OAG_{know}: Self-supervised Learning for Linking Knowledge Graphs

Xiao Liu, Li Mian, Yuxiao Dong, Fanjin Zhang, Jing Zhang, Jie Tang*, *IEEE Fellow*, Peng Zhang, Jibing Gong, Kuansan Wang, and Evgeny Kharlamov

Abstract—We propose a self-supervised embedding learning framework—SelfLinKG—to link concepts in heterogeneous knowledge graphs. Without any labeled data, SelfLinKG can achieve competitive performance against its supervised counterpart, and significantly outperforms state-of-the-art unsupervised methods by 26%-50% under linear classification protocol. The essential components of SelfLinKG are local attention-based encoding and momentum contrastive learning. The former aims to learn the graph representation using an attention network, while the latter is to learn a self-supervised model across knowledge graphs using contrastive learning. SelfLinKG has been deployed to build the the new version, called OAG_{know} of Open Academic Graph (OAG). All data and codes are publicly available.

Index Terms—Concept Linking, Self-supervised learning, Contrastive Learning, Knowledge Base

1 Introduction

Concept linking, with the goal of linking concepts of the same meaning, is critical for document-based data systems such as Academic Search (AMiner, Microsoft Academic Graph) and Question Answering Platform (Reddit, StackOverflow), where knowledge bases containing concepts and their relations are independently developed within each system to help complicated searching and reasoning. Usually, these knowledge bases are incomplete, and to complement each other via concept linking is important for advanced applications. For example, the topic classification of academic papers in AMiner depends on concept extraction in paper abstract. But the incompleteness of concept taxonomy and lack of concepts' text descriptions leads to bad performance. For instance, we cannot literally distinguish ambiguous concepts such as "entropy" in physics and "entropy" in information science. In the past decades, quite a few approaches have been proposed to address several related topics, such as entity linking [24], [44], [46], schema matching [14], [15], entity resolution [31], [45], and ontology alignment [28].

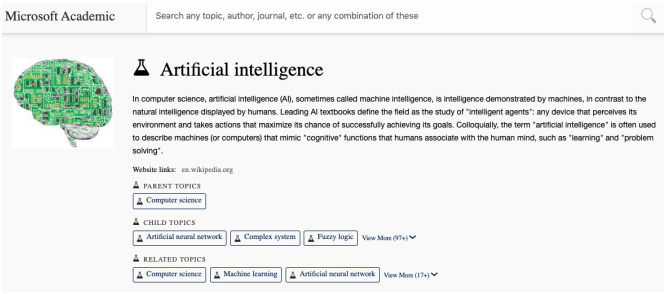

Fig. 1: An example of concept linking between the Open Academic Graph (OAG) and Wikipedia."Artificial intelligence" is a research concept in OAG, which is linked with the entry https://en.wikipedia.org/wiki/Artificial_intelligence in Wikipedia.

However, the problem of linking knowledge at the Web-scale remains an open question. Most of the aforementioned methods can only tackle the concept linking problem at relatively small-scale. In [44], the authors present a deep learning method—LinKG—to generate links between two of the largest publicly available academic graphs. However, the method operates in a (semi-)supervised way and requires sufficiently labeled data, usually *infeasible* when dealing with concept linking in the open Web. Moreover, the ambiguity issue with noise makes the problem more severe.

Current supervised embedding-based alignment algorithms fail to solve these problems. To enable large-scale and label-efficient concept linking, we have to develop a new method and evaluate it on publicly available large-scale heterogeneous knowledge bases. In this work, we make an attempt with both research and deployment purposes, to link research concepts (a.k.a., fields of study) in the Open Academic Graph (OAG) and concept entries in (En-

- *Xiao Liu, Fanjin Zhang, and Peng Zhang are with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. E-mail: liuxiao17@mails.tsinghua.edu.cn, zfj17@mails.tsinghua.edu.cn, zpjumper@gmail.com*
- *Li Mian is with the Beijing Institute of Technology, Beijing, China. Email: 1120161659@bit.edu.cn*
- *Yuxiao Dong and Kuansan Wang are with Microsoft Research, Redmond, WA 98052, USA. Email: ericdongyx@gmail.com, kuansan.wang@microsoft.com*
- *Jing Zhang is with the Renmin University of China, Beijing, China. Email: zhang-jing@ruc.edu.cn*
- *Jibing Gong is with the Department of Information Science and Engineering, Yanshan University, Qinhuangdao, China. Email: gongjibing@163.com*
- *Evgeny.Kharlamov is with Robert Bosch GmbH, Germany. Email: Evgeny.Kharlamov@de.bosch.com*
- *Jie Tang is with the Deptment of Computer Science and Technology, Tsinghua University and Tsinghua-Bosch Joint ML Center. E-mail: jietang@tsinghua.edu.cn, corresponding author*

. OAG_{know} data: https://aminer.cn/oag_know/
. Code: https://github.com/Xiao9905/OAG_know

glish) Wikipedia. OAG [44], which consists of the Microsoft Academic Graph (MAG) [26] and the AMiner Academic Graph [29], contains more than 700 million entities and 2 billion relationships, making it the largest publicly available academic entity graph to date. For the research purpose, we can have suitable large-scale datasets for evaluations. For deployment purpose, the OAG entity graph has several types of entities: papers, authors, institutions, conferences, journals, and research concepts. To tag the concepts to papers, e.g., to determine whether a paper is about "data mining", sufficient semantic information about the concept entities is desirable. Therefore, we propose to *enrich OAG concepts' semantic information by linking them with the same concept entries in Wikipedia.*

Figure 1 illustrates an example of the "Artificial intelligence" concept in OAG. In OAG, concepts are crawled from the Internet and then organized into taxonomy by calculating their pairwise subsumption (a form of co-occurrence) from millions of academic papers [25]. Thus the organization of this taxonomy is a bit different from Wikipedia. In addition, only 1/3 of the concepts are associated with Wikipedia entries (using the crawled URLs), leaving the rest of them with very little semantic information. From the figure, we can see at least two types of relations in OAG: *hypernym* and *related*. How to address and leverage the relation heterogeneity to improve concept linking is largely unexplored. In addition, there are many ambiguous and similar concepts. For the ambiguity issue, an example is that there are four different "entropy" entries in Wikipedia. For confusingly similar concepts, an example can be found between "artificial neural networks" in machine learning and "neural networks" in neuroscience in OAG. Therefore, the concept linking task must deal with these inherent issues. Finally, the OAG concept taxonomy covers over 679,921 concepts. To link OAG with Wikipedia, (semi-)supervised methods require massive label data, which is usually arduous and expensive to obtain.

In light of these issues and challenges, we propose a self-supervised representation learning framework—SelfLinKG—to link concepts between OAG and Wikipedia. The core idea is to leverage self-supervised contrastive learning [12] to learn the intrinsic relations between different parts of the data. SelfLinKG compromises two key components: local attention-based encoding and global momentum contrastive learning.

The local attention-based encoding, which is based on graph attention networks [35], focuses on incorporating heterogeneous information within a single graph, including neighborhood context and hierarchy context. The global momentum contrastive learning aims at teaching the encoder to learn shared critical features across multiple graphs without labels. With the instance discrimination as pre-training task and contrastive loss, the designed encoder can learn features that distinguish ambiguous concepts from the target concept in an self-supervised manner. With the shared encoder mechanism, the representations are forced to be effective across the two graphs. With the momentum update, the fluctuating training of self-supervised is therefore stablized.

