Knowledge-Based Systems 259 (2023) 110061

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Variational autoencoder densified graph attention for fusing synonymous entities: Model and protocol

Qian Li [a,b], Daling Wang [a,*], Shi Feng [a], Kaisong Song [a,c], Yifei Zhang [a], Ge Yu [a]

[a] School of Computer Science and Engineering, Northeastern University, Shenyang, China
[b] School of Computer Science and Engineering, Nanyang Technological University, Singapore
[c] DAMO Academy, Alibaba Group, Hangzhou, China

ARTICLE INFO

Article history:
Received 1 August 2022
Received in revised form 17 October 2022
Accepted 19 October 2022
Available online 27 October 2022

Keywords:
Open knowledge graph
Knowledge graph representation
Cluster ranking
Link prediction

ABSTRACT

The prediction of missing links of open knowledge graphs (OpenKGs) poses unique challenges compared with well-studied curated knowledge graphs (CuratedKGs). Unlike CuratedKGs whose entities are fully disambiguated against a fixed vocabulary, OpenKGs consist of entities represented by non-canonicalized free-form noun phrases and do not require an ontology specification, which drives the *synonymity* (multiple entities with different surface forms have the same meaning) and *sparsity* (a large portion of entities with few links). How to capture synonymous features in such sparse situations and how to evaluate the multiple answers pose challenges to existing models and evaluation protocols. In this paper, we propose VGAT, a variational autoencoder densified graph attention model to automatically mine synonymity features, and propose CR, a cluster ranking protocol to evaluate multiple answers in OpenKGs. For the model, VGAT investigates the following key ideas: (1) phrasal synonymity encoder attempts to capture phrasal features, which can automatically make entities with synonymous texts have closer representations; (2) neighbor synonymity encoder mines structural features with a graph attention network, which can recursively make entities with synonymous neighbors closer in representations. (3) densification attempts to densify the OpenKGs by generating similar embeddings and negative samples. For the protocol, CR is designed from the significance and compactness perspectives to comprehensively evaluate multiple answers. Extensive experiments and analysis show the effectiveness of the VGAT model and rationality of the CR protocol.

1. Introduction

Open information extraction systems [1,2] automatically extract triples (*"subject noun phrase"*, *"relation phrase"*, *"object noun phrase"*) from unstructured data. Open Knowledge Graphs (OpenKGs) [3,4] are constructed with these *"noun phrases"* as entities and *"relation phrases"* as relations, and do not require the specification of ontology or relational schema, which thus can easily adapt to new domains. As an important application requirement, link prediction of OpenKGs aims to predict missing links by learning the effective representations of entities and relations, in which similar entities are assigned close representations and different ones are assigned distant representations. Although studies have been concerned with link prediction of Curated Knowledge Graphs (CuratedKGs), relatively few studies have focused on that of OpenKGs. This study primarily focuses on identifying unique challenges in the link prediction of OpenKGs, investigating an effective model and evaluation protocol to address the challenges, and providing comprehensive empirical insights and analysis.

Unlike CuratedKG whose entities are fully disambiguated against a fixed vocabulary, OpenKGs consist of entities represented by non-canonica-lized free-form noun phrases. The non-canonicalized free-form phrases and roughness of non-ontological construction methods drive the two inherent characteristics: synonymity, i.e., multiple entities with different surface forms have the same meaning, e.g., entities *"NBC-TV"*, *"NBC"* and *"NBC Television"* in Fig. 1(a); and sparsity, i.e., a large portion of entities have few links, e.g., entities with less than two degrees account for 40% and 82% in the two standard OpenKGs: ReVerb20K and ReVerb45K [4]. These synonymity and sparsity characteristics pose challenges to models and protocols in link prediction of OpenKGs: (1) model: how to capture *synonymous* features in such *sparse* situations? (2) protocol: how to evaluate the multiple *synonymous* answers?

The model challenge is how to capture synonymous features in such sparse situations. A better link prediction model for OpenKGs should make the representations of synonymous entities closer to each other, while ones with different meanings be

* Corresponding author.
E-mail address: wangdaling@cse.neu.edu.cn (D. Wang).

https://doi.org/10.1016/j.knosys.2022.110061
0950-7051/

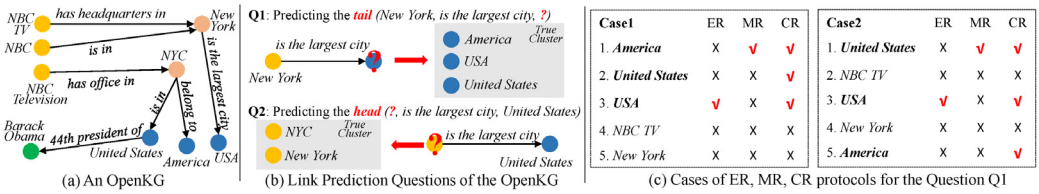


Fig. 1. (a) A sub-graph of OpenKG. (b) Two link prediction examples of OpenKG, in which Q1 predicts the head and Q2 predicts the tail. (c) Two cases of predicted results and ER, MR and CR scores for Q1, in which *True Cluster* has three entities: *America*, *USA* and *United States*, and entity *USA* is assumed to be the true entity for the ER score.

more distinguished in vector spaces. In this case, models for CuratedKGs, such as TransE [5], DistMult [6], ComplEx [7], ConvE [8], KBGAT [9], GGAE [10] exhibit poor performance in OpenKGs owing to the synonymity and sparsity. CaRe [11] attempts to learn canonicalization infused embeddings for OpenKGs, which fakes synonymous entities by integrating canonicalization and side information in an error-conscious manner. OKGIT [12] employs pretrained language models (PLMs) as type compatibility scores to improve the CaRe model. These OpenKG-specific models attempt to fake synonymous entities with side information and alleviate the sparsity with PLMs. However, they are sensitive to external information and PLMs, which are difficult to train, reproduce, and transfer to downstream tasks. The intention of constructing an OpenKG is to convert text into structure for easy calculation, that is, an OpenKG is rich in world knowledge. Therefore, we focus on automatically mining synonymity features with information contained by an OpenKG.

The protocol challenge is how to evaluate the multiple synonymous answers. Unlike the link prediction of CuratedKGs which involves predicting only one answer entity, the link prediction of OpenKGs is to predict multiple answer entities caused by entity synonymity.[1] Entity Ranking (ER) [5] is the most commonly used single-test protocol in CuratedKGs which have only one answer entity. Using ER directly to evaluate the results of OpenKGs is inappropriate owing to entity synonymity. For example in Fig. 1(b), Q1 is used to predict tail entities according to head entity *"New York"* and relation *"is the largest city"*. Entities in the true cluster, *"America"*, *"USA"*, *"United States"*, all belong to answer entities. ER only using entity *"USA"* as the answer, often provides one-sided evaluation feedback. Mention Ranking (MR) [3,11], which is a single-test protocol designed for OpenKGs, regards the minimum ranking position of all answer entities as the ranking score, which considers only an answer entity with the best score but still ignores the influence of other answer entities. For example in Fig. 1(c), the MR scores of case1 and case2 are 1, providing the same criteria to two different results. Therefore, existing protocols cannot satisfy the requirements of OpenKGs, and this motivates the proposal of an effective protocol for the link prediction of OpenKGs.

In this paper, we propose a **V**ariational autoencoder densified **G**raph **A**ttention model (**VGAT**) for the first challenge, and propose a **C**luster **R**anking protocol (**CR**) for the second challenge. The VGAT model investigates the following three components: (1) phrasal synonymity encoder attempts to capture phrasal features, to automatically make entities with synonymous texts have closer representations. (2) neighbor synonymity encoder mines structural features with graph attention network, to recursively make entities with synonymous neighbors have closer representations. (3) densification attempts to densify the OpenKGs by generating similar embeddings and negative samples. The CR protocol is designed from significance and compactness perspectives to comprehensively evaluate multiple answers. Through extensive experiments on benchmarks, VGAT significantly outperforms state-of-the-art models and CR correctly evaluates the models that cannot be evaluated by existing protocols. In summary, we highlight the key contributions:

- We propose a novel VGAT model to automatically mine the synonymous features of OpenKGs.
- We propose a novel CR protocol to evaluate multiple answers for the link prediction of OpenKGs.
- Extensive experiments show the effectiveness of the VGAT model and rationality of the CR protocol. We release our code at https://github.com/feiwangyuzhou/VGAT.

2. Related work

2.1. Open knowledge graphs and representations

With the rapid development of artificial intelligence [13–15], graphs [16,17] and knowledge graphs [12] are rich in structural inferential features and have been widely used in knowledge tracing [18,19], recommendation systems [20], sentiment analysis [21] and link prediction [10]. OpenKG is a broader type of knowledge graph with noun phrases as entities and relation phrases as relations. As key tools for constructing OpenKGs, Open Information Extraction systems (OIE) [22], such as ReVerb [23], OLLIE [24], ClauseIE [25], RelNoun [26], BONIE [27], CALMIE [1] and OPIEC [2], have been used to automatically extract phrase triples from raw data. Owing to the methods of their automatic construction, OpenKGs tend to be nosier without canonicalizing the naming of entities and relations. Many studies have conducted automatic canonicalizations of OpenKGs to obtain CuratedKGs, then apply CuratedKGs to downstream tasks. Logical rules and cluster algorithms have been adopted to classify relation phrases [28] and noun phrases [29]. CESI [4] performs canonicalization with hierarchical agglomerative clustering by combing embeddings and relevant phrase side information. MGNN [30] applies a graph neural network model to canonicalize OpenKG by aggregating the representations of noun and relation phrases through a multi-layered meta-graph structure. However, in the canonicalization process, considerable textual and structural semantic information may be lost. Directly learning the representations of OpenKGs benefits the capturing of rich characteristics. The rich textual and structural semantic information contained in OpenKGs is more conducive to downstream tasks than pure textual features in unstructured data and pure structural information in CuratedKGs. However, considerably few studies have paid attention to OpenKGs embeddings. CaRe [11] learns canonicalization infused embedding for OpenKGs. OKGIT [12] uses the output of PLMs as type compatibility scores to improve the CaRe model in terms of type compatibility. OKGIT is a state-of-the-art model for OpenKGs, which is sensitive to side information and masked language models and ignores potential features contained in the OpenKG. The effective embedding of OpenKGs still remains a challenge.

[1] Our evaluation protocol is a single-test protocol.

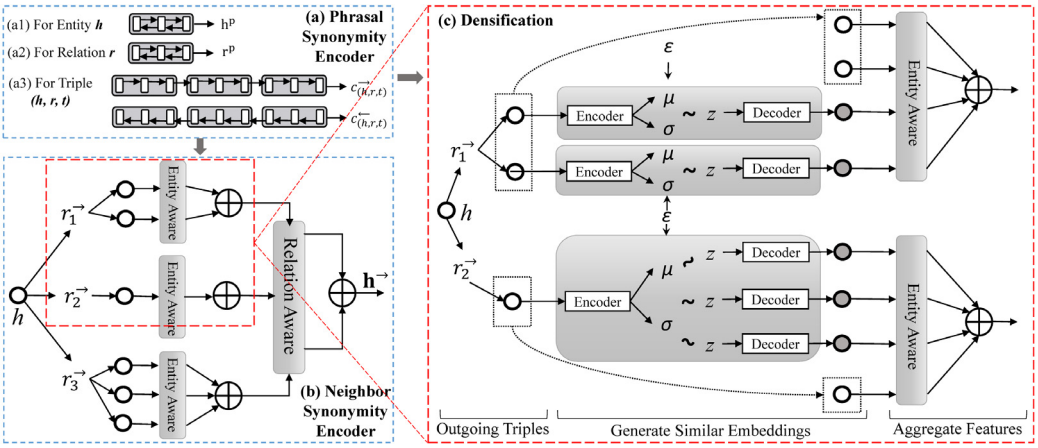


Fig. 2. Framework of proposed VGAT model with (a) phrasal synonymity encoder, (b) neighbor synonymity encoder and (c) densification.

