A Scalable Method for Large-scale Entity Alignment via Multi-Channel Retrieval and Fusion

Ningxin Chen[1], Zhichun Wang[2,*,†]

Beijing Normal University, 100875, Beijing

Abstract

Entity **A**lignment (EA) aims to identify equivalent entities across different knowledge graphs. Existing methods mainly focus on EA for small to medium-sized knowledge graphs and are ineffective for large-scale knowledge graphs. This paper proposes a method called ScaleEA (**S**calable **E**ntity **A**lignment Method) for large-scale EA tasks, which can minimize the number of model parameters and training time while ensuring EA accuracy. ScaleEA combines the advantages of pre-trained language models and graph neural network. First, pre-trained language models obtain entity names and property encodings. Then, graph neural network aggregate relation and attribute information of each entity. Finally, candidate entity sets are retrieved and fused separately within each channel. Experiments on DBP1M show that compared with existing models, ScaleEA has significant advantages in both time and alignment accuracy.

Keywords

Entity Alignment, Pre-trained Language Model, GNN

1. Introduction

Knowledge **G**raph (**KG**) storing entities, relations and attributes in graph, has been widely used in applications for recommendation system [1], natural language processing [2], and question answering [3]. Current KGs are heterogeneous and cannot be merged together easily. If multiple KGs from different sources can be combined, a more complete and detailed KG will help accomplish applications. **E**ntity **A**lignment (**EA**) is a fundamental and critical step in solving KG heterogeneity problem. Its goal is to identify entities with the same referential meaning in different knowledge graphs. These aligned entities will be used for subsequent graph fusion. The main process of EA is: (1) collect seed set of aligned entities; (2) build an EA model and train it under the supervision of seed alignments; (3) use the trained model to discover potential aligned entities.

Existing EA solutions mainly focus on: (1) KG structure (2) entity property information. KG stores relations between entities and their neighbors. The assumption of structure alignment is that two aligned entities are likely to have aligned neighbors [4], and the neighbors closer to entities are more likely to transmit EA information. Besides, aligned entities also share similar property information, such as entity names.

However, many of these methods suffer from significant scalability issues. While models achieve high accuracy in small KGs, they cannot be scaled to fit real-world KGs, which have million entities far more than artificially constructed KGs. The larger KG lead to greater memory consumption, including both entity embedding and model parameters, and higher training costs. The mainstream approach is to divide large KG into smaller ones, while inevitably loses structure and property information.

This article proposes an efficient approach: build a scalable model using property information to assist with alignment, and conduct as little training as possible overall. The specific approach is as follows. First, fine-tune cross-lingual pre-trained BERT model on a small dataset, using entity names of seed alignments as input. Second, directly apply the fine-tuned model to larger KGs, obtaining initial entity embedding from entity name, and obtain relation, attribute, and value embedding from their names. Third, add relation and attribute channels to aggregate information from both neighboring

OM-2024: The 19th International Workshop on Ontology Matching collocated with the 23rd International Semantic Web Conference (ISWC 2024), November 11th, Baltimore, USA.

nxchenbnu@mail.bnu.edu.cn (N. Chen); zcwang@bnu.edu.cn (Z. Wang)

subgraphs. The name, relation, and attribute channels are used individually for target retrieval, and then we fuse candidate entity set to obtain complete candidate sets.

Overall, the innovative points of this article are as follows:

(1) Fine-tune the pre-trained language model BERT to directly generate entity embedding, the training parameters only from model.

(2) Directly apply the fine-tuned model to a large dataset, fully utilizing the similarity in the distribution of knowledge graph. Without any training, the required time is only for computing entity embedding.

(3) Use graph neural network to directly aggregate relation and attribute information from neighbouring subgraphs, which compensates for property information heterogeneity.

2. Related Work

2.1. Embedding Based Entity Alignment

Early EA methods relied on handcrafted features, crowdsourcing, and OWL semantic methods [5, 6, 7, 8]. Due to the reliance on manual rules, they are not suitable for highly heterogeneous linguistic scenarios. With the development of deep learning, today EA methods focus on using graph structure, which further can be divided into two categories, translation-based EA and GNN-based EA. Translation-based EA originated from TransE [9], with other representative studies including MTransE [10], AlignE [11], and SEA [12]. Some of them focus on exploring long-term relational dependencies of entities, such as IPTransE [13], RSN4EA [14], and IMEA [15]. With GNN achieving better results in graph representation, recent research pays more attention to GNN-based EA [16, 17, 18, 19, 19, 20].

In addition, many methods have revealed that KG's property information can improve the accuracy of EA by supplementing structure information, including names [18, 21, 22, 23, 4, 24, 25], descriptions [4, 24, 26], images [27], and attributes [25, 12, 28]. Since each entity has own name and it does not need to be pre-processed, most methods tend to use name.

2.2. Entity Alignment for Large-scale KGs

Although embedding based EA methods achieved high accuracy in finding potential aligned entities of small-scale datasets, they cannot be well scaled to large-scale KGs. Faced with large-scale KGs, most of them lead to insufficient GPU memory and very long time. Recent research decompose the large-scale EA task into independent subtasks [29, 30, 31, 32], and each subtask includes small subgraphs generated from original KGs. For each subtask, existing EA models can be naturally combined to predict unknown mappings based on seed alignments.