To summarize, our work makes the following contributions:

- First, we propose to study the concept linking problem across multiple large-scale knowledge bases, specifically between the large-scale OAG and Wikipedia. The challenges of this problem are identified: various relations, ambiguous concepts, scarce label data, and scalability.
- Second, to address them, we present a self-supervised embedding learning framework for concept linking–SelfLinKG—-which leverages state-of-the-art deep learning techniques for learning semantic and structural representations, even competitive against its supervised counterpart.
- Third, we conduct extensive experiments on OAG and Wikipedia, which suggest that SelfLinKG can achieve very high accuracy of 97.33% in the real application, significantly outperforming baseline models.
- Finally, together with the linking results, we make the Open Academic Graph with Knowledge (OAG_{know}) publicly available. OAG_{know} consists of 93 million concepts, which can be used for various research problems such as text mining, question answering, and knowledge reasoning.

2 The Concept Linking Problem

In this section, we formalize the problem of concept linking across knowledge bases and present the task of linking academic graphs with Wikipedia.

Knowledge bases (KBs) comprise both structured and unstructured information. Broadly speaking, it could be a taxonomy, an encyclopedia, or a knowledge graph. Formally,

Definition 1. Knowledge Base (KB)
A KB is defined as a graph $G = \{C, R, A\}$*, where the concept* $c \in C$ *contains semantic attributes and the relation* $r \in R$ *is associated with its type mapping functions* $\tau(r) : R \to D$ *with* $|D| > 1$*, where* D *is the set of relation types (e.g.* $D = \{\text{hypernym, related}\}$*), and* $A = \{c_i, r_{ij}, c_j\}$ *is the adjacency set recording the connectivity between concepts.*

For example, Wikipedia is a KB comprising of concepts (entries), and its relation set includes 1) the *hypernym* relations between concepts with a broad meaning and those with more specific meaning and 2) the *related* relations between concepts sharing weaker association. Compared with Wikipedia, taxonomies usually contain only the *hypernym* relations.

Definition 2. Concept Linking across KBs
Given m *knowledge bases* $KB_p (p = 1, \ldots, m;$ *with* $m > 1)$*, the goal is to generate concept linkings* $L = \{(c_i^{(p)}, c_j^{(q)}) | c_i^{(p)} \in KB_p, c_j^{(q)} \in KB_q, p \neq q\}$ *such that* $c_i^{(p)}$ *and* $c_j^{(q)}$ *represent exactly the same concept in* KB_p *and* KB_q*.*

In this work, we focus on the problem of concept linking between two public knowledge bases—the Open Academic Graph (OAG) [44] and the English Wikipedia. OAG is to date the largest publicly available academic graph. It contains five types of entities, including papers, authors, affiliations, venues (journals and conferences), and research concepts (topics). The goal here is to link OAG's concepts with Wikipedia's concept entries by using 1) OAG's concept taxonomy and concept content, and 2) Wikipedia's content and entry taxonomy.

The problem is challenging, as it is difficult to acquire sufficiently labeled data to train an effective machine learning model. To deal with this issue, we desire to have a powerful unsupervised or semi-supervised model. Second, there is also the name disambiguation issue. For example, there are four different "entropy" entries in Wikipedia, and how to link different ones with those in academic graphs is challenging. Finally, the model needs to handle the scalability, as to train and deploy a model to deal with thousands of millions of concepts is not an easy task.

To deal with the aforementioned issues, especially the first one, we further clarify the supervised and unsupervised (or self-supervised) embedding learning for concept linking in the following definition.

Definition 3. Embedding Learning for Concept Linking Given m knowledge bases represented as m graphs $\mathcal{G}_p = \{C_p, R_p, A_p\}(p = 1, \ldots, m)$, a embedding function $f : c_i^{(p)} | G \rightarrow R^d$ is learned such that for each concept $c_i^{(p)} \in C_p$, embedding $v_i^{(p)} = f(c_i^{(p)} | G_p)$ could be efficiently utilized to recover the full concept linkings $\mathcal{L} = \{(c_i^{(p)}, c_j^{(q)}) | c_i^{(p)} \in C_p, c_j^{(q)} \in C_q, p \neq q\}$ in:

1) Supervised setting: part of $\mathcal{L}$ is provided as the training set for training f.
2) Unsupervised (Self-supervised) setting: none of $\mathcal{L}$ is provided for training f.

3 The SelfLinKG Framework

In this section, we present the self-supervised embedding learning framework—SelfLinKG—for linking concepts across knowledge bases. We will first discuss the motivation of SelfLinKG and then introduce its two components.

3.1 Motivation

In related fields of concept linking, such as entity alignment, embedding-based methods are generally based on supervised learning. Supervised learning has achieved great success in the last decade, but it suffers from heavy dependency on manual labels and poor scalability on unseen data. These problems are especially fatal to large-scale concept linking and entity alignment. A large amount of manually labeled data is too expensive, and to make the linking system online, we need to make the algorithm scalable.

Despite the drawbacks of supervised learning, however, previously people have few choices but to choose it because of two important reasons as shown in Figure 2:

1) Lack of embedding consistency. For concepts in different KBs, their representations are located in different and inconsistent embedding spaces (just like two people using two languages). To make their embeddings consistent, we can either use a supervised classifier to bridge the gap (a translator [16]) or let them fall into the same embedding space by anchor nodes (both turn to use the third language [2], [18], [31], [45]). Both methods require external supervision.
2) Lack of training objective. In supervised learning, labels serve as objectives for encoders to draw near positive samples and push away negative samples. Without labels, such a goal seems to be impossible because we can not draw near positive pairs.

Fig. 2: Motivation of SelfLinKG from perspectives of embedding consistency and training objective.

Are there any means to cope with these problems, or part of them, without labels? Fortunately, recent breakthroughs in self-supervised learning shed light on this question.

In terms of embedding consistency, if KBs are in the same language, we can leverage the inherent embedding space of it. Instead of using word embeddings trained separately on different KBs, pre-trained language models such as BERT can provide a unified initial embedding space for concepts from different KBs. During the training, a shared encoder that yields embeddings for concepts from different KBs will further ensure the consistency.

In terms of training objectives, without labels, we cannot draw near positive sample pairs. However, there are always abundant negative samples. If we can push away negative samples from each other as much as possible, it equals we relatively draw near positive ones that share similarity to some extent. The instance discrimination pretext task with contrastive losses are born for that purpose.

To sum up, we propose SelfLinKG, a concept learning framework to deal with the large-scale heterogeneous concept linking problem without an arduously expensive process for producing massive labeled data. We propose to leverage self-supervised learning to learn the intrinsic relations between concepts across the two knowledge bases, which also help mitigate the scalability issue for handling large-scale data. In the following sections, we will introduce two components that SelfLinKG comprises of in details: 1) local attention-based encoding and 2) global momentum contrastive learning. Figure 3 illustrates the architecture of SelfLinKG.

Local Attention-based Encoding. The local attention-based encoding aims to tackle the data heterogeneity and map both data into the same latent space at both entity-level and graph-level. For entity-level, both semantic information and structural information are involved. We design a heterogeneous graph-attention-based encoder to aggregate information from the taxonomy structures (both hierarchy and neighborhood). For graph-level, we formulate taxonomies, encyclopedias, and knowledge graphs into unified attributed graphs with two types of relations (hyponym and related) to simplify the problem.