2.2. Evaluation protocols

The evaluation protocols [31] of link prediction can be summarized as: single-test protocol for a single test triple, and all-test protocol for all test triples. These two types of protocols are progressive, i.e., all-test protocol must be calculated after the single-test protocol. Some common evaluation protocols, such as *H@N* [5,6], *AR* [8,31] and *ARR* [3,11], belong to the all-test protocols designed for all test triples when the scores for each test triple are known. The existing studies on these protocols have focused more on the all-test protocol, and few studies have paid attention to the single-test protocol. ER [5] is a classic single-test protocol for CuratedKGs. Because only one answer exists for a link prediction question of CuratedKGs, ER is adequate for obtaining the ranking score for the answer. However, in OpenKGs, multiple answers exist for a link prediction question owing to entity synonymity. Directly using ER to evaluate the results of OpenKGs is inappropriate. MR [3,11], designed for OpenKGs, regards the minimum ranking position of all answer entities as the ranking score. To the best of our knowledge, MR is a unique single-test protocol for OpenKGs. However, MR considers only the answer entity with the best score but still ignores the influence of other answer entities. Therefore, we propose CR to account for the deficiencies of these evaluation protocols.

3. Preliminaries

Definition 1 (*Open Knowledge Graph (OpenKG) and Representations*)**.** In an OpenKG $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, let a triple be (h, r, t), where $h, t \in \mathcal{E}$ represent entities and $r \in \mathcal{R}$ represents the relation between entities h and t. $|\mathcal{E}|$, $|\mathcal{R}|$ are the number of entities and relations. Entities h, t and relation r are represented by non-empty sequences of words from a word vocabulary, where $w_h = \{w_{h,i}\}_{i=1}^{|w_h|}$, $w_r = \{w_{r,i}\}_{i=1}^{|w_h|}$ are the word sequences of entity h and relation r, separately. $|w_h|$, $|w_r|$ are the numbers of words. The representations of entities and relations are $\mathbf{E} \in \mathbb{R}^{N_e \times T}$ and $\mathbf{R} \in \mathbb{R}^{N_r \times T}$, where N_e, N_r are the numbers of entities and relations, and T is feature dimension. $\mathbf{h} \in \mathbf{E}$ is initial representation of entity h, and $\mathbf{r} \in \mathbf{R}$ is initial representation of relation r. In an OpenKG, there are multiple entities with the same meaning but different expression forms [3]. Entities with the same meaning belong to the same entity-cluster, denoted as $C_j = \{C_{j,i}\}_{i=1}^{|C_j|}$, where $C_{j,i} \in \mathcal{E}$, $|C_j|$ is the number of entities in the entity-cluster. Let $C(\varepsilon) = \{C_j\}_{j=1}^{|C(\varepsilon)|}$ denote the set of all entity clusters, where $|C(\varepsilon)|$ is the number of all entity-clusters.

Definition 2 (*Link Prediction of OpenKGs*)**.** The link prediction of OpenKGs is to predict all answer entities in the answer-cluster for the two questions: Q1, predicting the head entity-cluster $Q_{C_t} = (?, r, h)$ and Q2, predicting the tail entity-cluster $Q_{C_t} = (h, r, ?)$. Let the answer-cluster be $C_t = \{C_{t,i}\}_{i=1}^{|C_t|}$. We denote all candidate answer-clusters as $\zeta_C = \{C_j\}_{j=1}^{|C(\varepsilon)|}$, all candidate answer entities as $\zeta_E = \{t_i\}_{i=1}^{|\mathcal{E}|}$.

4. Proposed VGAT model

We propose the VGAT model to automatically learn synonymous features for OpenKGs. The framework of VGAT is shown in Fig. 2. In the following, we propose a phrasal synonymity encoder for capturing phrasal features in Section 4.1, a neighbor synonymity encoder for mining structural features in Section 4.2, densification module to alleviate the sparsity by generated similar embeddings and negative samples in Section 4.3, a decoder for fusing these above synonymity embeddings and apply them to downstream task in Section 4.4, and a training procedure in Section 4.5.

4.1. Phrasal synonymity encoder

Synonymity in OpenKGs is manifested in different expression forms of multiple entities with the same meaning. The different expression forms refer to the expressions of multiple phrases with the same semantics. Because of the recent achievements in natural language processing [32,33], many models, e.g., GRU [33], LSTM [34], and BERT [35], can now learn the textual semantic features for natural language phrases and sentences. The representations of entities with the same meaning and different expression forms, e.g., *"NBC"*, *"NBC-TV"*, and *"NBC Television"*, modeled by textual technologies can be more similar to each other. Therefore, we propose using textual technologies to learn textual synonymous features. For a sequence $w_i = \{w_{i,m}\}_{m=1}^{|w_i|}$,

$$\mathbf{w_i^{\rightarrow}} = \phi^{\rightarrow}(w_i), \quad \mathbf{w_i^{\leftarrow}} = \phi^{\leftarrow}(w_i) \tag{1}$$

where $\phi^{\rightarrow}, \phi^{\leftarrow}$ are phrasal operations in two opposite directions, $\phi^{\rightarrow}$ is used to capture phrasal features from left-to-right and $\phi^{\leftarrow}$ is used to capture phrasal features from right-to-left.

For head entity h, relation r and tail entity t, the word sequence of entity h is denoted as $w_h = \{w_{h,m}\}_{m=1}^{|w_h|}$, that of relation r as $w_r = \{w_{r,1}\}_{m=1}^{|w_r|}$, and that of tail entity t as $w_t = \{w_{t,1}\}_{m=1}^{|w_t|}$. For $w_i \in \{w_h, w_r, w_t\}$, $\mathbf{w_i^{\rightarrow}}$ and $\mathbf{w_i^{\leftarrow}}$ are concatenated to obtain a bidirectional phrasal embedding $\mathbf{w_i^p} = [\mathbf{w_i^{\rightarrow}}; \mathbf{w_i^{\leftarrow}}]$. We denote

the phrasal embeddings of head entity h, relation r and tail entity t as $\mathbf{h^p}$, $\mathbf{r^p}$, $\mathbf{t^p}$, respectively (Fig. 2(a1, a2)).

For a triple (h, r, t), its word sequence is denoted as $w_{(h,r,t)} = [w_h; w_r; w_t]$, which is the concatenation of the word sequences of head entity h, relation r, and tail entity t. We use the left side of Eq. (1) to capture embedding $\mathbf{w}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ in the direction of $h \rightarrow r \rightarrow t$, and use the right side of Eq. (1) to capture embedding $\mathbf{w}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ in the direction of $h \leftarrow r \leftarrow t$. Moreover, we concatenate phrasal embeddings $[\mathbf{h^p}; \mathbf{r^p}; \mathbf{t^p}]$ $(h \rightarrow r \rightarrow t)$ and $[\mathbf{t^p}; \mathbf{r^p}; \mathbf{h^p}]$ $(h \leftarrow r \leftarrow t)$ to obtain another representations for the triple. When the triple (h, r, t) acts as an outgoing neighbor triple of entity h, embedding $\mathbf{w}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ or $[\mathbf{h^p}; \mathbf{r^p}; \mathbf{t^p}]$ can capture outgoing features. When the triple (h, r, t) acts as an incoming neighbor triple of entity t, embedding $\mathbf{w}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ or $[\mathbf{t^p}; \mathbf{r^p}; \mathbf{h^p}]$ can capture incoming features. Therefore, the representations of triple (h, r, t) (Fig. 2(a3)) are expressed as follows:

$$c^{\rightarrow,\mathbf{w}}_{(h,r,t)} = A_1 \mathbf{w}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}, \qquad c^{\leftarrow,\mathbf{w}}_{(h,r,t)} = A_1 \mathbf{w}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})} \tag{2}$$

$$c^{\rightarrow,\mathbf{p}}_{(h,r,t)} = A_2[\mathbf{h^p}; \mathbf{r^p}; \mathbf{t^p}], \quad c^{\leftarrow,\mathbf{p}}_{(h,r,t)} = A_2[\mathbf{t^p}; \mathbf{r^p}; \mathbf{h^p}] \tag{3}$$

where A_1 and A_2 denote linear matrices.

By capturing textual features, entities with synonymous texts are closer in semantic representation, which can preliminarily address the key challenge of textual synonymity.

4.2. Neighbor synonymity encoder

The neighbor synonymity encoder aims to derive synonymous structure features of entities based on the following assumption: if two entities are synonymous, their neighbors are often synonymous or even connected. For example, in Fig. 1(a), entities *"NYC"* and *"New York"* are synonymous, the neighbor *"America"* of *"NYC"* is also synonymous with the neighbors *"USA"* and *"United States"* of *"New York"*.

For an entity h, its direct outgoing $D^{\rightarrow}_h$ and incoming $D^{\leftarrow}_h$ neighbors are stored in a triple set:

$$D^{\rightarrow}_h = \{\{(h, r_j, t_i)\}_{t_i \in \mathcal{E}^{\rightarrow}_{(h,r_j)}}\}_{r_j \in \mathcal{R}^{\rightarrow}_h} \tag{4}$$

$$D^{\leftarrow}_h = \{\{(t_i, r_j, h)\}_{t_i \in \mathcal{E}^{\leftarrow}_{(r_j,h)}}\}_{r_j \in \mathcal{R}^{\leftarrow}_h} \tag{5}$$

where $\mathcal{E}^{\rightarrow}_{(h,r_j)}$ contains entities with the same pair (h, r_j), and $\mathcal{R}^{\rightarrow}_h$ contains relations with h as the head entity. $\mathcal{E}^{\leftarrow}_{(r_j,h)}$ contains entities with the same pair (r_j, h), and $\mathcal{R}^{\leftarrow}_h$ contains relations with h as the tail entity. The benefits of each neighbor triple to the entity h are inconsistent, some neighbor triples are more profitable than others. We design a two-layer graph attention mechanism to learn entity-aware and relation-aware features. This mechanism assigns different weights to different neighbors according to their benefit, and aggregate favorable information with the weights. Next, we take outgoing neighbors $D^{\rightarrow}_h$ as an example to introduce the remaining component. The outgoing neighboring embedding $\mathbf{h}^{\rightarrow}$ of entity h is computed as:

$$\mathbf{h}^{\rightarrow} = \sum_{r_j \in \mathcal{R}^{\rightarrow}_h} a^{\rightarrow}_{(h,r_j)} \underbrace{\Big(\sum_{t_i \in \mathcal{E}^{\rightarrow}_{(h,r_j)}} a^{\rightarrow}_{(h,r_j,t_i)} c^{\rightarrow}_{(h,r_j,t_i)} \Big)}_{} \tag{6}$$