There are four representative graph division alignment methods.

LargeEA [29] first uses METIS to divide KGs into independent subgraphs. After dividing source KG, edge weights of target KG are modified based on seed entities and METIS is used again to partition target KG. LIME [30] uses a bidirectional partition method to preserve KG structure and further increases the coverage of seed mappings in each subtask. While compared with LargeEA, it doubles the partitioning time. DivEA [31] also divides source KG, then it expands mapping target subgraphs by modeling unmatched entities to decrease information loss. LargeGNN [32] merges the seed alignments from two KGs to form a unified KG and then partition it, ensuring all pre-aligned entities are located in the same subgraph blocks. It designs centrality-based subgraph generation and cross subgraph negative sampling algorithms to recall landmark entities. In conclusion, graph division methods solve the memory problem and speed up the EA task, however, it inevitably lowers precision because of incomplete graph context.

3. ScaleEA Method

The core of the proposed ScaleEA (Scalable Entity Alignment Method) method is to fine-tune pre-trained language model on small data, and calculate the initial vector embedding of entity names, relations,

attributes, and values. Then, entity embedding are generated separately for the three respective channels of name, relation, and attribute, and candidate entity sets are retrieved separately within each channel. Finally, candidate sets from three channels are sorted and fused together to obtain the target aligned entities.

3.1. Problem Definition and ScaleEA Framework

3.1.1. Knowledge Graph

A Knowledge Graph (KG) can be represented as $G = (E, R, A, V, T)$, where E is the set of entities, R the set of relations between entities, A the set of attributes, V the set of property values of attributes, and T the set of triples in the graph, which can be categorized into relation triples T_r and attribute triples T_a. Relation triples represent the connection between two entities through a directed relation, such as (Beijing, the capital of, China), indicating that Beijing is the capital of China. Attribute triples represent the attributes of an entity, where the property value can be a string or a number, such as (Kobe Bryant, birthDate, "1978-08-23"), indicating that Kobe Bryant's birth date is August 23rd, 1978.

3.1.2. Entity Alignment

Given a source KG G_s and a target KG G_t, entity alignment is to find equivalent entity sets φ between the two KGs, $\varphi = \{(e_s, e_t)|e_s \equiv e_t, e_s \in G_s, e_t \in G_t\}$, where φ represents the aligned entities. During the experiments, we divide the data into training and testing sets, with the training set usually accounting for 30% of the data.

3.1.3. ScaleEA Framework

The framework of ScaleEA consists of three modules, including the entity information encoding module, similarity calculation module, and channel ensemble module, as shown in Fig.1.

(1) Entity information encoding module: We fine-tune BERT on a small-scale multilingual dataset. After freezing the model parameters, input text representations of entity names, relations, attributes, and values to obtain corresponding embedding. Use three channels to obtain embedding, which include relation channel (relation, neighboring entity), attribute channel (attribute, value), and entity name channel.

(2) Similarity calculation module: Within each channel, the target entity is retrieved based on the similarity of the embedding between the two KGs.

(3) Channel ensemble module: The candidate entity pairs from the entity, relation, and attribute channels are fused to obtain the final aligned entities.

3.2. Property Encoding Based on Pretrained Models

This part introduces the generation method and specific details of entity property encoding. The overall structure is based on BERT, which generates corresponding embedding for entities, relations, attributes, and values based on their text, and fine-tunes BERT according to the aligned seed entities. BERT has strong understanding ability for semantic information and can be applied to various tasks after fine-tuning and EA is regarded as a classification task. Given a set of seed entities φ, training data D is constructed as,

$$D = \{(e, e', e-, e'-)|e \equiv e', (e, e-) \in G_s, (e', e'-) \in G_t\} \tag{1}$$

where $(e, e') \in \varphi$, $e-$ and $e'-$ are randomly sampled negative entities within the G_s, G_t. For the input of the model, both the entity name or description information can be used. In practical experiments, although BERT has better processing ability for longer texts and they contain richer semantics, for shorter processing time, the entity name will be prioritized.