Global Momentum Contrastive Learning. After encoding concepts' into vectors in the first step, we propose to use

a self-supervised representation learning solution to link concepts across the two datasets. This solution requires a proper self-supervised training task and objective function. Note that under this setting, there is no existing link across the graphs. Inspired by recent progress on self-supervised contrastive learning in computer vision [12], we propose to use instance discrimination as the pre-trained task and leverage momentum contrastive learning as the objective. We use an online encoder (updated by direct gradient descends) and a target encoder (updated by momentum), which are trained across both graphs, to ensure the training stability and that embeddings are projected into the same embedding space.

3.2 Local Attention-based Encoding

We introduce the local attention-based encoder, which aims to efficiently capture semantic and structural information by learning to aggregate information from concepts. The encoder comprises three components: semantic embedding generation, neighborhood aggregation, and hierarchy aggregation.

Semantic embedding generation. For a concept in the knowledge base, we embed its semantic information into the latent space. Traditionally, we can initialize embeddings randomly or train phrase-level embeddings on each KB using methods like doc2vec [13]. However, when it comes to large-scale applications across multiple knowledge bases, such methods are computationally expensive and can not easily scale up to concepts absent in training corpus. Additionally, semantic embeddings trained separately in each KB will spoil the embedding consistency we have discussed in Section 3.1 that is critical for a self-supervised setting.

Therefore, we adopt the recent pre-trained language model BERT [5], [41] as the basic encoder, which is pre-trained over vast amounts of corpora (including whole Wikipedia) and has been proved to be effective for almost all the language tasks. With the help of BERT, concepts with similar semantics in different KBs will still be embedded into similar vectors and maintain the embedding consistency. What is more, since many concepts are professional terms, the conventional embedding methods such as GloVe fails to cover them. But BERT's sub-token technique, which split an unknown token into known sub-tokens, can preserve the semantics in these terms.

For most concepts, such as in many taxonomies, their names are the only semantic information we can utilize, and the semantic embedding $\mathbf{h}_i$ is generated by: $\mathbf{h}_i = \text{encoder}(\text{name}_i)$. When there are attributes available, we can extend the representation by using aggregators like pooling or soft attention mechanisms [31].

In detail, for a sequence to encode, we turn it into lowercase with the max sentence length as 25 with padding for shorter ones. Notice that we allow for special tokens in BERT tokenizer. Since BERT is a bidirectional transformer, the encoded output is also a sequence with the same length as the original sequence. To transform this sequence to a fixed-length vector as sequence embedding, follow tradition, we apply the average pooling from the second output to the final output. The first output is the embedding of the special token [CLS] in BERT, meaning start of the sequence, which is usually used for natural language understanding task such as text classification. In our setting it is discarded because we want embeddings that represent token-level information rather than sentence-level. The BERT output units have a dimension of 768, to transform into a given dimension for other networks, we apply the max pooling and reduce the dimension to 150.

Neighborhood aggregation. In real-world applications, raw semantic information is usually minimal and does not contain sufficient information for high-quality links. Therefore, structural contexts are often utilized to generate high quality and distinguishable embeddings. To put correct weights on a concept's neighbors, we propose to use the multi-head graph attention networks [35] with trainable unique embeddings.

Given a concept $c_i^{(p)}$ in knowledge base KB_p (for short, we use c_i), the goal of neighbor aggregation is to learn the attention coefficient $\text{attn}(c_i, c_j)$, which implies the aggregation weight of a concept c_j's influence on target concept c_i. The attention coefficient is learned by the self-attention mechanism:

$$o_{ij} = \text{attn}(\mathbf{W}\mathbf{h}_i; \mathbf{W}\mathbf{h}_j) \tag{1}$$

where o_{ij} indicates the importance of concept c_j's features to concept c_i, $\mathbf{h}_i$ is concept c_i's semantic embedding, and $\mathbf{W}$ is a shared projection matrix. By utilizing the neighborhood structure, the graph attention layer only needs to compute o_{ij} for concepts c_j that have the related relations with c_i, i.e. concepts $c_j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is the neighborhood of node c_i. Then o_{ij} can be normalized across all possible c_j by using the softmax function

$$_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{W}\mathbf{h}_i + \mathbf{W}\mathbf{h}_j))}{\sum_{k \in \mathcal{N}_i} \exp(\text{LeakyReLU}(\mathbf{W}\mathbf{h}_i + \mathbf{W}\mathbf{h}_k))} \tag{2}$$

Then we employ the multi-head attention to generate concept c_i's output embedding $\mathbf{h}_i^0$. Here we continue to use $c_i^{(p)}$ and $\mathbf{h}_i^{(p)0}$ to emphasize that they only involve information from KB_p:

$$\mathbf{h}_i^{(p)} = \text{fc}\Big(\Big\|_{k=1}^{K} \Big(\sum_{j \in \mathcal{N}_i} {}_{ij}^{k} \mathbf{W}_h^k \mathbf{h}_j^{(p)}\Big)\Big) \tag{3}$$

where $\|$ represents the concatenation operation, $\text{fc}(\cdot)$ is fully-connected neural networks, f is the activation function, $\mathbf{W}_h$ denotes specified projection matrix for semantic embedding, $\mathcal{N}_i$ is the neighborhood of node c_i, and K is the head number.

In practice, consider the average degree of 2.56 in MAG and 22.24 in EnWiki (see details in Section 4.1), we sampling a fixed number of 20 neighbor nodes as the $\mathcal{N}_i$ for concept c_i and employ a 2-hop multi-head attention networks to encode the embedding. Since the $|\mathcal{N}_i|$ is fixed, when the 1-hop neighbors of c_i are not enough, we will continue to sample 2-hop and even k-hop neighbors if needed.

The reason for using multi-head attentions are as follow. Analogous to different kernels and channels in convolutional neural networks, different attention heads in attention networks captures different patterns, while a single attention head could only captures one pattern and leads to unstable and poor performance, which has been proved in various work [3], [34], [35], [37].

Fig. 3: An illustrative architecture of SelfLinKG framework. Local Attention-based Encoding, i.e. the encoder, encodes neighbor and hierarchical information into vector embedding. Global Momentum Contrastive Learning leverages the output embedding to calculate contrastive loss using self-supervision rather than labeled data. Online encoder f uses momentum to update the target encoder f_t.

Up until now, we follow the traditional setting of graph attention networks. However, this will lead to several problems:

1) Semantic embedding is static during the training, but it should be trainable to adapt to better representation.
2) Ambiguous concepts (i.e., with similar or the same name) could not be well distinguished.
3) Under heterogeneous multi-graph settings, synonym concepts should be linked, but they actually have very different semantic embedding.

To solve the tough problems above, beyond semantic embedding, we introduce unique embedding. For each concept $c_i^{(p)}$, we create a unique trainable embedding u_i. In the self-supervised mode, every concept has its own unique embedding. Therefore, besides W_h we create another set of projection matrix W_u for aggregating the unique embedding as

$$u_i^0 = fc\Big(\Big\|_{k=1}^{h}\Big(\sum_{j \in N_i} {}^{k}_{ij}W_u^k u_j^0\Big)\Big) \tag{4}$$

and the final output of neighborhood aggregation is to concatenate the aggregation results of semantic embedding and unique embedding and pass to a fully-connected layers

$$v_i^{(p)} = fc([h_i^{(p)} \| u_i^0]) \tag{5}$$

The auxiliary unique embedding successfully solves the problems above. First, unique embedding could now be trained to provide concepts with dynamic adaption. Second, ambiguous concepts could gain a distinct representation through differentiate unique embedding of themselves while not changing the original semantic embedding. Finally, synonym concepts could narrow the gap of their representation by having a shared or similar unique embedding.