$$a^{\rightarrow}_{(h,r_j)} = \frac{exp(\varrho(e^{\rightarrow}_{(h,r_j)}))}{\sum_{r_k \in \mathcal{R}^{\rightarrow}_h} exp(\varrho(e^{\rightarrow}_{(h,r_k)}))} \tag{7}$$

$$a^{\rightarrow}_{(h,r_j,t_i)} = \frac{exp(\varrho(c^{\rightarrow}_{(h,r_j,t_i)}))}{\sum_{t_k \in \mathcal{E}^{\rightarrow}_{(h,r_j)}} exp(\varrho(c^{\rightarrow}_{(h,r_j,t_k)}))} \tag{8}$$

where $a^{\rightarrow}_{(h,r_j,t_i)}$ is the entity-aware attention weight for entity $t_i^{\rightarrow}$ with the pair-$(h, r_j^{\rightarrow})$, and $a^{\rightarrow}_{(h,r_j)}$ is the relation-aware attention weight for relation $r_j^{\rightarrow}$ with head entity h. ϱ is LeakyReLU non-linearity function, and *exp* is an exponential function. $c^{\rightarrow}_{(h,r_j,t_i)}$ denotes the representation of triple (h, r_j, t_i) in Eqs. (2) or (3). Therefore, we obtain two outgoing neighboring embeddings: $\mathbf{h}^{\rightarrow}_{\mathbf{w}}$ and $\mathbf{h}^{\rightarrow}_{\mathbf{p}}$, where $\mathbf{h}^{\rightarrow}_{\mathbf{w}}$ with $c^{\rightarrow,\mathbf{e}}_{(h,r_j,t_i)}$ in Eq. (2) and $\mathbf{h}^{\rightarrow}_{\mathbf{p}}$ with $c^{\rightarrow,\mathbf{p}}_{(h,r_j,t_i)}$ in Eq. (3). The framework of outgoing neighbors is shown in Fig. 2(b).

Thus far, the outgoing neighboring embedding $\mathbf{h}^{\rightarrow}_{\mathbf{w}}$, $\mathbf{h}^{\rightarrow}_{\mathbf{p}}$ of entity h is obtained, and the incoming neighboring embedding $\mathbf{h}^{\leftarrow}_{\mathbf{w}}$, $\mathbf{h}^{\leftarrow}_{\mathbf{p}}$ of entity h can be computed in a similar manner. By capturing structural features, entities with similar connected relations and multi-hop neighborhoods are automatically closer in the semantic representation.

4.3. Densification

OpenKGs are extremely sparse, e.g. entities with degrees less than two account for 40% of ReVerb20k and 82% of ReVerb45k (Fig. 4). Having few neighbors makes graph attention ineffective. Therefore, we attempt to densify the OpenKG by generating similar samples with Variational AutoEncoder (VAE) [36,37].

Generate Similar Embeddings As an unsupervised generative framework, a VAE can generate unseen samples with the same distribution as the input data. We design two VAEs to generate similar embeddings for entities (EVAE) and triples (TVAE). EVAE uses the embedding of an entity as input and generates one and more similar embeddings of the entity, while TVAE uses the embedding of a triple, i.e. (h, r, t) as input and generates one and more similar embeddings of the triple. These generated embeddings are similar but differ from the representations of target entities or triples, which can help the model understand them from multiple perspectives. We now describe below the inner workings of EVAE, and an analogous description follows for TVAE. The framework of EVAE is shown in Fig. 2(c). EVAE uses the projection module to learn the latent Gaussian distribution of input representation $\mathbf{x}$ and generation module to reconstruct $\widetilde{\mathbf{x}}$ by sampling from the latent Gaussian distribution. For the projection,

$$\mu_x = A_{\mu_x}(f_e(\mathbf{x})), \quad \sigma_x^2 = A_{\sigma_h}(f_e(\mathbf{x})) \tag{9}$$

$$q(z|x) = \mathcal{N}(\mu_x, \sigma_x^2 \mathbf{I}) \tag{10}$$

where f_e denotes a two layer fully connected neural network with a tanh non-linearity activation function, A_{μ_x}, A_{σ_x} denote linear matrices. For the generation, we use the reparameterization trick to sample z, and reconstruct $\widetilde{\mathbf{x}}$ from z:

$$z = \mu_x + \sigma_x \circ \epsilon, \quad \widetilde{\mathbf{x}} = f_d(z) \tag{11}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ (i.e., a standard normal distribution), $\circ$ denotes element-wise multiplication, f_d $(f_d \neq f_e)$ represents another two-layer fully connected neural network with a tanh non-linearity activation function.

Aggregate Features For an entity t, EVAE generates a similar embedding $\widetilde{\mathbf{t}}^{\mathbf{p}}$ with phrasal embedding $\mathbf{t^p}$ as input. For a triple (h, r, t), TVAE generates similar embeddings $\widetilde{\mathbf{w}}^{\rightarrow}_{(h,r,t)}$ or $\widetilde{\mathbf{w}}^{\leftarrow}_{(h,r,t)}$ with phrasal embeddings $\mathbf{w}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ or $\mathbf{w}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ as input. Therefore, the VAE-based representations of a triple (h, r, t) are generated as follows:

$$\widetilde{c}^{\rightarrow,\mathbf{w}}_{(h,r,t)} = A_1 \widetilde{\mathbf{w}}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})} \qquad \widetilde{c}^{\leftarrow,\mathbf{w}}_{(h,r,t)} = A_1 \widetilde{\mathbf{w}}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})} \tag{12}$$

$$\widetilde{c}^{\rightarrow,\mathbf{p}}_{(h,r,t)} = A_2[\mathbf{h^p}; \mathbf{r^p}; \widetilde{\mathbf{t}}^{\mathbf{p}}] \quad \widetilde{c}^{\leftarrow,\mathbf{p}}_{(h,r,t)} = A_2[\widetilde{\mathbf{t}}^{\mathbf{p}}; \mathbf{r^p}; \mathbf{h^p}] \tag{13}$$

Next, the generated similar embeddings are injected into the graph attention mechanism in Eqs. (6) and (8):

$$\mathbf{h}^{\rightarrow} = \sum_{r_j \in \mathcal{R}_h^{\rightarrow}} \Big[a^{\rightarrow}_{(h,r_j)} \Big(\sum_{t_i \in \mathcal{E}^{\rightarrow}_{(h,r_j)}} a^{\rightarrow}_{(h,r_j,t_i)} c^{\rightarrow}_{(h,r_j,t_i)} + \sum_{t_i \in \widetilde{\mathcal{E}}^{\rightarrow}_{(h,r_j)}} \widetilde{a}^{\rightarrow}_{(h,r_j,t_i)} \widetilde{c}^{\rightarrow}_{(h,r_j,t_i)} \Big) \Big] \tag{14}$$

$$\widetilde{a}^{\rightarrow}_{(h,r_j,t_i)} = \frac{exp(\varrho(\widetilde{c}^{\rightarrow}_{(h,r_j,t_i)}))}{\sum_{t_k \in \{\mathcal{E}^{\rightarrow}_{(h,r_j)}, \widetilde{\mathcal{E}}^{\rightarrow}_{(h,r_j)}, n\mathcal{E}^{\rightarrow}_{(h,r_j)}\}} exp(\varrho(c^{\rightarrow}_{(h,r_j,t_k)}))} \tag{15}$$

where the range of t_k extends from $\mathcal{E}^{\rightarrow}_{(h,r_j)}$ in Eq. (8) to $\{\mathcal{E}^{\rightarrow}_{(h,r_j)}, \widetilde{\mathcal{E}}^{\rightarrow}_{(h,r_j)}, n\mathcal{E}^{\rightarrow}_{(h,r_j)}\}$ in Eq. (15). $\widetilde{\mathcal{E}}^{\rightarrow}_{(h,r_j)}$ contains the entities with generated embeddings whose number is specific to each entity: 0 if $N_e > |\mathcal{E}^{\rightarrow}_{(h,r_j)}|$ and $N_e - |\mathcal{E}^{\rightarrow}_{(h,r_j)}|$ otherwise, where N_e is the maximum number of adjacent entities for each entity, and $|\mathcal{E}^{\rightarrow}_{(h,r_j)}|$ is the number of true adjacent entities for a pair (h, r_j). That is, if the number of true adjacent entities is less than the above maximum value N_e, the embeddings generated by the densification module are used to complement. In addition to the generated samples $\widetilde{\mathcal{E}}^{\rightarrow}_{(h,r_j)}$, we construct negative samples $n\mathcal{E}^{\rightarrow}_{(h,r_j)}$ by corrupting the true tail entities in $\mathcal{E}^{\rightarrow}_{(h,r_j)}$, and contrast these negative samples with the true and generated samples to learn distinguishable features.

4.4. Decoder

The decoder attempts to fuse the synonymity embeddings and apply them to downstream tasks. For an entity h, the synonymity embeddings ($\mathbf{h^p}$) from the phrasal synonymity encoder (Section 4.1) and ($\mathbf{h}^{\rightarrow}_{\mathbf{w}}$, $\mathbf{h}^{\rightarrow}_{\mathbf{p}}$, $\mathbf{h}^{\leftarrow}_{\mathbf{w}}$, $\mathbf{h}^{\leftarrow}_{\mathbf{p}}$) from the neighbor synonymity encoder (Section 4.2) with densification (Section 4.3) are fused with attention weights to obtain the final embedding $\mathbf{h^f}$:

$$\mathbf{h^m} = \sum_{\mathbf{h_i} \in \{\mathbf{h^p}, \mathbf{h}^{\rightarrow}_{\mathbf{e}}, \mathbf{h}^{\rightarrow}_{\mathbf{p}}, \mathbf{h}^{\leftarrow}_{\mathbf{e}}, \mathbf{h}^{\leftarrow}_{\mathbf{p}}\}} b_i \mathbf{h_i} \tag{16}$$

$$b_i = \frac{exp(\varrho(A_g[\mathbf{h}; \mathbf{h_i}]))}{\Sigma_{\mathbf{h_k} \in \{\mathbf{h^p}, \mathbf{h}^{\rightarrow}_{\mathbf{e}}, \mathbf{h}^{\rightarrow}_{\mathbf{p}}, \mathbf{h}^{\leftarrow}_{\mathbf{e}}, \mathbf{h}^{\leftarrow}_{\mathbf{p}}\}} exp(\varrho(A_g[\mathbf{h}; \mathbf{h_k}]))} \tag{17}$$

where A_g denotes the linear matrix, and $\mathbf{h} \in \mathbf{E}$[2] is the trainable embedding for entity h. The final embedding $\mathbf{h^f}$ of entity h is $\mathbf{h^f} = \mathbf{h^m} \oplus \mathbf{h^p} \oplus \mathbf{h}$, where $\oplus$ denotes aggregation operations, such as addition and concatenation.

The final embedding $\mathbf{r^f}$ of relation r is the synonymity embedding from the phrasal synonymity encoder (Section 4.1), that is, $\mathbf{r^f} = \mathbf{r^p}$.