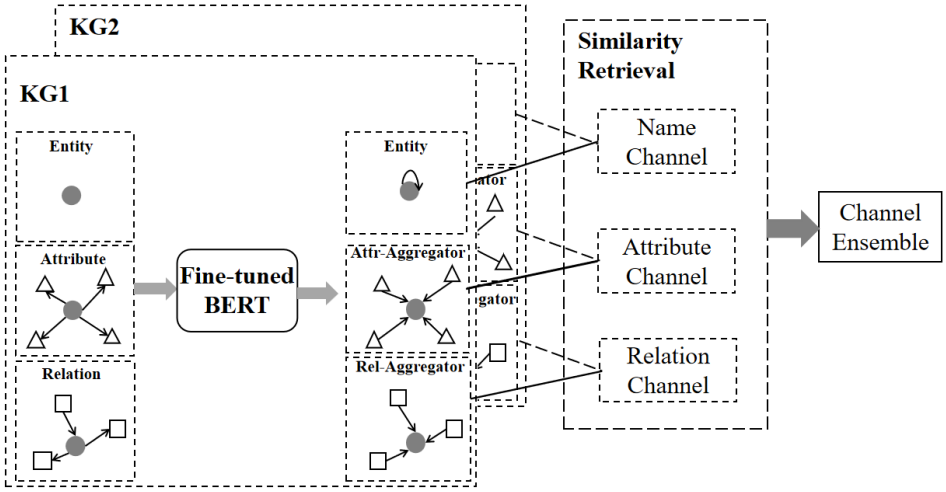

Figure 1: Framework of ScaleEA

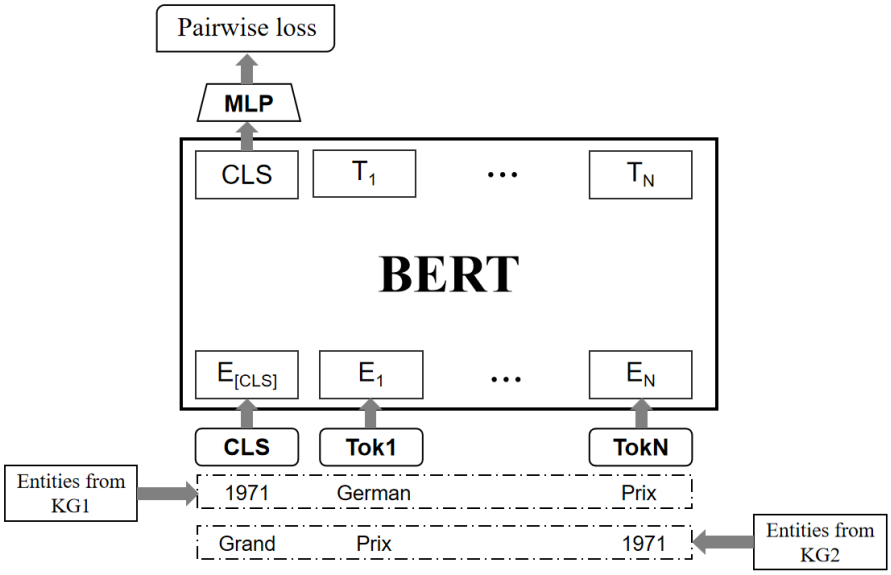

Figure 2: Fine-tuning with pre-trained cross-lingual BERT. A batch of text is inputted each time, and the corresponding output at the CLS position is obtained as the initial embedding of the input.

The overall process is shown in Fig.2. A pre-trained cross-lingual BERT is used to input the entity name and obtain the entity embedding from [CLS] vector. The [CLS] vector in the BERT hidden layer represents the semantic information of the entire sentence. In order to reduce the dimension and improve task performance, an additional trainable MLP layer is added to reduce the embedding dimensions from 768 to 300. The final representation of the entity embedding is as follows,

$$\hat{e} = MLP(Bert(e)[CLS]) \tag{2}$$

For the source entity embedding $\hat{e}$, the probability of aligning with an aligned entity $\hat{e'}$ is given by:

$$p(\hat{e}, \hat{e'}) = \frac{sim(\hat{e}, \hat{e'})}{sim(\hat{e}, \hat{e'}) + sim(e\hat{-}, \hat{e'}) + sim(\hat{e}, e'\hat{-})} \tag{3}$$

where $sim(\cdot, \cdot)$ is the cosine similarity between entity embedding. The probability of aligned entities

found p_e is calculated from entity pairs $(\hat{e}, \hat{e'})$, $(e\hat{-}, \hat{e'})$ and $(\hat{e}, e'\hat{-})$, which refers to softmax. The ground-truth distribution of entity pairs is q_e, where aligned entity pairs is 1, and not unaligned entity pairs is 0. The cross-entropy function is used to calculate the loss l_1 between the ground truth q_e distribution and the predicted distribution p_e by the model,

$$l_1 = \sum_{e \in \varphi_{train}} CE(q_e, p_e) \tag{4}$$

where l_1 is used for backpropagation and fine-tune the BERT and MLP layers.

The model is trained on a small dataset and is used to generate embedding for entity property information, which is the basis for subsequent multi-channel information fusion. The training of the model can be end-to-end, or the parameters can be trained first and then frozen. Due to the large-scale of the task, the latter is chosen in specific practice to save time. In addition, the trained model will be directly used in the large dataset to calculate the corresponding initial embedding.

The initial embedding of entity relation and attribute information do not consider the graph structure and do not require vector random initialization followed by training. Instead, the BERT fine-tuned in the previous section is directly inputted with the relation name, neighbor entity name, attribute name, and value to calculate the embedding for each item. The overall process is shown in Figure 3. The information is represented as the concatenation of the relation and neighboring entity $(\hat{r}, \hat{e})$, and the attribute information is represented as the concatenation of the attribute and value $(\hat{a}, \hat{v})$.