Hierarchy aggregation. In this part, we introduce the idea of extending the hierarchical context of a concept using breadth-first-search (BFS) and applying a similar graph attention mechanism in neighborhood aggregation to perform the hierarchical aggregation.

First, we define the hierarchical context to be the hypernym (parent concepts) context rather than the hyponym (children concepts) because children concepts are usually noisy and excessive. In practice, we view children concepts as ordinary related concepts, and encode them in the neighborhood aggregation step.

The intuition for extending hierarchical context is that hierarchical structure in different knowledge bases is heterogeneous. For instance, in MAG the concept Machine learning has the parent concept Computer science. However, in EnWiki this structure is more elaborate and fine-grained: Machine learning has the parent Artificial Intelligence, and Artificial Intelligence has the parent Computer science. If we only consider direct hypernyms of the concept, we will not be able to deal with the heterogeneous problem that widely exists in real datasets.

A naive way is to use the top-down path from the root concept to the target concept. However, for most knowledge bases, the hierarchy is not organized as a tree, but a directed acyclic graph (DAG); and a concept could possibly have more than one direct parent. To solve the problem, we propose to extend the hierarchical context by bottom-up breadth-first-search (BFS).

Given d as number of samples, we perform BFS to sample d parent concepts from target concept $c_i^{(p)}$ in KB_p and generate the subgraph $G_i = \{E_i; R_i\}$ where $E_i = \{e_j | j = 1; 2; :::; d\}$ is the set of parent concepts and $R_i = \{(c_i^{(p)}; e_j) | j = 1; 2; :::; d\} \cup \{(e_j; e_k) | e_k$ is the parent of $e_j\}$. In this subgraph, we perform graph attention aggregation to generate the hierarchical representation as

$$_j = \frac{\exp(\text{LeakyReLU}(W_h(c_i^{(p)}) + W_h(e_j)))}{\sum_{k \in E_i} \exp(\text{LeakyReLU}(W_h(c_i^{(p)}) + W_h(e_k)))} \tag{6}$$

$$m_i^{(p)} = fc(\sum_{k=1}^{k_n}(\sum_{j \in E_i} {}^{k}_{j} W_m^k h(e_j))) \quad (7)$$

where $m_i^{(p)}$ is the hierarchical representation, $h(\ )$ denotes semantic embedding, W_m is the projection matrix for hierarchy aggregation. In practice, we sampling a xed number of 5 hypernym nodes as the E_i for concept c_i and employ a 2-hop multi-head attention networks to encode the embedding. Similarly, because hypernym relations are very sparse, 5 hypernym nodes could usually cover 2-hop and even 3-hop hypernym nodes for concept c_i (see Section 4.1).

In all, we nally get the overall representation $v_i^{(p)}$ for concept $c_i^{(p)}$ by integrating all these representation up as

$$v_i^{(p)} = fc([h_i^{(p)} k v_i^{(p)} k m_i^{(p)}]) \quad (8)$$

where $h_i^{(p)}$ denotes the original semantic embedding, $v_i^{(p)}$ denotes the neighborhood aggregation result and $m_i^{(p)}$ denotes the hierarchy aggregation result.

3.3 Global Momentum Contrastive Learning

In this section, we present the self-supervised global momentum contrastive learning framework with three key ideas: shared encoder, contrastive loss as instance discrimination, and momentum update.

Shared encoder As we have discussed in Section 3.1, during the shift from supervised embedding learning to self-supervised embedding learning, lack of embedding consistency and training objectives are the main obstacles. For the consistency problem, BERT embeddings have provided us with a uni ed initial embedding space for concepts from different KBs. However, if we train encoders for each KB respectively, each of these encoders will learn their own parameters and therefore spoil the consistency.

For ensuring consistency during training, a natural idea is to "merge" KBs into one. This is what happens to the shared encoder, that it is jointly trained over KBs to maintain the uni ed embedding space because it only learns one set of parameters. Our ablation study shows that the shared encoder mechanism brings in a noticeable improvement.

Contrastive loss as instance discrimination Recent studies show promising results on self-supervised representation learning [17] using contrastive loss [9] in computer vision [12], [30], [42], natural language processing [4] and graph learning [36]. Given a positive sample pair $(x; y)$ and a set of negative samples $Y = y_1; :::; y_k$, the contrastive loss is formulated as:

$$\begin{aligned} L_{\text{contrast}} &= E[\ \log \frac{e^{f_x^T f_{y^+}}}{e^{f_x^T f_{y^+}} + \sum_i e^{f_x^T f_{y_i^-}}}] \\ &= \underbrace{E[\ f_x^T f_{y^+}]}_{\text{alignment}} + \underbrace{E[\log(e^{f_x^T f_{y^+}} + \sum_i e^{f_x^T f_{y_i^-}})]}_{\text{uniformity}} \end{aligned} \quad (9)$$

where $f(\ )$ refers to the encoder and is the temperature hyperparameter. The rst term aims at "alignment" and the second aims at "uniformity" of sample vectors on a sphere given the normalization condition.

The traditional supervised entity alignment algorithms can also be categorized into contrastive learning with manually labeled positive pairs. Based on the assumption that after encoder f converges, a concept x from KB_1 and its positive paired y from KB_2 should have similar embeddings ($P(f_x = f_y) = 1$). If we always normalize the representation ($\|f_x\| = 1$), according to [38], we have $f_x^T f_y$ 1 and the contrastive loss can be further written as:

$$L_{\text{contrast}} \xrightarrow{P(f_x = f_y)=1} 1 = + E[\log(e^{1=} + \sum_i e^{f_x^T f_{y_i^-} =})] \quad (10)$$

which indicates that the main problem of embedding-based concept linking lies in uniformity rather than alignment given the prerequisite. This provides us with a solid theoretical foundation for applying self-supervised learning to concept linking problem, that if we can guarantee $P(f_x = f_y) = 1$ during the training, we can still learn a good representation for concept linking without manually labeled positive samples.

This condition does approximately hold during the training of SelfLinKG. Because we have already uni ed the embedding space of different KBs, suppose x from KB_1 and y from KB_2 are identical concepts, they must share some semantic and structural similarities (otherwise no machine learning algorithm will be able to link them). From the perspective of representation, f_x and f_y are naturally close to each other in the embedding space. In SelfLinKG, we only push other noisy concepts away from them as far as possible to make their distance relatively small and do not try to draw them near. Of course, the performance will not be as good as supervised learning with abundant labels, but we will show in the experiments that our self-supervised SelfLinKG is competitive and even better than supervised ones in few-label situations.

Internal Deduplication Assumption (IDA) for negative sampling In practice, we propose to leverage contrastive loss as instance discrimination to train our encoder. For each concept x, we use the initial BERT embeddings to select out a set of top-k similar concepts y_i as hard negative samples from the KB that contains x rather than the KB we want to link x to, and then maximize the similarity of $f_x = f(x)$ encoded by the online encoder f with $f_y = f_t(x)$ encoded by the target encoder f_t and minimize the similarity of f_x with f_{y_i}.