We take the link prediction task as the downstream task. The aim of link prediction task is to predict whether a triple is correct (large similarity score) or incorrect (little similarity score). The similarity score for each test triple is computed by capturing potential connections between entities and relations:

$$S(h, r, t) = \sigma(\text{Linear}(\sigma([\widehat{\mathbf{h}^{\mathbf{f}}}; \widehat{\mathbf{r}^{\mathbf{f}}}] * \omega)))\mathbf{t^f} \tag{18}$$

where we use a 2D convolutional network [8] with filters ω to learn potential connections. $\widehat{\mathbf{h}^{\mathbf{f}}} \in \mathbb{R}^{T_1 \times T_2}$ are reshaped from $\mathbf{h^f} \in \mathbb{R}^T$, $T = T_1 T_2$, and $\widehat{\mathbf{r}^{\mathbf{f}}}$ has a similar reshaping. σ denotes the ReLU activation. Convoluted embedding is projected to dimension T with Linear and multiplied by the embedding $\mathbf{t^f}$ of the tail entity.

[2] Each entity and relation in the KG can be assigned a uniquely initialized and trainable embedding, where $\mathbf{h} \in \mathbf{E}$, $\mathbf{r} \in \mathbf{R}$ are trainable embeddings for entity h and relation r, respectively.

4.5. Training procedure

The loss function used in this study comprises two components: one for the link prediction task and another for the VAEs. For link prediction task, we use a binary cross-entropy loss to optimize the parameters:

$$L_{task} = -\frac{1}{\mathcal{M}} \sum_{i=1}^{\mathcal{M}} (Y_i \cdot \log S(h, r, i) + (1 - Y_i) \cdot \log(1 - S(h, r, i))) \tag{19}$$

where $Y_i = 1$ if triple (h, r, i) is correct; $Y_i = 0$ otherwise. $\mathcal{M}$ is the number of training triples with both correct ($\in \mathcal{G}$) and incorrect ($\notin \mathcal{G}$) triples. When predicting the head h based on a pair (r, t), we reverse the relation by adding a special symbol to obtain (t, r_{rev}, h) and then train it with the above modules.

For the VAEs, the loss is the sum of the reconstruction loss and kullback–Leibler divergence:

$$\begin{aligned} L_{vae} &= L_{Recon} + L_{KL} \\ &= \tfrac{1}{\mathcal{M}} \textstyle\sum_{i=1}^{\mathcal{M}} \big(\|\mathbf{x_i} - \widetilde{\mathbf{x}}_\mathbf{i}\|^2 + \mathcal{KL}(\mathcal{N}(\mu_{x_i}, \sigma_{x_i}^2), \mathcal{N}(0, 1)) \big) \end{aligned} \tag{20}$$

where $\mathbf{x_i} = \mathbf{h^p}$ for EVAE, and $\mathbf{x_i} = \mathbf{w}^{\rightarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ or $\mathbf{w}^{\leftarrow}_{(\mathbf{h},\mathbf{r},\mathbf{t})}$ for TVAE. Finally, we add the above task and VAE losses as the final loss.

Our VGAT aims to fuse synonymous entities with the VAE densified graph attention model, which can be applied to diverse KG types, such as OpenKGs and CommonsenseKGs, and can be applied to diverse KG-intensive applications, such as link prediction, representation learning and entity canonicalization. In addition, the principle of densification can be applied to sparse and few-shot scenarios. Because our focus in this work is the synonymity of OpenKGs, we design experiments with link prediction tasks in OpenKGs.

5. Proposed cluster ranking protocol

We propose CR, which is a single-test protocol for OpenKGs to evaluate multiple answers caused by synonymity. CR is designed based on the following assumption: scores of synonymous entities are close for any link prediction question, i.e., ranking positions of synonymous entities should be compact. A better model should make the ranking positions of all answer entities, both front and compact. Therefore, CR is proposed from significance (Section 5.1) and compactness (Section 5.2) perspectives, followed by a unified score (Section 5.3) to effectively integrate significance and compactness scores.

First, we provide some definitions. Given a model M to be evaluated, we use a link prediction question $Q_{C_t} = (h, r, ?)$ to evaluate the quality of M and have the known cluster $C_t = \{C_{t,i}\}_{i=1}^{|C_t|}$ as the answer to the question, $\zeta_C = \{C_j\}_{j=1}^{|C(\varepsilon)|}$ as candidate answer clusters and $\zeta_E = \{t_i\}_{i=1}^{|\mathcal{E}|}$ as candidate answer entities. Each candidate answer cluster has at least one candidate answer entity. We then obtain the ranking positions of all candidate answer entities (ζ_E). We input all candidate answer entities to model M and obtain their probability scores. All candidate answer entities are arranged in descending order according to their probability scores such that we obtain the ranking position $P(t_i)$ of each candidate answer entity. The ranking scores of all candidate answer entities are denoted as $P_E = \{P(t_i)\}_{i=1}^{|\varepsilon|}$.

5.1. Significance Cluster Ranking (SCR)

SCR is used to measure whether the ranking positions of all answer entities are front. For each candidate answer cluster $C_j = \{C_{j,i}\}_{i=1}^{|C_j|}$, ranking the scores of all entities are $\{P(C_{j,i})\}_{i=1}^{|C_j|}$ with

Table 1
Cases I–IV. For question $Q_{C_t} = (h, r, ?)$, let $\zeta_E = \{t_1, t_2, \ldots, t_{10}\}$, $\zeta_C = \{C_1, C_2, C_3, C_4\}$, where $C_1 = \{t_1, t_2, t_3\}$, $C_2 = \{t_4, t_5\}$, $C_3 = \{t_6, t_7, t_8, t_9\}$, $C_4 = \{t_{10}\}$. Let the answer cluster be $C_t = C_1 = \{t_1, t_2, t_3\}$. The *"Ranking Positions of Clusters"* column show ranking positions of cluster in ζ_C, where the numbers in brackets are the preliminary scores (Eq. (21)). $SCR(C_1)$ is ranking position of answer cluster C_1. $CCR(C_1)$, $CR(C_1)$ are computed by Eqs. (22) and (23). *MR* is the score of Mention Ranking.

Case	Ranking positions of entities										Ranking positions of clusters				$SCR(C_1)$	$CCR(C_1)$	$CR(C_1)$	*MR*
	1	2	3	4	5	6	7	8	9	10	1	2	3	4				
I	$\boldsymbol{t_2}$	$\boldsymbol{t_3}$	$\boldsymbol{t_1}$	t_6	t_7	t_9	t_5	t_{10}	t_8	t_4	$\boldsymbol{C_1}$(2.0)	C_3(6.0)	C_4(8.0)	C_2(8.5)	1	0	1	1
II	$\boldsymbol{t_1}$	t_4	t_6	t_7	$\boldsymbol{t_3}$	t_{10}	t_8	t_9	$\boldsymbol{t_2}$	t_5	$\boldsymbol{C_1}$(5.0)	C_3(5.5)	C_2(6.0)	C_4(6.0)	1	9	1.64	1
III	t_6	t_7	t_{10}	t_4	t_8	t_5	t_9	$\boldsymbol{t_3}$	$\boldsymbol{t_1}$	$\boldsymbol{t_2}$	C_4(3.0)	C_3(3.8)	C_2(5.0)	$\boldsymbol{C_1}$(9.0)	4	0	4	8
IV	$\boldsymbol{t_1}$	t_6	t_7	t_4	$\boldsymbol{t_3}$	t_8	t_5	t_9	t_{10}	$\boldsymbol{t_2}$	C_3(4.8)	$\boldsymbol{C_1}$(5.3)	C_2(5.5)	C_4(9.0)	2	10	2.71	1

$P(C_{j,i}) \in P_E$. We average these ranking scores of all entities in C_j as the preliminary score of this cluster:

$$P_C(C_j) = \frac{1}{|C_j|}\sum_{i=1}^{|C_j|} P(C_{j,i}),\ C_{j,i} \in C_j \tag{21}$$

Using Eq. (21), we can obtain the preliminary scores for all candidate answer clusters. Then, all candidate answer clusters in ζ_C are arranged in ascending order according to the preliminary scores. After sorting, the ranking position of answer cluster C_t is its final score $SCR(C_t)$.

Note that ranking has two levels: rank entities first and then rank clusters. *SCR* can aggregate the ranking scores of multiple answer entities for a single test triple. Table 1 lists some cases for computing *SCR* scores ($SCR(C_1)$). MR (existing protocol) cannot distinguish lines I, II, and IV, whereas *SCR* illustrates that line IV is worse than lines I and II. However, *SCR* has some shortcomings. For example, lines I and II cannot be distinguished by either *MR* and *SCR*. In the next section, we address this challenge from a compactness perspective.

5.2. Compactness Cluster Ranking (CCR)

CCR measures the proximity of synonymous entities with relative positions; the closer the positions of synonymous entities, the more similar their representations. We use the average difference of all answer entities in the answer cluster as the *CCR* score.[3] For answer cluster $C_t = \{C_{t,i}\}_{i=1}^{|C_t|}$, its *CCR* score is:

$$CCR(C_t) = \left[\sum_{i=1}^{|C_t|}(P(C_{t,i}) - MR(C_t))\right] - \frac{|C_t| * (|C_t| - 1)}{2} \tag{22}$$

where $C_{t,i}$ is the ith entity in answer cluster C_t, $MR(C_t)$ is the MR score that represents the ranking position of an answer entity with the most front position. $\frac{|C_t|*(|C_t|-1)}{2}$ is a dynamic normalization weight used to balance the non-overlapping property caused by multiple answers. With multiple answers, each answer entity has a unique ranking position that cannot be overlapped. This non-overlapping property leads to injustice, i.e., if there are multiple link prediction questions whose numbers of answer entities differ, the *CCR* scores are unfair (if not subtracting a dynamic normalization weight). For example, with two link prediction questions $Q_{C_{t1}}, Q_{C_{t2}}$ and their answer clusters $C_{t1} = \{C_{t1,i}\}_{i=1}^{|C_{t1}|}$, $C_{t2} = \{C_{t2,i}\}_{i=1}^{|C_{t2}|}$, if the ranking positions of all answer entities are closely connected from small to large, e.g. $P_E(C_{t1}) = \{k+i\}_{i=1}^{|C_{t1}|}$, $P_E(C_{t2}) = \{k+i\}_{i=1}^{|C_{t2}|}$, the *CCR* scores of the two questions should all be 0. However, the *CCR* score (if not subtracting a dynamic normalization weight) is not 0 and different, where $CCR(C_{t1}) = \frac{|C_{t1}|*(|C_{t1}|-1)}{2}$, $CCR(C_{t2}) = \frac{|C_{t2}|*(|C_{t2}|-1)}{2}$. Thus, we balance this defect using a dynamic normalization operation.

[3] The importance of each answer entity is equal; thus, selecting the average difference as the metric is sufficient and better than other dispersion functions.