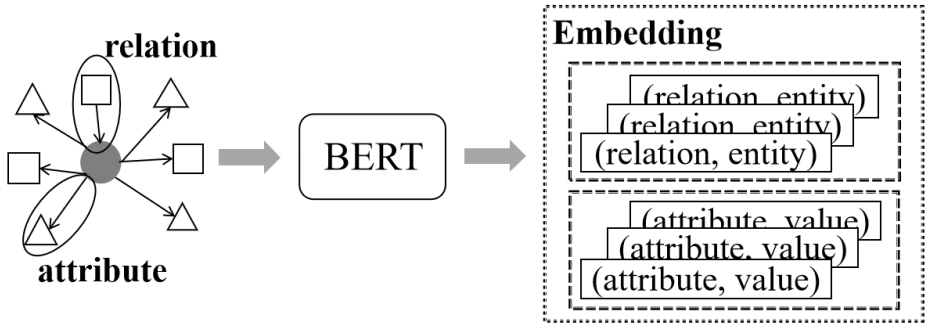

Figure 3: Initialize embedding of relation and attribute information. Each is represented as the concatenation of pairs.

3.3. Entity, Relation, and Attribute Channels Encoding

This chapter introduces attribute encoding based on graph neural network. In the previous section, pure semantic calculation was used, which resulted in the loss of a large amount of structural information. Graph neural network can be used to compensate for this deficiency. By aggregating entity relation and attribute information, the relation and attribute encoding of the entity are calculated and generated. With the original entity encoding, a triple-channel encoding is formed for each entity, fully integrating its semantic and structural information. The overall process is shown in Fig.3.

From the perspective of the entity channel, the entity embedding $\hat{e}$ in Section 3.2 is used, and the entity channel encoding $InfoE_e = \hat{e}$ contains the semantic information of the entity name. In the relation channel, the information of neighbor triples $(e, r_1, ne_1), \cdots, (e, r_m, ne_m)$ of the entity e is used, and the aggregate function $\sigma(\cdot)$ is used to aggregate the information of triples. The entity encoding in the relation channel is as follows:

$$InfoE_r = \sigma_1([r_1; ne_1], \cdots, [r_m; ne_m]) \tag{5}$$

where $[r_i; ne_i]$ is the embedding concatenated in Section 3.2. In attribute channel, the method is similar to the relation channel, and the information of attribute triples $(e, a_1, v_1), \cdots, (e, a_m, v_m)$ is

aggregated:

$$InfoE_a = \sigma_2([a_1; v_1], \cdots, [a_n; v_n]) \tag{6}$$

where aggregation function σ_1 and σ_2 are supposed to be a function that can filter information.

3.3.1. Importance Aggregator

Instead of averaging the embedding, weights are added to represent the importance of attributes and relations. Since there are many entity relations and attributes, and the attribute values are obviously heterogeneous, it is necessary to select important relation and attribute features. Not all relations and attributes can represent the distinctive features of the entity. For example, many cities share the same time zone, so the importance of attributes such as time zone offset should be relatively low. Referring to the method of relation inference in PARIS [33], the importance of relation r_i and attribute a_i is represented as:

$$\begin{aligned} func(r_i) &= \frac{|\{ne|(e, r_i, ne) \in T_r\}|}{|\{(e, ne)|(e, r_i, ne) \in T_r\}|} \\ func(a_i) &= \frac{|\{v|(e, a_i, v) \in T_a\}|}{|\{(e, v)|(e, a_i, v) \in T_a\}|} \end{aligned} \tag{7}$$

when $fun(\cdot) = 1$, the neighbors and attribute values corresponding to different relations and attributes must be different, which means that this relation or attribute can represent the unique property of the entity.

When calculating the channel encoding by importance weighted summation, it is also necessary to normalize the importance $fun(\cdot)$ of all neighboring relations and attributes of the entity e_i:

$$\begin{aligned} func'(r_j) &= \frac{exp(func(r_j))}{\sum_{r_k \in N(e_i)} exp(func(r_k))} \\ func'(a_j) &= \frac{exp(func(a_j))}{\sum_{a_k \in N(e_i)} exp(func(a_k))} \end{aligned} \tag{8}$$

where $N(e_i)$ represents all neighboring relations or attributes of the entity e_i.

3.4. Entity Retrieval and Results Fusion in the Isolation Channels

Based on the previous process, we obtained entity embedding from entity, relation, and attribute channels respectively. The challenge in this section is how to fuse the information in the three channels without interfering with each other. We use the idea of **channel isolation**, which is to perform entity retrieval separately in the three channels, forming their respective candidate entity sets. We do not perform fusion training of embedding, but focus effectively fusing candidate entities, because relations, attributes, and values are heterogeneous, and the concatenation or training makes no sense and will harm the performance.

Facing the retrieval of large-scale KGs, this article uses the Locality Sensitive Hashing (LSH) algorithm to perform approximate nearest neighbor search in high-dimensional space. This algorithm has good accuracy and scalability. In the three channels, retrieval is performed separately, and the similarity threshold γ is set filter out low similarity entities to form target entity set $S_{channel}$:

$$S_{channel} = \{e'|dist(e, e') < \gamma\} \tag{9}$$

where $dist(\cdot)$ is calculated by LSH, and we form S_{name}, $S_{relation}$ and $S_{attribute}$ respectively.

To effectively fuse candidate entity sets, it is necessary to correctly sort them. The experimental results show that the accuracy of one-channel sorting is $name > relation > channel$, so the candidate sets are sorted one-dimensionally according to this order. When the matching of valid target entities cannot be achieved because of the mismatching names, the relation and attribute channels can provide structural information as compensation, and the noise does not interfere with each other.