This strategy works under the assumption that a KB is internally deduplicated, which we call "Internal Deduplication Assumption (IDA)". In fact, a KB usually does not contains two distinct concepts that have identical meaning. In that case, we can view any other concepts in the KB as negative ones to a given target x, and therefore would not select out true positive samples even without groundtruth because we are selecting negative samples from KB that the concept x comes from rather than the counter KB. Compared with Local Closed World Assumption [6] which assumes triplets not in a set of KBs as negative samples, our assumption is made within a deduplicated KB and assumes all concepts beyond the target itself could play the role as negative samples.

Algorithm 1: Global Momentum Contrastive Learning

Input : Knowledge Base $KB_p (p = 0; 1; :::; P)$; Online encoder f and params $\ $; Target encoder $f_t(\ )$ and params $_t$.
Output: Pretrained embedding v for linking task.
Initialize $\ = \ _t$.
while True **do**
  Randomly pick a KB_p from all KBs and a batch X from KB_p;
  for concept $c_i \ 2 \ X$ **do**
    $f_x = f(c_i)$;
    /* 1. instance discrimination */
    $f_y = f_t(c_i)$;
    sample $Y = \{y_1; :::; y_k\}$ from KB_p;
    $f_{y_i} = f_t(y_i)$;
    $L = \ f_x^T f_y = + \log(e^{f_x^T f_y = } + \sum_i e^{f_x^T f_{y_i} = })$
    /* 2. gradient update on f */
    $\ = \ \frac{@}{@ } L$
    /* 3. momentum update on f_t */
    $_t = m \ _t + (1 \ \ m)$

Fig. 4: Conceptual comparison of (a) end-to-end training and (b) momentum update. End-to-End use single encoder, which suffers from training instability and could not converge in the experiment. However, momentum update helps the target encoder to change steadily along the direction of gradient momentum, rather than that of the instant gradient.

There may be another remaining question about why we use hard negative samples. As indicated Equation (10), the more negative samples rather than harder ones, the better uniformity. That is true for previous applications of contrastive learning that focus on image classification problems who leverage a memory bank or a queue to store a large amount of randomly selected negative samples ([12], [30], [42]). However, in concept linking, we find that since the semantic embeddings have been pre-trained and are quite distinctive in most cases, using random negative samples from memory banks or queue structures lead to trivial solutions in preliminary experiments. Thanks to the pre-training, trivial negative samples actually have been distributed far away from our target, and the main problem lies in those harder ones. Thus, we perform negative sampling based on semantic embedding similarity, i.e., we collect concepts that are similar to the target concept by name as negative sampled ones.

Momentum Update. In this part, we will introduce the final critical idea that ensures the feasibility of SelfLinKG—momentum update. Recall Equation (10), given a concept x and a set of negative samples y_i , we will update the encoder f by minimizing $L_{contrast}$. However, a question is that are we going to update f according to f_x, f_{y_i} , or both of them?

Traditionally, in the end-to-end training fashion, we will update f from both f_x and f_{y_i} . However, in the contrastive learning scenario, we only update f according to f_x and stop gradients being back-propagated to f from f_{y_i} in a momentum update paradigm. A conceptual comparison between them is shown in Figure 4. The reason is that f_{y_i} here serves as a stable ground for f_x to adjust its location in the embedding space. If we directly update them according to the direction of the instant gradient, so rapid are the uctuations of f_{y_i} at the beginning of the training that even x in adjacent batches may see drastically different embedding of the same negative sample y_i , leading to representation collapse [12]. Instead, as many famous optimization algorithms such as Adam and Adagrad do, we can update them according to the direction of the gradient momentum, which is far more steady. In Section 4.4, our experiment also support this conclusion.

On the other hand, the f_{y_i} should also be updated along with the learning of f . Thus, a compromise is to leverage momentum update with two encoders: the online encoder f and the target encoder f_t. Such techniques are also common in fields such as reinforcement learning, where Double Q-learning [33] with two encoders are proposed to deal with the learning collapse and instability. In this case, we can rewrite the contrastive loss as

$$\begin{aligned} L_{contrast} \ &= E[\ f(x)^T f_t(x) = \ ] \\ &+ E[e^{f(x)^T f_t(y) = } + \sum_i e^{f(x)^T f_t(y_i) = }] \end{aligned} \tag{11}$$

where the online encoder f is directly updated by gradients as

$$\frac{@}{@ } L_{contrast} \tag{12}$$

where is the learning rate. And the target encoder, by the momentum it is updated for every certain number of steps as

$$_t \quad m \ _t + (1 \quad m) \tag{13}$$

where $m \ 2 \ [0; 1)$ is the momentum coefficient that needs to be set as a large value like 0.999 to ensure smooth optimization. In this case, the representations of negative samples y_i are slowly updated following the rapidly updated online encoder. Our ablation study in Section 4.4 also shows a relatively bigger momentum value (i.e., a slower update to target encoder) could lead to better results.

3.4 SelfLinKG in the Supervised Setting

To demonstrate SelfLinKG's characteristic of label efficiency, in this section, we will briefly describe SelfLinKG in the supervised setting, which is used for ablation study in the experiment. We use $SelfLinKG_s$ to stand for supervised SelfLinKG.

In the supervised setting, i.e., there are groudntruth cross links $L_{train} = \{(x, y)\}$ where $x \in KB_1, y \in KB_2$ provided, according to Equation (9), we leverage the most ordinary form of contrastive loss for the concept linking task with supervision. We still preserve the shared encoder mechanism and momentum update to make a fair comparison. For the local attention-based encoding, those linked pairs also share identical unique embeddings. In the experiment, we adjust the ratio $|L_{train}|/|L|$ of cross links for training SelfLinKG to compare with SelfLinKG with no L_{train} provided.

We discuss the experiment results between SelfLinKG and $SelfLinKG_s$ in Section 4.4. The results show that SelfLinKG has a competitive and even much better performance in situations with fewer labels.

4 Experiment

Because SelfLinKG is designed to cope with concepts in real-world large-scale heterogeneous knowledge bases, conventional entity alignment benchmarks like DBP15k [27] are either too small, with little ambiguity, or cross-lingual (which must require external supervision), which fail to meet our requirements. Therefore, in this work, we decide to evaluate the SelfLinKG between two large heterogeneous knowledge bases: Microsoft Academic Graph (MAG) taxonomy ([25], [26]) in OAG [44] and English Wikipedia (EnWiki).

Based on experiments, we further conduct concept linking between 14 different knowledge bases. The final linked concept graph OAG_{know} is publicly available.

4.1 Dataset

MAG taxonomy. MAG taxonomy consists of 679,921 concepts collected from the Internet and 873,087 hypernym relations generated according to co-occurrence from 208,915,369 published papers. As some concepts in MAG are isolated — have no relations with the other concepts, we filter them out. Finally, the resultant MAG graph consists of 490,885 concepts and 873,087 hypernym relations.

English Wikipedia (EnWiki). As for EnWiki, we use the snapshot of June 2019. After cleaning, we have 7,598,399 terms and 1,584,269 categories, with 7,029,440 related relations and 1,395,321 hypernym relations. We apply three rounds of cleaning to EnWiki: First, since there are too many entities rather than concepts in EnWiki, we use concepts in the MAG subgraph to search top-10 similar entries in EnWiki by name as seeds. And then, we extend all concepts that are 1) hypernyms of seeds, 2) has the same name with seeds into the subgraph. Finally, because some categories share the exact same name with concepts, we merge them into one. Notice that there are no original hypernym relations in Wikipedia, so we view pages-in-category and subcategory as hypernym relations.. Finally, the obtained EnWiki graph consists of 620,557 concepts, 5,503,012 related relations and 1,395,321 hypernym relations.

