



Open-world knowledge graph completion for unseen entities and relations via attentive feature aggregation

Byungkook Oh, Seungmin Seo, Jimin Hwang, Dongho Lee, Kyong-Ho Lee*

Department of Computer Science, Yonsei University, Seoul, South Korea



ARTICLE INFO

Article history:

Received 1 September 2020

Received in revised form 27 November 2021

Accepted 28 November 2021

Available online 3 December 2021

Keywords:

Knowledge graph embedding

Link prediction

Graph-based learning

Graph convolutional networks

Deep learning

Attention model

ABSTRACT

Most of knowledge graph completion (KGC) models are designed for static KGs where entity and relation sets are fixed. These approaches are inherently *transductive* because they simply predict the plausibility of facts whose entities and relations had to previously appear in the training phase. However, since entities and relations are constantly added, removed, or changed over time, KGC models should be able to generalize to out-of-KG entities and relations in evolving KGs, which are more suited to real-world scenarios. Moreover, incorporating global graph-structured information into KGC models is another challenging issue. To overcome these issues, this paper proposes a novel **Inductive KG Embedding (IKGE)** model for open-world KGC, which accommodates out-of-KG entities and relations. Unlike training individual unique embeddings, the proposed model fundamentally learns an embedding generator function to elaborately generate fact embeddings in an *inductive* manner. Specifically, the feature information of each fact is extracted as a vector from its entity-related and relation-related side information via an attention mechanism. Then, to score a given fact, our neighborhood feature aggregator hierarchically accumulates the feature information of multi-hop neighbors. Experimental results show that IKGE outperforms existing approaches in both transductive and inductive setups by successfully aggregating neighborhood features.

© 2021 Elsevier Inc. All rights reserved.

1. Introduction

Knowledge graphs (KGs) are efficient data models, which structuralize real-world facts about entities and organically link them for machine understandability. Each fact in KGs is typically expressed as a directional relationship $\langle h, r, t \rangle$ (edge), where head entity h and tail entity t (nodes) are connected to each other via a semantic relation r (edge type). Recently, KGs have been spotlighted since they facilitate the semantic integration and interoperation of heterogeneous data. Prominent large-scale KGs, such as DBpedia [1], YAGO [2], NELL [3], WikiData [4], Freebase [5], WordNet [6], and Google Knowledge Graph,¹ have been constructed for various practical applications, including question answering [7], recommender systems [8,9], information extraction [10,11], and entity linking [12,13]. However, the symbolic nature and arbitrary structure of such facts make it hard to elaborately manipulate KGs in many knowledge inference tasks.

In recent years, knowledge graph embedding (KGE) has become a successful and promising approach, especially in the knowledge graph completion (KGC) tasks including link prediction and triple classification. Given two elements in a fact

* Corresponding author.

E-mail addresses: smseo@icl.yonsei.ac.kr (S. Seo), jmhwang@icl.yonsei.ac.kr (J. Hwang), dhlee@icl.yonsei.ac.kr (D. Lee), khlee89@yonsei.ac.kr (K.-H. Lee).

¹ <https://developers.google.com/knowledge-graph>.

$\langle h, r, t \rangle$ with a scoring function $f_r(h, t)$, link prediction decides the remaining element that yields the highest plausibility score of the fact. Triple classification checks whether an unseen fact, not contained in training data, is valid or not based on a relation-specific threshold learned in advance. The key idea behind embedding-based KGC is to automatically embed structural information into latent task-specific low-dimensional vectors (or matrices) through maximizing the total plausibility score of existing facts in KGs. Based on the score of a given fact, KGC tasks refine KGs often suffering from incompleteness and noise even in manually curated KGs.

Despite the successes of the KGC models, they have a serious limitation: KGE for KGC mostly learns only from static KGs where entity and relation sets are fixed. This is often called the closed-world assumption [14]. They score a fact $\langle h, r, t \rangle$ in a test phase, where entities h and t , and relation r had to previously appear while training. In other words, they do not target evolving KGs more suited to real-world scenarios, where entities and relations are constantly added, removed, or changed over time (i.e., open-world assumption) and maintain connectivity patterns and semantic meanings. In practice, NELL has continuously collected the facts extracted from the Web since 2010. DBpedia added 200 new entities every day for 6 months in 2016 in addition to a lot of the facts changed and deleted [14]. In the existing literature [14,15], the emerging entities and relations are referred to as out-of-KG entities and relations. KGC dealing with them is also known as open-world (or zero-shot) KGC.