4. Experiment

This chapter presents a series of experiments to verify the effectiveness of ScaleEA, including experimental setting, results analysis, ablation experiment, and scalability test. The experimental settings introduce the datasets, evaluation metrics, details, and benchmarks, providing necessary background for the subsequent analysis. The result analysis focuses on the performance of ScaleEA on large, medium, and small KGs, indicating it has advantages in memory and time efficiency. The ablation experiments explore the impact of ScaleEA modules. The scalability test evaluates the performance of the model under different dataset sizes, revealing that the required time increases almost linearly with dataset size while accuracy maintains at a good level.

4.1. Experiment Setting

4.1.1. Dataset

The experiments use cross-lingual datasets generated from DBpedia, which are widely used in EA task. In respect of KG languages, there are EN-FR (English with French) and EN-DE (English with German). In respect of scales, there are 15K, 100K, and 1M, which represent the approximate number of entities in one KG. IDS15K and IDS100K are benchmark datasets (V1) from OpenEA [34]. DBP1M is formed by LargeGNN [32], which is specifically designed for large-scale EA tasks by removing all entities with no neighbours and leaving half of entities unable to find their aligned counterparts. The number of entities, relations, attributes, and values in the dataset are shown in Table 1. The meaning of variables are as follows: entity, number of entities in corresponding KG; rel, number of relations; attr, number of attributes; val, number of attribute values. No attribute and value information of large-scale dataset are given since they won't be used. Following LargeGNN [32], the dataset is divided into train, test, and validation set at a 30%, 60%, and 10% ratio.

Table 1
Details about IDS15K, IDS100K, DBP1M datasets.

Datasets		Entity	Rel	Attr	Val
IDS15K EN-FR	EN	15,000	267	308	45,784
	FR	15,000	210	404	39,817
IDS15K EN-DE	EN	15,000	215	286	49,956
	DE	15,000	131	194	57,663
IDS100K EN-FR	EN	100,000	400	466	253,146
	FR	100,000	300	519	228,356
IDS100K EN-DE	EN	100,000	381	451	271,949
	DE	100,000	196	252	326,166
DBP1M EN-FR	EN	1,211,270	595	-	-
	FR	1,176,869	400	-	-
DBP1M EN-DE	EN	1,160,306	596	-	-
	DE	849,646	247	-	-

4.1.2. Metrics

We use Hits@N, MRR, and running time as our metrics. **Hits@N** also known as N-hit rate, indicating the proportion of aligned entities in the top N matching results. The higher number equals better performance in covering aligned entities in the top N query results. **MRR**, which stands for **M**ean **R**eciprocal **R**ank, indicating the average of the reciprocal rank of aligned entities in the query results. Specifically, if the rank of aligned entity is k, then the MRR is $1/k$. If no matching entity is found, then the MRR is 0. **Running time** refers to the time required for the entire experiment. In large-scale experiments, time is especially significant.

4.1.3. Implementation Details

The specific experimental details of ScaleEA are as follows. In the model fine-tuning phase, the dimension of the vector embedding is fixed at 300, and the AdamW optimizer is used to train for 15 epochs. As there are a large number of noisy property values in large-scale KG, for efficiency, the attribute channel is only added to the small-scale dataset. Only entity name and relation channels are used to align large-scale KGs. In entity retrieval phase, a distance threshold $\gamma = 0.15$ is used to filter low-similarity entities. All experiments are conducted on one NVIDIA GeForce RTX 3090 GPU.

4.1.4. Baselines

As most embedding-based methods cannot run on large-scale datasets, we only compare our approach with large-scale EA methods. All these methods divide graphs into small blocks, then do block alignment. (1) LargeEA [29] partitions the source and target KGs using METIS and constructs entity alignment blocks. (2) LIME [30] follows the center idea of LargeEA, but further uses a bidirectional partitioning method, which can preserve the KG structure and increases the recall of seed mappings in each subtask. (3) DivEA [31] build context graphs for subtasks to save more important information, and increase the recall of potential aligned entities. (4) LargeGNN [32] merges the two KGs and then partition unified KG. To better preserve landmark entities, it designs a subgraph generation algorithm using centrality and cross-subgraph negative sampling. Later analysis reveal that name channel counts in EA task, while it not been used in original LIME, DivEA, LargeGNN. A simple but effective name information combination method is left for future work.

4.2. Results Analysis

Table2 shows results of ScaleEA method on the DBP-datasets. Compared with existing models, our method achieves best performance in terms of accuracy and time, which outperforms the second-best baseline model LargeEA. The main reason why ScaleEA achieves good performance on such a large dataset is because of literal information. Although the improvement of hit@N is obvious, we also find that the increase of hit@5 is not as high as hit@1. The reason behind it comes from the rough channel fusion, as we only sort the candidate sets by a strict order: name, relation, attribute. If the name channel gives wrong candidates, they can't be filtered out effectively.

In terms of time, the required time for our method is also the shortest, as we don't need any training except fine-tuning. The other methods combine GNN training methods with graph division, training all sub-graphs to construct a robust EA model. For a fair comparison, we use the same training method RREA in different methods, and the results show training time is much longer than the division. For our method ScaleEA, since the overall training is minimal, the time is mainly spent on computing entity embedding. Although it's short, it is difficult to be further compressed.