5.3. Final Cluster Ranking (CR)

The *SCR* and *CCR* scores should be integrated to comprehensively evaluate the performance of models. We believe that *SCR* is more important than *CCR*, because compactness makes sense only if the positions of all answer entities are front. Taking cases II and III in Table 1 as examples, the *SCR* of case II is better than that of case III, whereas the *CCR* score of case II is worse than that of case III. In deed, the result of case II {1, 5, 9} is better than that of case III {8, 9, 10}. Therefore, the *SCR* should be considered the main evaluation protocol, the *CCR* should be considered the auxiliary evaluation protocol, and both are indispensable. The main function of *CCR* is used to distinguish cases with identical *SCR* scores, e.g., cases I and II. The final CR score can be computed as:

$$CR(C_t) = SCR(C_t) + \lambda * CCR(C_t) \tag{23}$$

$$\lambda = \frac{1}{\left(\sum_{i=1}^{|C_t|-1}(|\mathcal{E}| - i)\right) - Del(C_t)} \tag{24}$$

where λ is a variable threshold that varies with C_t, and $|\mathcal{E}|$ is the number of all candidate answer entities. The variable threshold λ can ensure that $0 \le \lambda * CCR(C_t) \le 1$ to dominate the rule of *SCR* dominated and *CCR* supplemented. The proof is as follows.

$$\begin{aligned}
&\textstyle\sum_{i=1}^{|C_t|}(P(C_{t,i}) - MR(C_t)) \\
&= \textstyle\sum_{i=1}^{|C_t|-1}(P(C_{t,i}) - MR(C_t)),\ P(C_{t,|C_t|}) = MR(C_t) \\
&= \textstyle\sum_{i=1}^{|C_t|-1} P(C_{t,i}) - \sum_{i=1}^{|C_t|-1} MR(C_t) \\
&\le \textstyle\sum_{i=1}^{|C_t|-1}(|\mathcal{E}| - i + 1) - \sum_{i=1}^{|C_t|-1} MR(C_t) \\
&\le \textstyle\sum_{i=1}^{|C_t|-1}(|\mathcal{E}| - i) + \sum_{i=1}^{|C_t|-1}(1 - MR(C_t)) \\
&\le \textstyle\sum_{i=1}^{|C_t|-1}(|\mathcal{E}| - i)
\end{aligned} \tag{25}$$

$$\lambda * CCR(C_t) = \frac{\left(\sum_{i=1}^{|C_t|}(P(C_{t,i}) - MR(C_t))\right) - \frac{|C_t|*(|C_t|-1)}{2}}{\left(\sum_{i=1}^{|C_t|-1}(|\mathcal{E}| - i)\right) - \frac{|C_t|*(|C_t|-1)}{2}} \le 1 \tag{26}$$

From Section 5.2, $CCR(C_t) \ge 0$ is obvious; thus $\lambda * CCR(C_t) \ge 0$. Essentially, λ represents the worst distribution for $CCR(C_t)$ in which the ranking position of an answer entity is in the position-{1}, and those of other answer entities are in position-$\{|\mathcal{E}| - i + 1\}$. This proof is completed.

Regarding the necessity of merging the *SCR* and *CCR*, consider the following examples. From Table 1, if only *SCR* is selected as an evaluation criterion, the results (*SCR*) of cases I and II are the same; however, this is inconsistent with the polymerization degree (*CCR*) of case II being worse than that of case I. If only the *CCR* is selected as an evaluation criterion, the results (*CCR*) of cases I and III are the same; however, this is inconsistent with the entity positions of case III being worse than those of case I. Therefore, combining *SCR* and *CCR* is indispensable. In addition, we determined that the *CR* is a general version of *ER* in CuratedKGs. When OpenKGs degenerates into CuratedKGs, that is, if each $|C_i| = 1$ then $CR = ER$. Concerning the *MR* which simply considers an answer entity, *SCR* aggregates the scores of multiple answer entities in terms of significance and compactness.

Table 2
Datasets details.

Dataset	ReVerb20K					ReVerb45K				
	100%	80%	60%	40%	20%	100%	80%	60%	40%	20%
Entity	11.1K	8.9K	6.7K	4.4K	2.2K	27.0K	21.6K	16.2K	10.8K	5.4K
Relation	11.1K	6.9K	4.6K	2.2K	0.6K	21.6K	14.7K	9.1K	4.2K	1.2K
Train	15.5K	9.5K	6.1K	2.8K	0.7K	36.0K	23.0K	13.3K	5.7K	1.5K
Valid	1.6K	0.8K	0.5K	0.2K	21	3.6K	2.1K	1.1K	0.4K	70
Test	2.3K	1.3K	0.8K	0.2K	33	5.4K	3.2K	1.7K	0.6K	0.1K
Cluster	10.9K	8.8K	6.6K	4.4K	2.2K	18.6K	16.0K	12.9K	9.2K	5.0K

6. Experiments on protocol and model

6.1. Datasets

Table 2 summarizes the statistics of the datasets. ReVerb45K and ReVerb20K are two benchmark OpenKGs [4], which is constructed using the ReVerb open knowledge base [23]. The entity clusters of ReVerb45K and ReVerb20K are extracted using Freebase entity linking information [11]. The degree distributions are shown in Fig. 4. To test the scalability, we use different numbers of entities [38] to induce four datasets by respectively selecting {20%, 40%, 60%, 80% } of the entities and related links from the original (100%) datasets. Excepting this scalability experiment, the other experiments use 100% of dataset.

6.2. Compared models and settings

To prove the feasibility of the proposed models, we select several latest high-performing baselines:

- General: applies latest models for general KGs to OpenKGs. It includes TransE [5], DistMult [6], ComplEx [7], ConvE [8], ConvTransE [39], KBGAT [9], and GGAE [10].
- OpenKG: comprises the baselines designed for OpenKGs. It includes CaReTransE and CaReConvE [11].
- OpenKG+PLM: uses PLMs to improve the performance of the OpenKG models. It includes OKGIT+Bert and OKGIT+Roberta [12]. To the best of our knowledge, these OKGIT models are state-of-the-art.
- Ours: VGAT is the proposed base model, and VGAT+BERT is our model improved with BERT, which employs the same type of compatibility score with OKGIT+BERT.

All models are implemented on a single GeForce RTX 2080 GPU. The baselines are reproduced based on their open-source implementations. Concretely, we reproduce TransE, DistMult and ComplEx from public code at,[4] ConvE and ConvTransE from public code at,[5] KBGAT and GGAE from public code at.[6] We use the grid search technique to select the optimal values of the hyperparameters for the baselines. For the OpenKG and OpenKG+PLM baselines, CaReTransE, CaReConvE are reproduced from the public code at[5], OKGIT+Bert and OKGIT+Rob are reproduced from the public code at.[7] The optimal values of the hyperparameters for these four baselines are consistent with those in their respective papers. The proposed VGAT and VGAT+BERT are implemented based on the PyTorch library and trained in two stages to accelerate and improve performance. We first train a model without densification, then train densification under the pretrained model. The optimizer is set to Adam, the embedding size is set to 300, and entity and relation embeddings are randomly initialized. The sequence module is set to BiGRU and word vectors are initialized with GloVe embeddings.[8] The aggregation operations is addition for ReVerb20K and concatenation for ReVerb45K. In the neighbor synonymity encoder and densification modules, the maximum number N_r of adjacent relations is searched from {2, 4}, and the maximum number N_e of adjacent entities is searched from {8, 16, 32}.[9] If the number of true adjacent entities is less than N_e, the embeddings generated by the densification module are automatically used to complement. That is, the number of embeddings generated by the densification module is specific to each entity: 0 if $N_e > |\mathcal{E}^{\rightarrow}_{(h,r_j)}|$, and $N_e - |\mathcal{E}^{\rightarrow}_{(h,r_j)}|$ otherwise; 0 if $N_e > |\mathcal{E}^{\leftarrow}_{(r_j,h)}|$, and $N_e - |\mathcal{E}^{\leftarrow}_{(r_j,h)}|$ otherwise, where $|\mathcal{E}^{\rightarrow}_{(h,r_j)}|$ and $|\mathcal{E}^{\leftarrow}_{(r_j,h)}|$ are the number of true adjacent entities for pairs (h, r_j) and (r_j, h), respectively. The other hyper-parameters include the learning rate $\in$ {0.001, 0.0001, 0.00008, 0.00005, 0.00001} and batch size $\in$ {32, 64, 128, 256, 512}. We use grid search to select the optimal hyperparameters based on the performance on the validation set. The optimal maximum numbers of adjacent (relations, entities) are (2, 32) for ReVerb20K and (4, 8) for ReVerb45K. The optimal value of the learning rate is 8e-5 for ReVerb20K and 5e-5 for ReVerb45K; batch size is 64 for ReVerb20K and 128 for ReVerb45K.

6.3. Compared protocols

The proposed CR protocol is designed for a single test triple. To the best of our knowledge, the existing single-test protocol which can evaluate multiple answers for OpenKGs is only the MR protocol. Therefore, we use the MR as the baseline protocol to compare with the CR. Each model is evaluated by both MR and CR protocols. Then, the MR scores of all test triples are aggregated with the all-test protocols: *AR*, *ARR*, and *H@N*; similarly for CR scores. A model with better performance should have lower *AR*, higher *ARR* and higher *H@N* for both MR and CR. The remainder of this paper describes the results in terms of model efficiency (Section 6.4) and protocol feasibility (Section 6.5).

6.4. Results on model

In this section, we evaluate the effectiveness of the proposed models. The results evaluated with MR and CR are given in Tables 3 and 4. We observed that the proposed models achieve substantial improvements compared with the baselines. For ReVerb20K in Table 3, the proposed VGAT achieves substantial improvements on all metrics in comparison to General and OpenKG baselines, notably 3.0, 1.2, 6.5, 5.7, 5.9 points in ARR, *H*@1,10,50,100 of MR and 4.4, 3.5, 7.7, 10.3, 8.1 points in ARR, *H*@1,10,50,100 of CR. In addition, VGAT+BERT outperforms the OpenKG+PLM baselines with strong improvements on most

[4] https://github.com/uma-pi1/kge.
[5] https://github.com/malllabiisc/CaRE.
[6] https://github.com/feiwangyuzhou/GGAE.
[7] https://github.com/Chandrahasd/OKGIT.

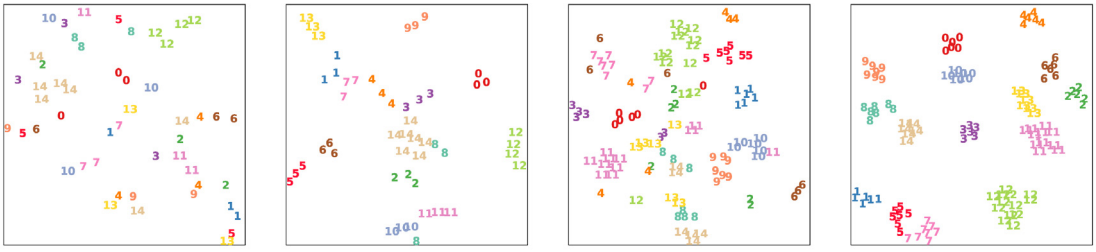
[8] VGAT+BERT uses BiGRU to encode sequences in the phrasal synonymity encoder and employs BERT to compute type compatibility scores similar to OKGIT+BERT.
[9] The range of the maximum number of adjacent relations can be selected according to the average and maximum degrees of relations, similarly for adjacent entities.

Table 3
Results with the MR and CR on ReVerb20K to compare the baselines and proposed models; the best score is in bold.