Groundtruth. As for evaluation, because MAG taxonomy is collected from the Internet, the original web page URLs for concepts are available. After examination, 233,010 concepts of the MAG taxonomy derive from EnWiki, in which some of the URLs are redirected to other new entries because the EnWiki is also evolving. We evaluate SelfLinKG and other baseline models based on these original links as groundtruth.

4.2 Setup

Evaluation Tasks. To systematically evaluate the proposed methodology, we design the following four tasks:

Synonym Linking: We utilize the redirect link in EnWiki to build a dataset consisting of 10,082 sample pairs, among which 5,041 are synonym concept pairs, and others are negative sample pairs by sampling similar terms. Following [31], we put the pair of embedding into a multi-layer neural network as a classifier to output the similarity score.

Disambiguation: Disambiguation of concepts sharing the exact same name is extremely difficult for linking. The ambiguous concepts are mainly from EnWiki, with a correct matching concept in MAG. We pick out ambiguous concepts by disambiguation pages in EnWiki and construct a dataset containing 730 unique terms and 3,548 concepts. Statistics of the disambiguation dataset are shown in Table 2. For each unique term, we re-rank concepts by L2 distance using trained representation. Because in the disambiguation task, the negative samples should only involve those ambiguous ones, Hit@K would be a more objective metric rather than Prec./Rec./F1 in which case negative samples are randomly selected and lead to a virtual-high result.

General Linking: We build a challenging dataset contains both simple matching cases and synonym cases, altogether 20,100 samples, including 70% simple matching cases and 30% synonym concepts cases and harden it by sampling negative concepts that have similar semantic embedding with positive pairs. The dataset has 60% for training, 20% for validation, and 20% for testing. Following previous works, we feed pairs of embeddings into a multi-layer neural network as a classifier to output the similarity score.

General Ranking: We further construct a ranking dataset using positive pairs from the General Linking dataset. For each positive sample, we find the top-20 similar concepts by name in EnWiki as candidates, and re-ranking them by L2 distance using trained representation and faiss toolkit.

Comparison Methods. Though many embedding-based entity alignment algorithms have emerged these years, most of them focus on supervised learning and are not fair to serve as baselines for self-supervised SelfLinKG. Therefore, we select a series of state-of-the-art unsupervised knowledge graph embedding methods that are used as baselines in various entity alignment papers to compare. We manually tune their hyperparameters for a solid comparison.

RESCAL [20]: This method is an approach to relational learning based on the factorization of a three-way tensor.

TransE [1]: A method that models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities.

ComplEx [32]: This is a simple approach to perform matrix and tensor factorization for link prediction data

TABLE 1: Results of linking performances under unsupervised settings.

Task & Metrics		RESCAL	TransE	ComplEx	HolE	DistMult	GAKE	SelfLinKG
Synonym Linking	Prec.	54.92	55.94	56.44	56.45	56.15	46.86	67.85
	Rec.	25.51	56.91	35.16	23.58	32.01	40.96	85.77
	F1	34.84	56.42	43.33	33.26	40.78	43.71	75.76
	AUROC	54.44	60.10	56.96	54.50	56.41	49.86	80.64
Disam-biguation	Hit@1	21.78	22.05	34.10	19.58	36.43	38.35	48.21
	Hit@2	58.90	57.94	63.69	56.98	67.26	64.65	70.00
	Hit@3	75.20	76.71	80.00	76.30	80.41	80.41	82.73
	Hit@5	91.09	92.19	92.32	93.15	91.64	92.32	93.28
General Linking	Prec.	54.36	58.67	53.36	56.26	55.25	52.75	75.85
	Rec.	51.81	59.53	62.95	35.58	53.27	57.72	76.15
	F1	53.05	59.10	57.76	43.59	54.24	55.12	76.00
	AUROC	54.09	59.79	54.55	54.86	56.47	53.04	80.75

#Candidates	2	3	4	5	6	> 7
Portion(%)	31.4	20.0	16.0	10.4	9.7	12.5

TABLE 2: Statistics for Disambiguation Task Dataset

that uses vectors with complex values and retains the mathematical de nition of the dot product.

HolE [19]: Holographic embeddings (HOLE) is a method to learn compositional vector space representations of entire knowledge graphs. It is related to holographic models of associative memory in that it employs the circular correlation to create compositional representations.

DistMult [43] : This method focuses on the study of neural-embedding models, where the representations are learned using neural networks with energy-based objectives.

GAKE [8] : This method formulates the knowledge base as a directed graph, and learns representations for any vertices or edges by leveraging the graph's structural information. In this method, three types of graph context for embedding are introduced: neighbor context, path context, and edge context; each re ects properties of knowledge from different perspectives.

SelfLinKG : self-supervised SelfLinKG. In this method, we input the subgraph of a concept, and trained the shared encoder across MAG and EnWiki with only the instance discrimination pre-train task, which is an unsupervised method. The unique embedding has no sharing in these settings, i.e., every single concept has its unique embedding.

We utilize OpenKE [10], an Open-source Framework for Knowledge Embedding organized by THUNLP based on the TensorFlow toolkit. The authors use C++ to implement some underlying operations, such as data preprocessing and negative sampling. For each speci c model, it is implemented by TensorFlow with Python interfaces so that there is a convenient platform to run models on GPUs. For GAKE, we download the authors' source codes written in C++ to perform training.

Environment and Settings. In the experiment, the input dimension and hidden dimension for graph attention layers are 150; attention dropout is 0.3; the number of attention head is 4. The input dimension of the fully-connected-layer in the encoder is set to 600 and output a vector with a dimension of 150.

For the hyperparameters of global momentum contrastive learning, we set the momentum value m to 0.999, temperature to 10. The networks are optimized with Adam optimizer. All codes are implemented in Python3 and run by interpreter Python3.6. The experiments were conducted on a CentOS server with a 14 cores Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz, 640G System RAM, and a Tesla V100-SXM2 32GB RAM GPU.

An important technique in our method is negative sampling based on name similarity. Here we apply the Faiss, a library for ef cient similarity search [1]. In the candidate searching period, we apply the IndexFlatL2 as an indexer based on L2 distance. The search was previously conducted respectively in the MAG subgraph and EnWiki subgraph. It is quite ef cient because once the indices are built by KD-tree, the closest neighbor can be easily found.

4.3 Results of Unsupervised Linking and SelfLinKG

Table 1 shows the overall linking performance on three tasks by embedding the training method: Synonym Linking, Disambiguation, and General Linking. The results of the General Ranking task are showed in Figure 5. All the methods compared are unsupervised or self-supervised embedding learning methods. Results show that our method SelfLinKG consistently outperforms other alternatives (26%-33%) in every task. We will discuss and compare results on each of them.

Synonym Linking. For Synonym Linking, the names of a positive concept pair are not identical, sometimes even very different from each other literally. SelfLinKG performs the best among all methods with high recall, F1, and AUROC, which means that SelfLinKG only omits a little proportion of positive pairs, even for those with entirely different names. Its comparatively low precision is probably because it assumes some negative concept pairs with similar names as correct. TransE performs comparatively well, while other methods generally have low F1 scores around 30%-40%.

Disambiguation. In the Disambiguation task, SelfLinKG also signi cantly performs better than other methods, which means SelfLinKG has learned a highly distinguishable representation that can discriminate ambiguous concepts. GAKE also achieves superior performance to other baseline methods, probably because context plays a vital role in its

1. https://github.com/facebookresearch/faiss

Fig. 5: Ablation study on General Ranking. The SelfLinKG $_s$ here is trained under the best training ratio 0.8.