4.2.1. Ablation Study

We conduct ablation experiments testing performance of ScaleEA individual channels. Since only the small dataset use all channels, the experimental results on the 15K dataset are presented here. The situations of other datasets are very similar, with some slight numerical changes. Compared with the complete ScaleEA, the ablation experimental groups are: (1) Full ScaleEA; (2) ScaleEA without name channel; (3) ScaleEA without relation channel; (4) ScaleEA without attribute channel. The results of the ablation experiments are shown in figure4.

There are four main observations from the ablation experiments: (1) Entity name channel has a significant impact on accuracy. (2) Addition of the relation and attribute channels effectively improves the experimental results. As the fusion method for channels is to directly merge the candidate entity sets in order, when entity names are heterogeneous and cannot locate similar target entities, the relation and attribute information can compensate for it and effectively find target entities. (3) The attribute channel performs poorly and does not contribute much to the overall results. There are two reasons for

Table 2
Comparison of ScaleEA and other methods on different datasets.

Methods	DBP1M EN-FR				
	Hit@1	Hit@5	MRR	Time(h)	Mem(GB)
LargeEA	0.759	0.816	0.784	2.75	>24
LIME	0.171	0.224	0.218	0.78	33.1
DivEA	0.233	0.319	0.278	4.33	22.51
LargeGNN	0.178	0.314	0.450	0.77	16.67
ScaleEA	**0.795**	**0.822**	**0.807**	**1.63**	**20.54**

Methods	DBP1M EN-DE				
	Hit@1	Hit@5	MRR	Time(h)	Mem(GB)
LargeEA	0.784	0.832	0.805	2.36	>24
LIME	0.157	0.212	0.204	0.78	16.67
DivEA	0.307	0.383	0.345	4.12	21.39
LargeGNN	0.225	0.355	0.289	0.56	16.67
ScaleEA	**0.818**	**0.847**	**0.831**	**1.35**	**19.36**

Methods	IDS15K EN-FR			
	Hit@1	Hit@5	MRR	Time(s)
LargeEA	0.735	0.875	0.796	239
LIME	0.607	0.891	0.704	354
DivEA	0.581	0.765	0.663	450
LargeGNN	0.710	0.880	0.784	86
ScaleEA	**0.894**	**0.914**	**0.917**	**174**

Methods	IDS15K EN-DE			
	Hit@1	Hit@5	MRR	Time(s)
LargeEA	0.892	0.934	0.910	215
LIME	0.732	0.893	0.802	387
DivEA	0.734	0.862	0.790	535
LargeGNN	0.760	0.892	0.842	110
ScaleEA	**0.897**	**0.922**	**0.926**	**201**

Methods	IDS100K EN-FR			
	Hit@1	Hit@5	MRR	Time(s)
LargeEA	0.839	0.875	0.860	979
LIME	0.412	0.574	0.525	2256
DivEA	0.430	0.584	0.505	4578
LargeGNN	0.395	0.611	0.496	1141
ScaleEA	**0.841**	**0.865**	**0.852**	**506**

Methods	IDS100K EN-DE			
	Hit@1	Hit@5	MRR	Time(s)
LargeEA	0.856	0.891	0.870	980
LIME	0.518	0.672	0.586	2412
DivEA	0.522	0.651	0.584	4718
LargeGNN	0.531	0.699	0.610	1435
ScaleEA	**0.853**	**0.883**	**0.866**	**507**

this: (1) Candidate sets are directly sorted in order and attribute channel is ranked last, so its results are only added when neither the entity nor the relation channels match any entities. (2) Heterogeneous and noisy attributes and values may harm final results. How to effectively filtering and utilizing attribute information is challenging in EA, and it's left for future work. (4) The complete ScaleEA significantly outperforms all ablation models.

4.2.2. Scalability Test

In general, ScaleEA has good scalability, as shown in Figure5. In terms of time, the overall time required by the model increases linearly with the rise of the dataset size, and the time required for the 1M is about ten times that of the 100K. Specially, The time for the 15K includes fine-tuning model, while the time for medium and large-scale is only related to the number of entities, relations, attributes, and values. In terms of accuracy, as the dataset size increases, the noises rise up, leading to a decrease in accuracy. While Hit@1 from 15K to 1M decreases by a maximum of 0.099, accuracy remains at a relatively high level. In terms of datasets, the required time and alignment accuracy for different datasets(EN-FR, EN-DE) are similar, Compared with competitors, ScaleEA is less affected by the dataset lingual property.