There are several crucial issues to support the out-of-KG entities and relations for open-world KGC:

- **[C1] Generating Embedding:** Open-world KGC should be able to decide the plausibility of test facts even though entities and relations were not seen during training. This is hard for the conventional KGC models because they are inherently *transductive* (i.e., under closed-world assumption), where entities and relations are assigned unique vector representations to be learned. It is challenging to generate new embeddings for both unseen entities and relations on the fly (aka inductive learning), instead of simply retraining per the snapshot of an evolving KG.
- **[C2] Preserving Structure:** Since KGE typically exploits the structural similarity of connectivity (statistical) patterns in a KG [16], local neighborhood structure for each fact helps in improving the predictive power of its embedding. Due to the arbitrary structure of KGs, most models only consider local structures such as 1-hop neighbors or relation paths. It is challenging to incorporate global graph-structured information (e.g., multi-hop neighborhoods) into the fact embeddings.

However, there are only a few KGC models partially addressing these challenges. The compositional vector space model for open-world KGC [17] computes the vector representation of arbitrary-length relation paths and determines pre-learned unseen relations corresponding to the path vectors. Recently, ConMask [14] and DKRL [15] exploit entity descriptions to generate entity embeddings. Since the above models only handle either out-of-KG entities or relations for open-world KGC, they satisfy **C1** partially. LiLi [18] continuously and interactively learns new knowledge via a dialogue system in a lifelong learning manner, which retains new facts acquired during the conversation and leverages them in future learning. Thus, they handle both out-of-KG entities and relations in evolving KGs. However, they are less challenging than a holistic manner because they need interactions and sequential independent subtasks such as relation resolution, entity linking, etc. Moreover, all of the above models do not fulfill **C2**. Note that, although several state-of-the-art graph embedding (GE) models satisfy **C2**, they are less challenging than KGE. The reason is that they consider only nodes (i.e., entities) but not relations and do not deal with the open-world KGC task.

To tackle the above challenges, this paper proposes a novel **Inductive KGE (IKGE)** model for open-world KGC, which enables to efficiently generalize to out-of-KG entities and relations without model retraining while preserving graph structure. Unlike most existing *transductive* KGC models, the proposed model is *inductive* by fundamentally learning an embedding generator function to compute the vector representations of the facts containing out-of-KG entities and relations. Multi-hop neighborhoods are also considered to reflect graph-structured information in the embedding generation process.

Specifically, entities in a given fact are encoded into entity features from their description, where the description words are shared across all facts. Since the same description word may have different impacts on identifying facts, an attention mechanism is additionally employed in the entity encoding to capture more relevant description words for the fact. Then, the head and tail entity feature vectors are combined to extract fact features and utilized as an initial fact embedding. As a result, for each fact in a KG, the proposed approach inductively generates the fact embedding from its side information. The words in side information are shared across all facts to generalize facts whether or not they include out-of-KG entities and/or relations. As shown in Fig. 1, the State/Massachusetts-specific entity features for Harvard_University and the State/Harvard_University-specific entity features for Massachusetts are extracted from the word-level shared side information and then combined to create fact features as an initial fact embedding.

Then, IKGE collects multi-hop neighborhood features through the in-KG entities. Inspired by [19], our feature aggregator hierarchically accumulates fact feature vectors from the multi-hop neighbors to the target fact. By utilizing the aggregated neighborhood vector and the target fact's initial embedding, IKGE gets the inductively generated vector of the target fact and incorporates global graph-structured information into the target fact vector. This allows us to address the challenges **C1** and **C2** simultaneously. Moreover, with the above type constraints, the feature information can be reduced and aggregated by type matching in an end-to-end manner.

