	**61.4**	**63.1**	**62.2**	**44.3**	**55.9**	**49.5**	-	-	-
	DINGAL-O*	49.7	19.5	28.0	37.0	15.8	22.2	31.5	13.5	18.9	25.1	11.1	15.4
	DINGAL-O $^{\diamond}$	**69.5**	**39.5**	**50.4**	**47.4**	**39.1**	**42.8**	**34.8**	**34.9**	**34.8**	**26.9**	**31.0**	**28.8**
	ContEA	84.5	61.1	70.9	54.1	53.1	53.6	43.1	46.8	44.9	36.4	41.6	38.8
	ContEA $^{\diamond}$	**88.2**	**64.8**	**74.7**	**61.0**	**59.8**	**60.4**	**47.0**	**53.0**	**49.8**	**39.7**	**46.3**	**42.8**
	DEA-AttrAlign	87.3	65.7	75.0	60.4	57.8	59.1	49.1	50.5	49.8	42.1	43.8	42.9
	DEA-AttrAlign $^{\diamond}$	**91.1**	**71.5**	**80.1**	**66.4**	**64.7**	**65.5**	**52.4**	**57.5**	**54.9**	**45.1**	**50.6**	**47.7**
DBP_{JA-EN}	GCN-Align	62.1	24.4	35.0	31.9	19.3	24.1	21.6	15.7	18.2	16.2	12.9	14.4
	GCN-Align$^{\diamond}$	**79.1**	**46.3**	**58.4**	**51.1**	**42.8**	**46.6**	**37.6**	**36.7**	**37.1**	**29.9**	**31.1**	**30.5**
	KEGCN	58.4	22.7	32.7	36.2	18.2	24.3	29.3	14.9	19.8	25.7	12.4	16.7
	KEGCN $^{\diamond}$	**79.1**	**47.7**	**59.5**	**51.8**	**43.8**	**47.5**	**39.5**	**37.6**	**38.5**	**31.9**	**32.0**	**32.0**
	Dual-AMN	84.4	59.0	69.5	54.4	50.6	52.5	42.2	43.0	42.7	28.5	28.6	28.6
	Dual-AMN $^{\diamond}$	**86.7**	**61.9**	**72.3**	**57.5**	**55.6**	**56.5**	**45.0**	**48.3**	**46.6**	-	-	-
	DINGAL-O*	54.0	22.7	32.0	39.1	17.4	24.1	32.8	13.7	19.4	27.1	11.3	15.9
	DINGAL-O $^{\diamond}$	**66.4**	**35.9**	**46.6**	**43.4**	**35.1**	**38.8**	**31.1**	**30.0**	**30.6**	**24.2**	**25.6**	**24.9**
	ContEA	85.5	60.5	70.9	57.7	52.2	54.8	46.5	44.3	45.4	39.8	37.9	38.9
	ContEA $^{\diamond}$	**87.5**	**63.2**	**73.4**	**61.2**	**57.0**	**59.0**	**48.9**	**48.9**	**48.9**	**41.8**	**41.4**	**41.6**
	DEA-AttrAlign	88.6	66.1	75.7	61.3	55.9	58.5	50.4	46.5	48.4	43.7	38.7	44.1
	DEA-AttrAlign $^{\diamond}$	**90.2**	**68.5**	**77.8**	**62.4**	**61.3**	**61.8**	**50.1**	**52.6**	**51.3**	**41.7**	**44.2**	**42.9**
DBP_{FR-EN}	GCN-Align	62.1	24.4	35.0	31.9	19.3	24.1	21.6	15.7	18.2	16.2	12.9	14.4
	GCN-Align$^{\diamond}$	**79.1**	**46.3**	**58.4**	**51.1**	**42.8**	**46.6**	**37.6**	**36.7**	**37.1**	**29.9**	**31.1**	**30.5**
	KEGCN	58.4	22.7	32.7	36.2	18.2	24.3	29.3	14.9	19.8	25.7	12.4	16.7
	KEGCN $^{\diamond}$	**79.1**	**47.7**	**59.5**	**51.8**	**43.8**	**47.5**	**39.5**	**37.6**	**38.5**	**31.9**	**32.0**	**32.0**
	Dual-AMN	84.4	59.0	69.5	54.4	50.6	52.5	42.2	43.0	42.7	28.5	28.6	28.6
	Dual-AMN $^{\diamond}$	**86.7**	**61.9**	**72.3**	**57.5**	**55.6**	**56.5**	**45.0**	**48.3**	**46.6**	-	-	-
	DINGAL-O*	54.0	22.7	32.0	39.1	17.4	24.1	32.8	13.7	19.4	27.1	11.3	15.9
	DINGAL-O $^{\diamond}$	**66.4**	**35.9**	**46.6**	**43.4**	**35.1**	**38.8**	**31.1**	**30.0**	**30.6**	**24.2**	**25.6**	**24.9**
	ContEA	86.8	63.5	73.3	56.6	52.4	54.4	45.2	42.6	43.8	38.7	35.4	37.0
	ContEA $^{\diamond}$	**87.0**	**63.9**	**73.7**	**59.6**	**54.5**	**56.9**	**47.2**	**43.5**	**45.3**	**40.2**	**35.3**	**37.6**
	DEA-AttrAlign	85.7	62.1	72.0	55.5	49.7	52.5	46.0	39.6	42.6	41.3	31.8	35.9
	DEA-AttrAlign $^{\diamond}$	**91.8**	**75.4**	**82.8**	**59.2**	**64.2**	**61.6**	**46.6**	**51.3**	**48.8**	**39.0**	**41.5**	**40.2**

Table 4: Results of entity alignment with attribute integration. The symbol $^{\diamond}$ represents the results utilizing attribute information.