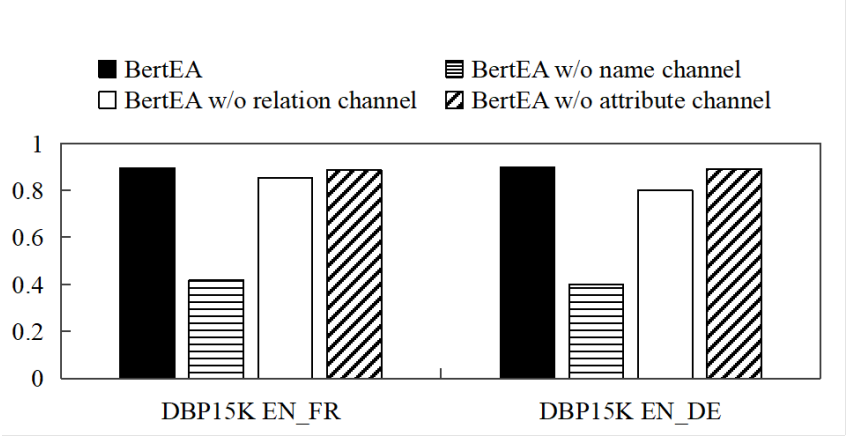

Figure 4: Ablation study

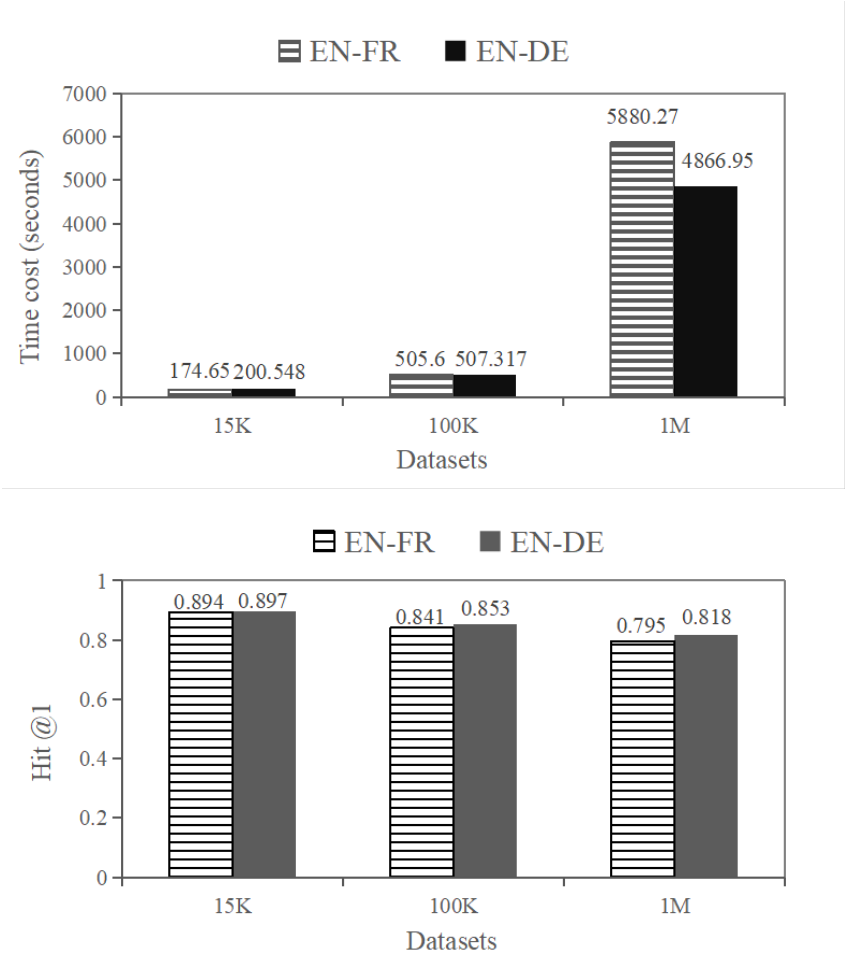

Figure 5: The scability test of ScaleEA.

5. Conclusion

This article proposes ScaleEA, a model for large-scale entity alignment tasks. Rather than partitioning large knowledge graph, ScaleEA fine-tune BERT and does not do additional training, ensuring the fewest training parameters and time. ScaleEA has significant advantages in terms of time and memory. The superior performance of ScaleEA reveals four points: (1) Use pre-trained language models to extract entity name embedding and match them with target entity has superior performance. (2) Fine-tune

on small datasets and directly apply the well-tuned model to large-scale datasets can achieve good results without training, and it also reduce memory burden. (3) Relation and attribute channels can effectively mitigate the heterogeneous interference of entity names. (4) Isolating candidate sets in each channel and directly sorting and merging sets can fully utilize structural information without causing interference.

In the future, we will explore better way to aggregate relation and attribute information by incorporating prior knowledge, mining potential structural information, and reducing noise interference from heterogeneous information. In each stage, we will explore more to shorten the computation time.

6. Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62276026).

References

[1] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering 34 (2020) 3549–3568.

[2] H. Fei, Y. Ren, Y. Zhang, D. Ji, X. Liang, Enriching contextualized language model from knowledge graph for biomedical information extraction, Briefings in bioinformatics 22 (2021) bbaa110.

[3] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, J.-R. Wen, A survey on complex knowledge base question answering: Methods, challenges and solutions, arXiv preprint arXiv:2105.11644 (2021).

[4] X. Tang, J. Zhang, B. Chen, Y. Yang, H. Chen, C. Li, Bert-int: a bert-based interaction model for knowledge graph alignment, interactions 100 (2020) e1.

[5] F. Mahdisoltani, J. Biega, F. M. Suchanek, Yago3: A knowledge base from multilingual wikipedias, in: CIDR, 2013.