Thus, facts and their relationships are identified based on the semantic similarity between word-level shared side information. Since the side information of a relation is much less, it is difficult to specify each relation. Instead, the proposed

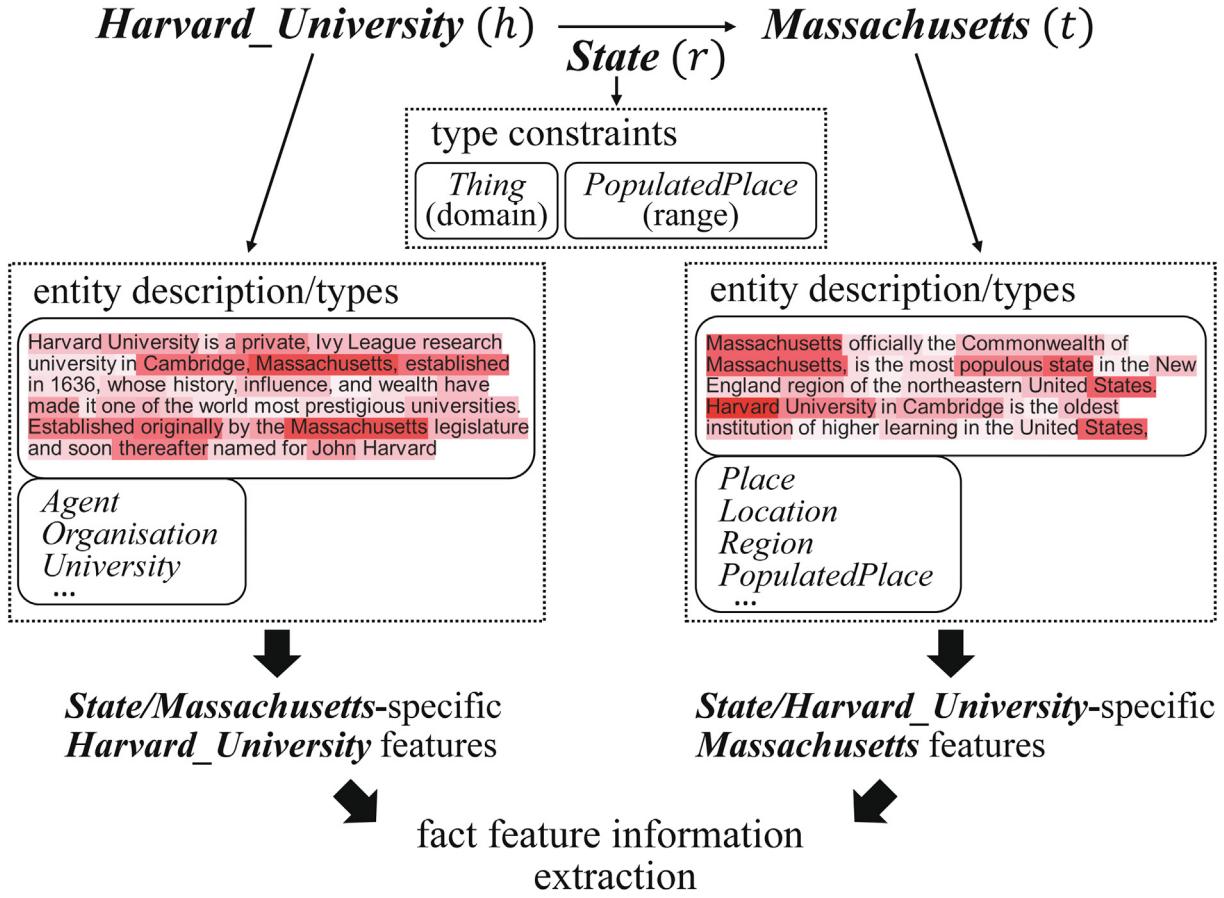


Fig. 1. Example of a fact feature extraction process. Entity features extracted from relation-related (name and type constraints) and entity-related (name, description, and types) side information are combined to inductively generate fact features as initial fact embedding.

model focuses on determining the plausibility of a target fact (i.e., open-world KGC) based on a scoring function. IKGE conducts link prediction and triple classification in the open-world scenario. Note that IKGE handles out-of-KG entities and relations even without considering its relation embedding. Experimental results show that IKGE outperforms all existing approaches in both transductive and inductive setups by successfully aggregating neighborhood features.

2. Related work

The main goal of GE/KGE is to learn embeddings of graphs while preserving graph structure and various side information. Most GE/KGE models can be described as an encoder-decoder structure. Here, an encoder maps a symbolic graph component to a learnable embedding. Given the embeddings, a decoder then optimizes them to decode task-specific information to be preserved, e.g., node-pair similarity for clustering or node-edge-node validity for link prediction. Readers may refer to comprehensive recent surveys on GE [20] and KGE [16,21]. Encoders can be roughly classified into simple and complex encoding approaches.