Type	Model	Mention ranking						Cluster ranking					
		AR	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100	*AR*	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100
General	TransE	1497	13.3	2.2	29.6	43.0	49.2	1934	9.1	9.7	20.7	30.4	34.9
	DistMult	4569	1.9	1.3	2.7	5.2	7.1	5309	0.6	0.4	1.0	1.8	2.8
	ComplEx	4376	2.0	1.4	3.0	5.6	7.7	5088	0.5	0.3	1.0	2.3	3.2
	ConvE	1085	25.5	19.9	35.8	50.1	57.2	1212	17.9	15.5	27.0	40.6	49.2
	ConvTransE	1080	26.1	20.5	35.9	50.0	57.1	1231	18.1	16.0	26.5	39.6	47.8
	KBGAT	1180	25.7	20.1	35.9	51.2	58.5	1339	18.7	12.5	27.2	42.0	50.7
	GGAE	1106	28.1	21.9	39.7	54.7	62.8	1152	21.5	14.3	30.7	46.1	56.3
OpenKG	CaReTransE	950	30.3	23.2	42.8	58.4	64.6	1004	23.3	21.5	34.4	48.4	58.5
	CaReConvE	801	31.6	25.6	42.9	56.7	63.4	1064	23.8	21.4	34.8	45.8	51.7
Ours	**VGAT**	587	34.6	26.8	49.4	64.1	70.5	623	28.2	25.0	42.5	58.7	66.6
	VGAT+BERT	**445**	**37.3**	**28.8**	**53.3**	**70.0**	**73.7**	641	**29.2**	**26.8**	**43.0**	59.6	67.5
OpenKG + PLM	OKGIT+BERT	524	35.1	27.5	49.5	65.9	72.7	**590**	28.1	25.1	41.7	**59.9**	**68.1**
	OKGIT+Rob	594	35.8	28.4	49.2	65.4	72.1	682	28.1	24.8	40.7	58.2	67.3

Table 4
Results with the MR and CR on ReVerb45K to compare the baselines and proposed models; the best score is in bold.

Type	Model	Mention Ranking						Cluster Ranking					
		AR	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100	*AR*	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100
General	TransE	2222	15.8	9.3	25.9	37.1	43.2	4366	3.6	3.1	5.8	9.1	11.2
	DistMult	5782	8.5	7.7	9.7	12.0	13.6	8455	0.9	0.8	1.1	1.6	2.1
	ComplEx	5173	8.9	7.5	11.3	16.0	18.9	8194	1.5	1.3	2.3	3.5	4.3
	ConvE	2483	22.1	16.6	32.4	43.3	47.9	4276	6.4	5.5	9.7	13.6	16.0
	ConvTransE	2490	23.4	17.9	33.8	44.4	48.8	4262	6.8	5.9	10.2	14.3	16.7
	KBGAT	2392	25.9	19.4	38.1	49.2	53.1	3936	9.1	4.7	13.9	19.8	22.5
	GGAE	2348	24.9	17.2	38.7	50.7	54.7	3947	9.3	4.9	14.5	21.1	24.0
OpenKG	CaReTransE	2414	19.5	7.8	37.5	47.5	51.4	4283	7.1	6.6	11.0	15.2	17.8
	CaReConvE	1589	29.7	23.4	41.3	53.6	58.7	3125	9.1	7.9	13.6	19.1	22.0
Our	**VGAT**	1093	32.9	25.7	47.0	59.8	64.8	2369	11.7	10.2	17.2	24.6	28.8
	VGAT+BERT	**696**	**33.8**	26.5	**48.0**	**61.7**	**67.5**	1662	11.7	9.8	17.5	26.0	31.0
OpenKG + PLM	OKGIT+BERT	735	33.7	**26.7**	47.1	59.8	65.2	**1552**	**15.3**	**13.4**	**21.9**	**29.5**	**33.7**
	OKGIT+Rob	849	33.4	26.5	46.4	58.8	63.9	1753	14.5	12.7	20.6	28.2	32.2



(a) RV20-CaReConvE (b) RV20-VGAT (c) RV45-CaReConvE (d) RV45-VGAT

Fig. 3. Visualization of CaReConvE and VGAT on ReVerb20K (RV20) and ReVerb45K (RV45), for which the same numbers denote similar entities.

metrics—the *H*@10 metric increases by 3.8 points of MR and 1.3 points of CR. For ReVerb45K in Table 4, VGAT achieves better results than the General and OpenKG baselines on all metrics, notably 3.2, 2.3, 5.7, 6.2, 6.1 points in ARR, *H*@1,10,50,100 of MR and 2.6, 2.3, 3.6, 5.5, 6.8 points in ARR, *H*@1,10,50,100 of CR. In addition, VGAT+BERT outperforms the OpenKG+PLM baselines with strong improvements in terms of the MR—the *H*@10,50,100 metrics increases by 0.9, 1.9, 2.3 points, respectively. However, when using the CR protocol, the performance of OKGIT+BERT is better than that of VGAT+BERT. This interesting phenomenon highlights the challenge of existing protocols and is analyzed in Section 6.5. In summary, the proposed VGAT and VGAT+BERT models can capture textual features using sequence technologies and structural features using a graph attention network, which makes synonymous entities closer in their spatial distances.

Visualization To qualitatively show that VGAT can make entities with the same meaning closer in the spatial distribution, we show the t-SNE visualization [40] in Fig. 3. For this visualization experiment, we selected 15 entity clusters for each dataset, in which each cluster has more than three entities, and the same numbers represent entities in the same entity cluster with the same meaning. CaReConvE is used as a baseline to compare with VGAT; both of them do not use PLMs. By observing the distributions of entities in Fig. 3, we determined that the embeddings of entities with the same meaning are closer in the proposed VGAT than whose in CaReConvE. The qualitative results are consistent with the quantitative results in Tables 3 and 4, which further verify the effectiveness of the proposed model.

Ablation Study To fully verify the effectiveness of the VGAT components, we conduct ablation study removing the following aspects from the *Full* model:

-*PSE*: removes the phrasal synonymity encoder (Section 4.1), which removes the phrasal synonymity embeddings $\mathbf{h}^{\mathbf{p}}$ and $\mathbf{r}^{\mathbf{p}}$.

-*NSE*: removes the neighbor synonymity encoder (Section 4.2), which removes the neighbor synonymity embeddings $\mathbf{h}_{\mathbf{w}}^{\overrightarrow{}}$, $\mathbf{h}_{\mathbf{p}}^{\overrightarrow{}}$, $\mathbf{h}_{\mathbf{w}}^{\overleftarrow{}}$, $\mathbf{h}_{\mathbf{p}}^{\overleftarrow{}}$.

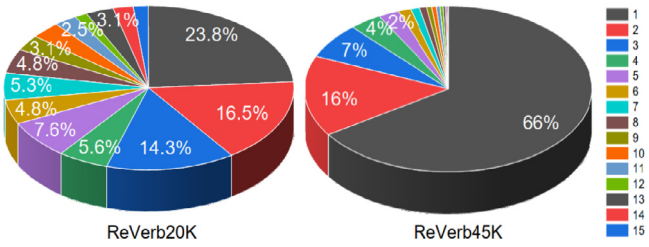


Fig. 4. Entity degree distributions; 1–15 indicate the degree.

-*VAE*: removes the VAE densification module (Section 4.3).
-*NEG*: removes the negative densification module (Section 4.3)
.
-*OUT*: removes outgoing neighbors (Section 4.2), where the embeddings $\mathbf{h}_{\mathbf{e}}^{\rightarrow}$, $\mathbf{h}_{\mathbf{p}}^{\rightarrow}$ of outgoing neighbors are removed.
-*IN*: removes incoming neighbors (Section 4.2), where the embeddings $\mathbf{h}_{\mathbf{w}}^{\leftarrow}$, $\mathbf{h}_{\mathbf{p}}^{\leftarrow}$ of incoming neighbors are removed.
-*TriW*: removes the representation $c_{(h,r,t)}^{\rightarrow,\mathbf{w}}$ and $c_{(h,r,t)}^{\leftarrow,\mathbf{w}}$ of each triple in Section 4.1, where the neighbor synonymity embeddings $\mathbf{h}_{\mathbf{w}}^{\rightarrow}$, $\mathbf{h}_{\mathbf{w}}^{\leftarrow}$ in Section 4.2 are removed.
-*TriP*: removes the representation $c_{(h,r,t)}^{\rightarrow,\mathbf{p}}$ and $c_{(h,r,t)}^{\leftarrow,\mathbf{p}}$ of each triple in Section 4.1, which involves removing the neighbor synonymity embeddings $\mathbf{h}_{\mathbf{p}}^{\rightarrow}$ and $\mathbf{h}_{\mathbf{p}}^{\leftarrow}$ (Section 4.2) .

The results of ablation study with MR are shown in Fig. 5. The scores of removing any component are weaker than those of *Full*, which proves the effectiveness of each component. In particular, -*PSE* is significantly weaker than *Full* on both datasets, which shows the indispensability of the phrasal synonymity encoder. -*NSE* is significantly weaker than *Full* on ReVerb20K and slightly weaker than *Full* on ReVerb45K. By observing the distribution of datasets (Fig. 4), we determined that ReVerb45K is sparser, in which 82% of entities have a degree lower than two, which makes attention ineffective in introducing considerable improvements on ReVerb45K. The densification modules with generated embeddings (-*VAE*) and negative samples (-*NEG*) play an important role in improving performance. The performance degradation of -*OUT* and -*IN* shows that outgoing and incoming neighbors can introduce different features to understanding each entity. The results of -*TriW* and -*TriP* indicate the capability in capturing directed text features.

Efficiency and Scalability We analyze efficiency and scalability of VGAT with time complexity, running time and performance. We first analyze the time complexity of VGAT, which primarily includes the phrasal synonymity encoder (Eqs. (1)–(3)), neighbor synonymity encoder (Eqs. (6)–(8)), and densification (Eqs. (9)–(15)) modules, as shown in Fig. 2. For the phrasal synonymity encoder, Eq. (1) requires $\mathcal{O}(N_s \cdot T^2)$ for each object, as well as Eqs. (2) or (3), in which T is the feature dimension and N_s is the maximum number of words for entities and relations. For the neighbor synonymity encoder, each object entity has at most N_r adjacent relations, each object entity-relation pair has at most N_e adjacent entities, so phrasal synonymity encoder of these adjacent relations and entities require $\mathcal{O}(N_r \cdot N_e \cdot N_s \cdot T^2)$. And Eqs. (6)–(8) require $\mathcal{O}(T^2)+\mathcal{O}(N_r \cdot N_e \cdot T)$. For densification module, the maximum number of generated embeddings is $N_r \cdot N_e$ in the worst case, so Eqs. (9)–(15) require $\mathcal{O}(N_r \cdot N_e \cdot T \cdot T_v)$ with T_v as the internal feature dimension of VAE. Overall, each object triple requires $\mathcal{O}(N_s \cdot T^2) + \mathcal{O}(N_r \cdot N_e \cdot N_s \cdot T^2) + \mathcal{O}(T^2) + \mathcal{O}(N_r \cdot N_e \cdot T) + \mathcal{O}(N_r \cdot N_e \cdot T \cdot T_v)$, omitted and abbreviated as $\mathcal{O}((1 + N_r N_e)N_s T^2 + N_r N_e T T_v)$. The state-of-the-art baseline CaReConvE has a phrasal encoder ($\mathcal{O}(N_s \cdot T^2)$) and attention encoder ($\mathcal{O}(T^2) + \mathcal{O}(N_r \cdot N_e \cdot N_s \cdot T^2) + \mathcal{O}(N_r \cdot N_e \cdot T)$), so its time complexity is abbreviated as $\mathcal{O}((1 + N_r N_e)N_s T^2)$. Our model has the same time complexity as CaReConvE in the best case when there are sufficient adjacent entities and relations. In general scenarios, our model is more complex but significantly improves the performance compared with CaReConvE.