[6] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledgebase, Communications of the ACM 57 (2014) 78–85.

[7] A. Gruenheid, D. Kossmann, R. Sukriti, F. Widmer, Crowdsourcing entity resolution: When is a= b?, Technical Report/ETH Zurich, Department of Computer Science 785 (2012).

[8] E. Jiménez-Ruiz, B. Cuenca Grau, Logmap: Logic-based and scalable ontology matching, in: The Semantic Web–ISWC 2011: 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I 10, Springer, 2011, pp. 273–288.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).

[10] M. Chen, Y. Tian, M. Yang, C. Zaniolo, Multilingual knowledge graph embeddings for cross-lingual knowledge alignment, arXiv preprint arXiv:1611.03954 (2016).

[11] Z. Sun, W. Hu, Q. Zhang, Y. Qu, Bootstrapping entity alignment with knowledge graph embedding., in: IJCAI, volume 18, 2018.

[12] S. Pei, L. Yu, R. Hoehndorf, X. Zhang, Semi-supervised entity alignment via knowledge graph embedding with awareness of degree difference, in: The world wide web conference, 2019, pp. 3130–3136.

[13] H. Zhu, R. Xie, Z. Liu, M. Sun, Iterative entity alignment via joint knowledge embeddings., in: IJCAI, volume 17, 2017, pp. 4258–4264.

[14] L. Guo, Z. Sun, W. Hu, Learning to exploit long-term relational dependencies in knowledge graphs, in: International conference on machine learning, PMLR, 2019, pp. 2505–2514.

[15] K. Xin, Z. Sun, W. Hua, W. Hu, X. Zhou, Informed multi-context entity alignment, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1197–1205.

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

[17] C. Li, Y. Cao, L. Hou, J. Shi, J. Li, T.-S. Chua, Semi-supervised entity alignment via joint knowledge embedding model and cross-graph model, Association for Computational Linguistics, 2019.
[18] X. Mao, W. Wang, H. Xu, M. Lan, Y. Wu, Mraea: an efficient and robust entity alignment approach for cross-lingual knowledge graph, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 420–428.
[19] Z. Sun, M. Chen, W. Hu, C. Wang, J. Dai, W. Zhang, Knowledge association with hyperbolic knowledge graph embeddings, arXiv preprint arXiv:2010.02162 (2020).
[20] Z. Wang, Q. Lv, X. Lan, Y. Zhang, Cross-lingual knowledge graph alignment via graph convolutional networks, in: Proceedings of the 2018 conference on empirical methods in natural language processing, 2018, pp. 349–357.
[21] Z. Liu, Y. Cao, L. Pan, J. Li, T.-S. Chua, Exploring and evaluating attributes, values, and structures for entity alignment, arXiv preprint arXiv:2010.03249 (2020).
[22] H. Nie, X. Han, L. Sun, C. M. Wong, Q. Chen, S. Wu, W. Zhang, Global structure and local semantics-preserved embeddings for entity alignment, in: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence, 2021, pp. 3658–3664.
[23] Z. Sun, W. Hu, C. Li, Cross-lingual entity alignment via joint attribute-preserving embedding, in: The Semantic Web–ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21–25, 2017, Proceedings, Part I 16, Springer, 2017, pp. 628–644.
[24] H.-W. Yang, Y. Zou, P. Shi, W. Lu, J. Lin, X. Sun, Aligning cross-lingual entities with multi-aspect information, arXiv preprint arXiv:1910.06575 (2019).
[25] Q. Zhang, Z. Sun, W. Hu, M. Chen, L. Guo, Y. Qu, Multi-view knowledge graph embedding for entity alignment, arXiv preprint arXiv:1906.02390 (2019).
[26] M. Chen, Y. Tian, K.-W. Chang, S. Skiena, C. Zaniolo, Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment, arXiv preprint arXiv:1806.06478 (2018).
[27] F. Liu, M. Chen, D. Roth, N. Collier, Visual pivoting for (unsupervised) entity alignment, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 4257–4266.
[28] K. Yang, S. Liu, J. Zhao, Y. Wang, B. Xie, Cotsae: co-training of structure and attribute embeddings for entity alignment, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 3025–3032.
[29] C. Ge, X. Liu, L. Chen, B. Zheng, Y. Gao, Largeea: Aligning entities for large-scale knowledge graphs, arXiv preprint arXiv:2108.05211 (2021).
[30] W. Zeng, X. Zhao, X. Li, J. Tang, W. Wang, On entity alignment at scale, The VLDB Journal 31 (2022) 1009–1033.
[31] B. Liu, W. Hua, G. Zuccon, G. Zhao, X. Zhang, High-quality task division for large-scale entity alignment, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 1258–1268.
[32] K. Xin, Z. Sun, W. Hua, W. Hu, J. Qu, X. Zhou, Large-scale entity alignment via knowledge graph merging, partitioning and embedding, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 2240–2249.
[33] F. M. Suchanek, S. Abiteboul, P. Senellart, Paris: Probabilistic alignment of relations, instances, and schema, arXiv preprint arXiv:1111.7164 (2011).
[34] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami, C. Li, A benchmarking study of embedding-based entity alignment for knowledge graphs, arXiv preprint arXiv:2003.07743 (2020).