Fig. 6: Ablation study on 1) SelfLinKG v.s. SelfLinKG $_s$, 2) shared encoder v.s. separate encoder

training. Moreover, it is also the structural and hierarchical context information that can help to disambiguate similar concepts. [31] also observes such insight.

General Linking. For General Linking, compared with Synonym Linking, it is easier because we add in many simple cases. However, hardened negative samples can still cause trouble. SelfLinKG still performs the best, and other baselines' performance also increases. TransE is still the top method among baselines, but other methods also have competitive performance compared to TransE. SelfLinKG shows a more balanced precision and recall in this task, demonstrating its ability to cope with linking problems in most cases.

General Ranking. Figure 5 display the result on General Ranking, which takes the top-20 similar concepts by name as negative samples for each positive pair from General Linking and re-rank them using trained representation by L2 distance. We only show the top-5 ranking results because the number of candidates for each concept is only 20. Noted that the SelfLinKG$_s$ here is trained under the training ratio 0.8 (see training ratio's detailed de nition in Section 4.4).

The sub gure (a) compares the performance on variants of SelfLinKG. It is surprising for us to see that in this challenging setting, the SelfLinKG overwhelms other variants by 5%-9%, even using supervision. We speculate that this is because in the SelfLinKG$_s$, we only train concepts in the $|L_{\text{train}}|$, which takes up 80% of the whole $|L|$ and approximately 25% of the whole MAG concepts, leading to an insuf cient uniformity because only 25% of the concepts are trained. However, in SelfLinKG, because we have no limitation of the training set, every concepts are trained using the self-supervised objective which results in a better uniformity over the whole dataset (as theoretically demonstrated in Eq. 9). This implies that our self-supervised objective could even be a complementary for the supervised setting to achieve better uniformity, but due to the limited passage we will leave it as an open problem for the following work.

For other unsupervised methods in sub gure (b), because their objectives do not focus enough on the discrimination task enough, they also fail to perform well in the General Ranking task.

4.4 Detailed Ablation Study

In this section, we conduct ablation studies on three critical factors in SelfLinKG: self-supervision, shared encoder, and momentum update.

Self-supervised v.s. Supervised. To further study the capability of the proposed self-supervised model SelfLinKG for concept linking, we develop a supervised version SelfLinKG $_s$ and evaluate the gap between the two versions. For implementation details of SelfLinKG $_s$, please refer to Section 3.4. We compare them in the Disambiguation and General Linking tasks. Among 233,010 anchor links we have, we take out 22k of them as available training labels for supervised SelfLinKG$_s$. Noted that these 220k anchor links do not overlap with the 10k used for evaluating the General Linking task.

Figure 6 shows the results of the different versions of SelfLinKG on these two tasks. For SelfLinKG and SelfLinKG (separate), their performance is static to the ratio of training labels provided because they are in the self-supervised setting. On the contrary, supervised SelfLinKG $_s$'s performance bene ts from the increasing ratio of provided training labels.

We observe from both (a) and (b) that the self-supervised SelfLinKG is signi cantly better than the supervised SelfLinKG$_s$ when the ratio of the training data is less than 0.5 (about 110k labels). Only when the training data ratio increases to nearly 0.6, the supervised version can take advantage of massive labeled information. Noted that in practice, collecting such a large number of training data is always expensive and infeasible.

Moreover, for multiple knowledge bases linking, this cost is growing quadratically with the number of knowledge bases involved. Suppose there is$m$ KBs, under the supervised setting, we have to manually label cross-links for every two KBs, which results in $\frac{m(m-1)}{2}$ needed dataset. Our unsupervised model SelfLinKG suggests that the self-supervised method could also perform ideally while cutting down the labeling costs.

Shared encoder v.s. Separate Encoder. From both Figure 6 and Figure 5, we also observe that the shared encoding is very helpful. With the shared encoding, SelfLinKG can obtain an improvement of 5.5% by Hit@1 rate in the Disambiguation task. In the General Linking task, we can also obtain an improvement of 2% by AUROC. This veri es the importance of sharing parameters in multi-graph learning and cross embedding space learning. A consistent embedding space is critical for the performance of concept linking and entity alignment.

Momentum Update v.s. End-to-end. In Table 3 we discuss the gap between momentum update and end-to-end training. Besides, we also investigate how to choose momentum value.

TABLE 3: Ablation study on momentum value m

Momentum	Hit@1	F1	AUROC
end-to-end (0.0)	failed	failed	failed
0.9	failed	failed	failed
0.99	46.57	74.74	81.43
0.999	48.21	76.00	80.75
0.9999	46.57	77.33	83.41

TABLE 4: Ablation study on multi-head attention

Heads	Hit@1	F1	AUROC
4	48.21	76.00	80.75
1	47.13	73.31	78.87

As we discussed in Section 3.3, the single-encoder-architecture (end-to-end) in contrastive learning leads to instability naturally. This is because, in training, the target encoderserves as the ground-truth, i.e., the distribution we want the online encodeto t on. In the single-encoder-architecture, model parameters are updated drastically at the beginning of training, leading to a rapid changing ground-truth and a failure. The momentum update can successfully deal with the problem by changing the target encoder slowly and smoothly, following the average direction of gradient optimization rather than the instant gradient.

For comparison between momentum update and end-to-end training, the end-to-end training equals momentum update when $m = 0$. In this situation, we discover that end-to-end training leads to a rapid collapse. This holds even when m is only a bit small such as 0.9. Our experiment shows that the training still fails very quickly.

Besides failed scenarios, we test on the other three typical values of m: 0.99, 0.999, 0.9999. The experiment indicates that the value between 0.999-0.9999 performs well. For value bigger than 0.9999, the target encoder is updated too slowly and causes the performance to drop.

Multi-head Attention We conduct further experiments to show that why multi-head attention is necessary. As authors in Graph Attention Network (GAT) claimed [35], they found the multi-head attentionto be bene cial similar to ndings in transformers [34]. The intuition is that a certain type of attention head usually captures a certain data pattern, like kernels in convolutional neural networks. With more heads, more parameters are engaged in the training which often yield better results.

In the Table 4, we compare the performance of traditional single-head attention and our multi-head attention (i.e. 4-head attention) on Disambiguation and General Linking tasks. The results show that our 4-head attention outperforms single-head attention on every tasks. Although more heads may yield marginal bene ts, but it also brings in a heavier model, so in this work we only choose 4 heads for our local attention network.

5 OAG$_{know}$ — Linked Concept Graph

Based on our proposed framework and public datasets, we have published Open Academic Graph Knowledge (OAG$_{know}$)[1], which integrates concepts from 14 bilingual knowledge bases. Some datasets such as AMiner, AMiner-NSFC, and Termonline are published for the rst time.

In practice, rst, we use semantic embedding and fuzzy matching to generate a similar candidate pool. Then we feed the neighborhood and hierarchy of concepts into SelfLinKG, and self-supervisedly train their embeddings. Finally, we leverage the trained embeddings to do ambiguous ranking and classi cation on pairs from the candidate pool to get the linked KBs. The accuracy of links is 97.33% by random sampling a small subset of OAG$_{know}$ for evaluation.

Table 5 shows the basic statistics of the OAG$_{know}$. Concepts are linked to 0.7 billion OAG entities (authors, papers, venues, af liations) through MAG and AMiner. Together with 93 million concepts from various KBs, OAG $_{know}$ is the largest public academic knowledge graph to date.