We test and compare the scalability of VGAT by reporting the running time and performance on differently scaled data [38]. We use different numbers of entities to induce five datasets by respectively selecting {20%, 40%, 60%, 80%, 100%} of the entities and related links from the original datasets. Table 2 details the scaled data. We report the running time in Fig. 6(a) and performance in Fig. 6(b). Consistent with the analysis of time complexity, the running time of VGA is longer than that of CaReConvE. Fortunately, the performance of our VGAT is significantly better than CaReConvE on differently scaled data, particularly as the *H*@100 increases by 5.4, 4.8, 4.7, 4.3, 7.1 points on 20%, 40%, 60%, 80%, 100% ReVerb20K datasets.

6.5. Results on protocol

We now analyze whether the proposed CR protocol exposes phenomena that cannot be reflected by existing evaluation protocol. The results of the existing protocol (MR) and proposed protocol (CR) are compared to analyze abnormal phenomena. We observed that the CR is more pronounced on ReVerb45K than on ReVerb20K. This is because of the few clusters with multiple entities on ReVerb20K. Table 2 shows that the proportion of entity clusters with multiple entities on ReVerb20K ((11.1K-10.8K)/10.8K = 3%) is smaller than that on ReVerb45K ((27.0K-18.6K)/18.6 = 45%). The ability of CR is to predict synonymous entities. With few clusters with multiple entities, CR is almost equivalent to MR. Next, we primarily analyze the results on ReVerb45K. Table 5 presents the head/tail results of the MR and CR on ReVerb45K.

First, we analyze abnormal phenomena between different models on the MR and CR. We compare ConvE with CaReTransE. The results on the *ARR* metric of MR is 22.9/21.2 for ConvE and 20.4/18.6 for CaReTransE, and the results on the *H*@1 metric of MR is 17.8/15.3 for ConvE and 8.0/7.6 for CaReTransE, indicating that ConvE is superior to CaReTransE on the *ARR*, *H*@1 metrics of MR. When considering the *H*@10,50,100 metrics of MR, the results show the superiority of CaReTransE over ConvE. This is contradictory which indicates the two models unable to compare.

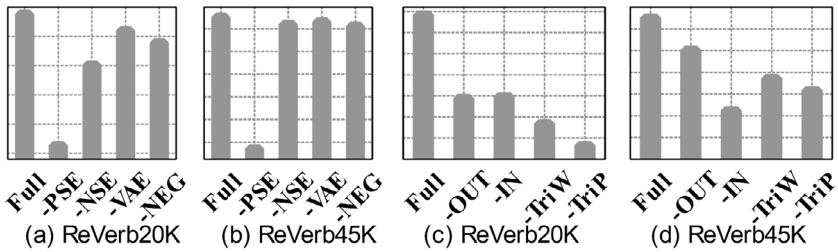


Fig. 5. Results of ablation study on (a) ReVerb20K and (b) ReVerb45K.

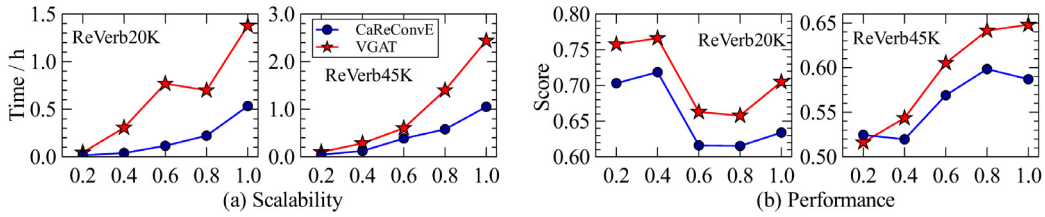


Fig. 6. Running time (h) and performance (*H*@100/MR) on differently scaled data.

Table 5
Head/tail results on ReVerb45K to compare the MR and CR protocols.

Model	Mention ranking						Cluster ranking					
	AR	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100	*AR*	*ARR*	*H*@1	*H*@10	*H*@50	*H*@100
TransE	2203/2240	16.2/15.4	9.5/9.1	26.1/25.8	35.9/38.3	41.4/45.0	4809/3924	2.0/5.2	1.6/4.6	3.2/8.4	5.4/12.8	7.0/15.5
DistMult	5139/6425	10.4/6.6	9.5/5.8	11.6/7.8	13.9/10.1	15.6/11.7	8350/8560	0.7/1.1	0.6/1.0	0.9/1.4	1.1/2.1	1.4/2.8
ComplEx	4687/5659	10.6/7.1	9.4/5.7	12.8/9.7	17.3/14.7	20.1/17.7	8238/8149	1.1/2.0	0.8/1.7	1.6/3.0	2.3/4.7	2.9/5.8
ConvTransE	2403/2577	24.4/22.4	18.9/16.8	34.3/33.4	44.6/44.2	48.9/48.7	4498/4026	4.9/8.6	3.9/8.0	7.5/12.9	10.8/17.7	13.1/20.2
CaReConvE	1531/1648	31.4/27.9	25.5/21.3	42.3/40.2	53.9/53.4	58.6/58.9	3263/2987	7.3/11.0	5.9/10.0	10.8/16.3	15.8/22.4	18.3/25.6
KBGAT	1891/2892	28.5/23.3	22.2/16.7	40.3/35.8	51.3/47.0	55.0/51.1	3502/4369	8.3/10.0	3.3/6.1	12.7/15.0	18.7/20.9	21.5/23.4
GGAE	2042/2654	27.2/22.5	19.7/14.7	40.9/36.6	52.4/49.0	56.4/52.9	3729/4164	8.2/10.4	3.2/6.6	13.2/15.8	19.9/22.4	22.9/25.0
VGAT	1086/1099	33.3/32.5	26.7/24.7	46.2/47.9	58.2/61.3	62.7/66.8	2420/2319	8.9/14.4	7.0/13.4	13.2/21.3	20.6/28.5	24.7/32.9
OKGIT+Rob	757/942	33.6/33.2	27.4/25.7	45.5/47.4	57.1/60.6	62.4/65.4	1627/1880	13.6/15.4	11.3/14.1	18.6/22.7	26.1/30.3	30.0/34.4
ConvE	2406/2560	22.9/21.2	17.8/15.3	32.8/32.1	42.9/43.8	47.6/48.2	4599/3953	4.5/8.3	3.5/7.5	6.7/12.6	9.9/17.3	11.8/20.1
CaReTransE	2140/2688	20.4/18.6	8.0/7.6	38.8/36.2	48.6/46.4	52.4/50.3	4437/4130	4.9/9.4	4.3/8.9	8.0/14.0	11.5/18.9	13.7/22.0
VGAT+BERT	636/755	34.6/33.0	28.0/25.1	48.0/48.1	61.1/62.3	66.5/68.5	1499/1825	9.8/13.6	7.4/12.1	14.7/20.2	22.9/29.0	28.3/33.6
OKGIT+BERT	638/832	33.9/33.5	27.3/26.0	46.2/47.9	58.8/60.7	63.9/66.4	1410/1694	14.4/16.1	12.1/14.8	20.5/23.3	27.6/31.5	31.6/35.9

Table 6
Statistical analysis.

Datasets	Test cases	Cases with more than one answers					
		Head	Tail	$s_1^{MR} = s_2^{MR}$		$s_1^{MR} = s_2^{MR}$, $s_1^{CR} \neq s_2^{CR}$	
				Head	Tail	Head	Tail
ReVerb20K	2325	1423	60	470	5	468(99.6%)	5(100%)
ReVerb45K	5395	4990	3528	1100	740	1100(100%)	739(99.9%)

As it happens, CR can solve this contradiction. The results on the *ARR* metric of CR is 4.5/8.3 for ConvE and 4.9/9.4 for CaReTransE, and the results on the *H*@1 metric of CR is 3.5/7.5 for ConvE and 4.3/8.9 for CaReTransE, indicating a consistent trend with the *H*@10,50,100 metrics of CR. That is, when evaluating all answer entities instead of a selected entity, CaReTransE outperforms ConvE. Consider the following comparison; VGAT+BERT outperforms OKGIT+BERT when using MR, but is weaker than OKGIT+BERT when using CR. That is, the ability of OKGIT+BERT to cluster similar entities is stronger than that of VGAT+BERT.

In addition to comparing the different models, the ability of the same model to predict head and tail answers differs on MR and CR. Taking ComplEx as an example, the results of MR show that ComplEx has a better ability to predict head answers than tail answers. However, when evaluating all answer entities with CR, the results completely contradict this reasoning. The MR, which evaluates a selected entity, often provides unfair scores for the link prediction task of OpenKGs with multiple answers. The proposed CR effectively exposes phenomena that cannot be reflected by the existing protocol and can evaluate the models from multiple perspectives.

Statistical Analysis We further prove the superiority of CR over MR with a statistical analysis. This part is designed as follows. For a test case in c^{All} (set of cases that have more than one answer entities) and a model to be evaluated, we obtain prediction scores using the model to compute the MR (s^{MR}) and CR (s^{CR}) scores. For two different models, e.g., CaReConvE and VGAT, the MR and CR scores (evaluate the same test case) are denoted as s_1^{MR}, s_1^{CR} for CaReConvE and s_2^{MR}, s_2^{CR} for VGAT, respectively. For each test case, we compare the MR scores of the two models and apply the case to a set c^{MR} if $s_1^{MR} = s_2^{MR}$ and then compare the CR scores of the two models and apply the case to a set c^{CR} if $s_1^{MR} = s_2^{MR}$ and $s_1^{CR} \neq s_2^{CR}$. The proportion of cases that MR cannot evaluate is $|c^{MR}|/|c^{All}|$, and the proportion of cases that MR cannot evaluate but CR can evaluate correctly is: $|c^{CR}|/|c^{MR}|$.

According to the statistics in Table 6, the proportion of cases that MR cannot evaluate is 470/1423 = 33.0% (Head) and 5/60 = 8.3% (Tail) on ReVerb20K and 1100/4990 = 22.0% (Head) and 740/3528 = 21.0% (Tail) on ReVerb45K. This shows considerable number of cases that MR cannot evaluate correctly; thus, designing the CR is necessary. The proportion of cases that MR cannot evaluate but CR can evaluate correctly is: 468/478 = 99.6% (Head) and 5/5 = 100% (Tail) on ReVerb20K and 1100/1100 = 100% (Head) and 739/740 = 99.9% (Tail) on ReVerb45K. Thus, CR can effectively distinguish cases that MR cannot evaluate, thereby proving the feasibility of CR.

Case Study We select some cases to further explain the working principle of CR in Table 7. Cases I–III show some cases with the same MR score and different CR scores, that is $s_1^{MR} = s_2^{MR}$ and $s_1^{CR} \neq s_2^{CR}$. Cases I–II predict the head entities according to the relation and tail entity, and case III predicts the tail entities according to the head entity and relation. Taking case I as an exemplary example, the model predicts the head entities according to the relation *"was developed by"* and tail entity *"texas instrument"*. The answer entities include *"digital light processing"* and *"dlp"* (as shown in the "Answer Entity" column). The "Position" column provides the ranking positions of the answer entities. Based on the "Position", MR is the minimum ranking position of the answer entities, and CR can be computed by Eq. (23). For case I, the MR scores of CaReConvE and VGAT are all 1, and the CR

Table 7
Case study of real cases predicted by CaReConvE (A) and VGAT (B). Cases I–III are cases that CR can evaluate correctly but MR cannot, where I–II are for the head and III is for the tail. Cases IV–VI are possible incorrect cases.

Case	Triple	Answer entity	Position		MR		CR	
			A	B	A	B	A	B
I	(?, *was developed by*, *texas instrument*)	*Digital light processing* *dlp*	0 2	0 1	1	1	1.00004	1.0
II	(?, *say of*, *nixon*)	*Dwight eisenhower* *Eisenhower*	1 7	1 5	2	2	4.00045	2.00027
III	(*rudolph giuliani*, *is a hero in*, ?)	*Nyc* *New york* *New york city* *Ny city* *New yawk*	7 45 2652 2888 5963	7 17 27 635 4477	8	8	1064	659
IV	(?, *smile at*, *sakura*)	*Uchiha* *Sasuke uchiha* *Sasuke* *Uchiha sasuke*	0 8 9 14	0 1 4 26	1	1	4.00075	4.00075
V	(*ethiopia*, *is in*, ?)	*East africa* *Eastern africa*	0 2	1 3	1	1	1.00004	1.00004
VI	(?, *be in*, *white house*)	*Clinton* *Bill clinton*	0 1	1 2	1	1	1.0	1.0

score of CaReConvE is 1.00004 and that of VGAT is 1.0. That is, for case I, the two models cannot be distinguished with MR, while VGAT is slightly better than CaReConvE when using CR. Therefore, CR can account for the shortcomings of MR. Please refer to Table 7 for cases II–III in detail.

Finally, we analyze three possible incorrect cases with the same MR and CR scores, which are shown in cases IV–VI in Table 7. For case IV, according to the "Position" of CaReConvE and VGAT, the *SCR* and *CCR* scores of these models are the same; thus, CR cannot distinguish them. However, for cases V and VI, the reason for the indistinguishability is the *filtering* mechanism, not a limitation of CR. The *filtering* mechanism is a general strategy in the test stage, which filters the triples that appear in the training set. Because a triple that appears in the training set is filtered out, the ranking positions of VGAT change from {1, 3} to {0, 2} for case V ({1, 2} to {0, 1} for case VI). Based on this, the ranking positions of VGAT are the same as those of CaReConvE; thus, VGAT and CaReConvE have the same effect, and the evaluation results of MR and CR are completely correct for cases V and VI.

In summary, this analysis proves the necessity and effectiveness of the proposed CR protocol.

7. Conclusion

In this paper, we provide empirical insights into unique challenges of investigating an effective model and evaluation protocol for OpenKGs. We propose VGAT for automatically mining synonymous features of OpenKGs. In addition, we propose the CR protocol to solve the challenge of evaluating multiple answers. Extensive experiments and comprehensive analysis show the effectiveness of VGAT and rationality of CR. VGAT augments the representation space with VAE variants in an implicit manner, we plan to consider its explicit explainability in future work. We hope to propose a comprehensive protocol for the evaluation of case V in Table 7. Using the CR in tasks beyond link prediction is another direction for further investigation.

CRediT authorship contribution statement

Qian Li: Conceptualization, Methodology, Software, Writing – original draft. **Daling Wang:** Funding acquisition, Project administration, Writing – review & editing. **Shi Feng:** Methodology, Formal analysis. **Kaisong Song:** Methodology, Formal analysis. **Yifei Zhang:** Formal analysis. **Ge Yu:** Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Thanks to the reviewers for their careful review. Thanks to all co-authors for their hard work. The work is supported by National Natural Science Foundation of China (62172086, 61872074, 62106039, 62272092), and Chinese Scholarship Council.

References

[1] S. Saha, Mausam, Open information extraction from conjunctive sentences, in: Proceedings of the 27th International Conference on Computational Linguistics, COLING, Santa Fe, New Mexico, USA, August 20-26, 2018, 2018, pp. 2288–2299.
[2] K. Gashteovski, S. Wanner, S. Hertling, S. Broscheit, R. Gemulla, OPIEC: An open information extraction corpus, in: 1st Conference on Automated Knowledge Base Construction, AKBC, Amherst, MA, USA, May 20-22, 2019, 2019, URL: https://doi.org/10.24432/C53W2J.
[3] S. Broscheit, K. Gashteovski, Y. Wang, R. Gemulla, Can we predict new facts with open knowledge graph embeddings? A benchmark for open link prediction, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL, Online, July 5-10, 2020, 2020, pp. 2296–2308, URL: https://doi.org/10.18653/v1/2020.acl-main.209.
[4] S. Vashishth, P. Jain, P.P. Talukdar, CESI: Canonicalizing open knowledge bases using embeddings and side information, in: Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW, Lyon, France, April 23-27, 2018, 2018, pp. 1317–1327, URL: https://doi.org/10.1145/3178876.3186030.
[5] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems. Proceedings of a Meeting Held December 5-8, 2013, Lake Tahoe, Nevada, United States, 2013, pp. 2787–2795.
[6] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
[7] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: Proceedings of the 33nd International Conference on Machine Learning, ICML, New York City, NY, USA, June 19-24, 2016, in: JMLR Workshop and Conference Proceedings, vol. 48, 2016, pp. 2071–2080.

[8] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2D knowledge graph embeddings, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, 2018, pp. 1811–1818.
[9] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in: Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers, 2019, pp. 4710–4723, URL: https://doi.org/10.18653/v1/p19-1466.
[10] Q. Li, D. Wang, S. Feng, C. Niu, Y. Zhang, Global graph attention embedding network for relation prediction in knowledge graphs, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[11] S. Gupta, S. Kenkre, P.P. Talukdar, CaRe: Open knowledge graph embeddings, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, Hong Kong, China, November 3-7, 2019, 2019, pp. 378–388, URL: https://doi.org/10.18653/v1/D19-1036.
[12] Chandrahas, P.P. Talukdar, OKGIT: Open knowledge graph link prediction with implicit types, in: Findings of the Association for Computational Linguistics: ACL/IJCNLP, Online Event, August 1-6, 2021, pp. 2546–2559, URL: https://doi.org/10.18653/v1/2021.findings-acl.225.
[13] C. Xu, W. Zhao, J. Zhao, Z. Guan, X. Song, J. Li, Uncertainty-aware multi-view deep learning for Internet of Things applications, IEEE Trans. Ind. Inf. (2022).
[14] D. Li, Z. Gu, Y. Wang, C. Ren, F.C. Lau, One model packs thousands of items with Recurrent Conditional Query Learning, Knowl.-Based Syst. 235 (2022) 107683.
[15] Y. Wu, G. Tie, Y. Yu, J. Li, J. Song, EBSS: A secure blockchain-based sharing scheme for real estate financial credentials, World Wide Web (2022).
[16] T. Cai, S. Yang, J. Li, Q.Z. Sheng, J. Yang, X. Wang, W.E. Zhang, L. Gao, Incremental graph computation: Anchored vertex tracking in dynamic social networks, IEEE Trans. Knowl. Data Eng. (2022).
[17] G. Xue, M. Zhong, J. Li, J. Chen, C. Zhai, R. Kong, Dynamic network embedding survey, Neurocomputing 472 (2022) 212–223.
[18] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, A. Mian, Bi-CLKT: Bi-graph contrastive learning based knowledge tracing, Knowl.-Based Syst. 241 (2022) 108274.
[19] X. Song, J. Li, Y. Tang, T. Zhao, Y. Chen, Z. Guan, Jkt: A joint graph convolutional network based deep knowledge tracing, Inform. Sci. 580 (2021) 510–523.
[20] M. Zhang, G. Wang, L. Ren, J. Li, K. Deng, B. Zhang, METoNR: A meta explanation triplet oriented news recommendation model, Knowl.-Based Syst. 238 (2022) 107922.
[21] H. Yin, X. Song, S. Yang, J. Li, Sentiment analysis and topic modeling for COVID-19 vaccine discussions, World Wide Web 25 (3) (2022) 1067–1083.
[22] O. Etzioni, A. Fader, J. Christensen, S. Soderland, Mausam, Open information extraction: The second generation, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, 2011, pp. 3–10, URL: https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-012.
[23] A. Fader, S. Soderland, O. Etzioni, Identifying relations for open information extraction, in: Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, a Meeting of SIGDAT, a Special Interest Group of the ACL, 2011, pp. 1535–1545.
[24] Mausam, M. Schmitz, S. Soderland, R. Bart, O. Etzioni, Open language learning for information extraction, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL, July 12-14, 2012, Jeju Island, Korea, 2012, pp. 523–534.
[25] L.D. Corro, R. Gemulla, ClausIE: clause-based open information extraction, in: 22nd International World Wide Web Conference, WWW, Rio de Janeiro, Brazil, May 13-17, 2013, 2013, pp. 355–366, URL: https://doi.org/10.1145/2488388.2488420.
[26] H. Pal, Mausam, Demonyms and compound relational nouns in nominal open IE, in: Proceedings of the 5th Workshop on Automated Knowledge Base Construction, AKBC@NAACL-HLT, San Diego, CA, USA, June 17, 2016, 2016, pp. 35–39, URL: https://doi.org/10.18653/v1/w16-1307.
[27] S. Saha, H. Pal, Mausam, Bootstrapping for numerical open IE, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers, 2017, pp. 317–323, URL: https://doi.org/10.18653/v1/P17-2050.
[28] L.A. Galárraga, C. Teflioudi, K. Hose, F.M. Suchanek, AMIE: association rule mining under incomplete evidence in ontological knowledge bases, in: 22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, 2013, pp. 413–422, URL: https://doi.org/10.1145/2488388.2488425.
[29] L. Galárraga, G. Heitz, K. Murphy, F.M. Suchanek, Canonicalizing open knowledge bases, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014, 2014, pp. 1679–1688, URL: https://doi.org/10.1145/2661829.2662073.
[30] T. Jiang, T. Zhao, B. Qin, T. Liu, N.V. Chawla, M. Jiang, Canonicalizing open knowledge bases with multi-layered meta-graph neural network, 2020, CoRR abs/2006.09610.
[31] S. Tiwari, I. Bansal, C.R. Rivero, Revisiting the evaluation protocol of knowledge graph completion methods for link prediction, in: WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021, 2021, pp. 809–820, URL: https://doi.org/10.1145/3442381.3449856.
[32] R. Lian, M. Xie, F. Wang, J. Peng, H. Wu, Learning to select knowledge for response generation in dialog systems, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, 2019, pp. 5081–5087, URL: https://doi.org/10.24963/ijcai.2019/706.
[33] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, a Meeting of SIGDAT, a Special Interest Group of the ACL, 2014, pp. 1724–1734, URL: https://doi.org/10.3115/v1/d14-1179.
[34] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[35] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186, URL: https://doi.org/10.18653/v1/n19-1423.
[36] D.P. Kingma, M. Welling, Auto-encoding variational Bayes, in: Y. Bengio, Y. LeCun (Eds.), 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, 2014.
[37] S. Yang, S. Verma, B. Cai, J. Jiang, K. Yu, F. Chen, S. Yu, Variational Co-embedding learning for attributed network clustering, 2021, arXiv preprint arXiv:2104.07295.
[38] Y. Yang, Z. Guan, J. Li, W. Zhao, J. Cui, Q. Wang, Interpretable and efficient heterogeneous graph convolutional network, IEEE Trans. Knowl. Data Eng. (2021).
[39] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, B. Zhou, End-to-end structure-aware convolutional networks for knowledge base completion, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI, Honolulu, Hawaii, USA, January 27 - February 1, 2019, 2019, pp. 3060–3067.
[40] L. Van der Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (11) (2008).