TABLE 5: Overall Statistics of OAG$_{know}$. T Taxonomy ; E Encyclopedia; K Knowledge Graph

Name	Type	Language	#Concepts	#Cross-links
MAG [26]	T	En	679,921	963,579
AMiner [29]	T	En&Zh	367,890	33,890
AMiner-NSFC	T	Zh	53,017	85,689
NSF	T	En	2,155	810
Termonline	T	En&Zh	105,298	92,754
GB	T	Zh	3,543	2,697
Bpress	T	En	1,269	1,157
Xlore [39]	E	En&Zh	508,768	252,144
EnWiki	E	En	7,598,399	17,552,975
ZhWiki	E	Zh	1,055,757	703,737
Baidu	E	Zh	10,423,650	229,137
HuDong	E	Zh	3,141,658	367,850
Wikidata	K	En	22,574,520	17,918,258
Freebase	K	En	47,294,433	4,021,644
Overall	-	-	93,810,278	42,226,321

6 Related Work

Concept linking, closely related to entity linking, ontology alignment, schema matching and data integration etc., has long been studied for decades [7]. Many approaches have been proposed to address this problem. For example, Li et al. [15] argue for rule-based methods and develop a rule discovery algorithm. Tang et al. [28] use machine learning and regard concept linking as minimizing Bayesian risk of decision making. As the size of KB increases, many semi-supervised or unsupervised methods appear. For example, Rong et al. [23] transfer the entity matching problem to a binary classi cation problem. Wang et al. [40] present a factor graph model to learn the alignment across knowledge bases. For data integration across social networks, Zhang et al. [46] propose COSNET, an energy-based model that considers global and local consistency. Pellissier et al. [21] utilize existed hyper-links and build an online platform for tagging manually.

Recently, network embedding and knowledge graph embedding have been proved to be ef cient in many downstream tasks. Nickel et al. [20] view the embedding training as tensor factorization. Bordes et al. [1] propose TransE to interpret multi-relational data as translations operating. Trouillon et al. [32] use complex embedding with composition to represent relation. Yang et al. [43] put forward DistMult to jointly embed entities and relations by neural networks. Feng et al. [8] formulate the knowledge base as a directed graph and learn representations leveraging the graph's structural information. These works are generally unsupervised learning on a single knowledge graph and pay little attention to cross knowledge bases situation.

In the more specific field of entity linking, many supervised embedding method appear these years. Chen et al. [2] propose an embedding-based model for multilingual entity alignment based on TransE. Zhu et al. [47] develop an iterative method for entity alignment via joint embeddings. Sun et al. [27] propose a joint attribute-preserving embedding model for cross-lingual entity alignment. Trivedi et al. [31] consider jointly combining link prediction and cross-linking tasks using attention. These methods rely on a large number of seed or anchor entities to ensure accuracy. However, few studies focus on the unsupervised embedding method and aim to find a unified solution for noisy and large-scale knowledge bases linking.

In this work, we propose a unified framework SelfLinKG to fill the gap. We first unify knowledge bases as heterogeneous information networks and employ attention to aggregate crucial structural information. To address the same embedding space problem, we propose to use a shared encoder mechanism and unique embedding. To solve the high-cost and ambiguous problem, we propose the unsupervised momentum contrastive learning. To learn more about self-supervised contrastive learning, please refer to [17].

7 Conclusion and Discussion

In this work, we propose an self-supervised model SelfLinKG for linking large-scale heterogeneous knowledge bases. Without labeled data, SelfLinKG uses global momentum contrastive learning to learn a shared representation among multiple knowledge bases. Our experiments on two large-scale graphs show that the proposed unsupervised SelfLinKG can achieve a comparable performance with its supervised counterpart. We apply the model to automatically generate linkings among 14 different knowledge bases and make the linked graphs publicly available.

As the future work, it would be rather interesting to design a knowledge linking system to automatically harvest knowledge (linking new knowledge into existing bases) from the open Web. It would be also exciting to explore novel methods to make the model more robust, as the open data is always noisy.

There are also some open problems about leveraging contrastive objective in knowledge graph embedding. It requires further study on whether our SelfLinKG is adaptive to KBs that have fewer cross-links or from very different domains. Intuitively, we think the number of cross-links is not a problem, because the contrastive objective aims at scattering nodes' embedding uniformly in the sphere space and we can still efficiently narrow down distances of positive pairs. For KBs from very different domains, it is probably very hard to self-supervisedly link them up if very little common information is shared between target pairs' attributes and structure, in which case supervised labels may be still indispensable.

For questions about whether this contrastive objective could help the embedding learning within one knowledge base, we think it depends on the downstream task. Some very recent work [11], [22] demonstrate the effectiveness of it on node classification and graph classification in networks, but also show that it may not help tasks such as relation prediction and link prediction. We believe it is probably the same for the single knowledge graph embedding task. Limited to the passage, we do not study the problem in this work.

Acknowledgement

The work is supported by the National Key R&D Program of China (2018YFB1402600), NSFC for Distinguished Young Scholar (61825602), NSFC (61836013) and Tsinghua-Bosch Joint ML Center, Deptment of Computer Science and Technology, Tsinghua University.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In NIPS, pages 2787–2795, 2013.
[2] M. Chen, Y. Tian, M. Yang, and C. Zaniolo. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. arXiv preprint arXiv:1611.03954, 2016.
[3] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning. What does bert look at? an analysis of bert's attention. arXiv preprint arXiv:1906.04341, 2019.
[4] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: Pre-training text encoders as discriminators rather than generators. In ICLR, 2019.
[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
[6] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 601–610, 2014.
[7] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. TKDE, 19(1):1–16, 2007.
[8] J. Feng, M. Huang, Y. Yang, and X. Zhu. Gake: Graph aware knowledge embedding. In COLING 2016: Technical Papers, pages 641–651, 2016.
[9] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In (CVPR'06), volume 2, pages 1735–1742. IEEE, 2006.
[10] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li. Openke: An open toolkit for knowledge embedding. In EMNLP, pages 139–144, 2018.
[11] K. Hassani and A. H. Khasahmadi. Contrastive multi-view representation learning on graphs. arXiv preprint arXiv:2006.05582, 2020.
[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. arXiv preprint arXiv:1911.05722, 2019.
[13] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In ICML, pages 1188–1196, 2014.
[14] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. IEEE TKDE, 21(8):1218–1232, 2008.
[15] L. Li, J. Li, and H. Gao. Rule-based method for entity resolution. TKDE, 27(1):250–263, 2015.
[16] L. Liu, W. K. Cheung, X. Li, and L. Liao. Aligning users across social networks using network embedding. In Ijcai, pages 1774–1780, 2016.
[17] X. Liu, F. Zhang, Z. Hou, Z. Wang, L. Mian, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. arXiv, pages arXiv–2006, 2020.
[18] T. Man, H. Shen, S. Liu, X. Jin, and X. Cheng. Predict anchor links across social networks via an embedding approach. In Ijcai, volume 16, pages 1823–1829, 2016.
[19] N. Maximilian, R. Lorenzo, and P. Tomaso. Holographic embeddings of knowledge graphs. arXiv preprint arXiv:1510.04935, 2015.
[20] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In Icml, volume 11, pages 809–816, 2011.
[21] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher. From freebase to wikidata: The great migration. In WWW, pages 1419–1428, 2016.