The complex encoding approaches in GE and KGE are related to open-world learning. Open-world learning predicts unseen classes which do not belong to training data, whereas supervised learning assumes that test data always belong to known classes [22]. GE/KGE models in open-world scenarios focus on the out-of-graph nodes or edges that have no pre-assigned representations to be learned. In order to predict them, the vector representations generated from their side information are required.

Simple Encoding Approach. Simple encoders directly optimize a unique embedding assigned to each node. Then, traditional GE models employ a pairwise decoder to approximate the similarity (e.g., inner product) of two embeddings to node-pair similarity in graphs (e.g., adjacency matrix). A lot of studies embed graphs via a pairwise decoder while preserving the local proximities of nodes in random walks [23–25] or additionally preserving the first-order proximity between connected two nodes [25]. Intuitively, KGE is more challenging than GE due to the nature of labeled directed edges. Thus, KGC models extend pairwise decoders to triple-wise decoders with relation-specific parameters to score facts.

Traditional KGC models, including translational and compositional models, embed entities and relations based on the structural (topological) similarity obtained from connectivity patterns in KGs. The topology-based models leverage only structural information without side information, e.g., a single fact alone [26–28] and multiple facts such as neighborhoods [29] and relation paths [17].

However, since real-world KGs hold a tremendous number of entities and relatively very sparse links, it is not easy to characterize an individual entity or relation via the topology-based embeddings. To cope with the problem, the main focus of recent KGC models lies in better characterizing entities and relations than the topology-based models by exploiting a variety of side information associated with a given fact, such as entity types, textual descriptions, images, non-discrete attributes, and even logical rules. With the beneficial features, there has been a surge of advanced KGC models that incorporate some side information into their scoring functions. Therefore, compared to the topology-based models, advanced KGC models can learn predictive embeddings, which better characterize entities and relations individually so as to further improve the KGC tasks.

The compositional vector space model supports an open-world KGC [17] by computing the vector representations of arbitrary-length relation paths between two entities. Then, an unseen relation between the entities can be determined by comparing the path vector with pre-trained relation vectors. However, they only aim to predict an unseen relation (not entity) from a fixed relation set with pre-trained embeddings. Recently, ConMask [14] and DKRL [15] exploit entity descriptions to generate entity embeddings. However, unlike our IKGE model, the models cannot handle both out-of-KG entities and relations at the same time for open-world KGC.

Complex Encoding Approach. Complex encoders learn functions to elaborately generate node embeddings compared to the simple encoders that directly optimize a unique embedding assigned to each node. Thus, most complex encoders are usually combined with any decoder of simple encoding models (e.g., node2vec [24] and DeepWalk [23]) for specific downstream tasks.

Graph convolutional networks (GCNs) adapt a convolution operation to arbitrarily structured graphs. Given a node, GCN-based complex encoders exploit local neighborhood structure and various features (e.g., image, text) of neighbors to obtain its embedding. Patchy-san [30] normalizes a local neighborhood of each node to uniquely transform the neighbors into a node sequence via a pre-defined node ordering. The node sequences are utilized as receptive fields for convolutions. GCNs based on spectral convolutions are first proposed by [31], where the key idea is to compute a node embedding by iteratively aggregating its local neighborhood features fully [32] or partially [19]. Inspired by the works, R-GCNs [33] extends GCNs for KGC by employing a scoring function (i.e., decoder) of existing KGC models such as DistMult [27] and ComplEx [28]. Despite the success of GCNs, most models are developed for a transductive inference over fixed graphs because a node embedding is generated with the full graph Laplacian and the features of a node and its neighbors.

Recently, inductive GE models have received significant attention which can generalize to evolving nodes. GraphSAGE [19], an extended GCN encoder, enables to generate a node embedding of a new node by aggregating features of sampled local neighbors. However, the new node must have connections to existing nodes for integrating its neighbors' features.

3. Preliminaries

First of all, we introduce some notations used in the rest of this paper. Boldface upper-case (e.g., \mathbf{F}) and lower-case (e.g., \mathbf{f}) letters represent matrices/tensors and vectors, respectively. Sets are expressed in non-bold upper-case type (e.g., \mathcal{F}), whereas variables or scalar values are represented in non-bold lower-case type (e.g., f).

A typical knowledge graph $\mathcal{G} = \{\langle h, r, t \rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ can be intrinsically represented by a labeled directed graph, where \mathcal{E} and \mathcal{R} indicate the fixed sets of entities and relations, respectively. $\langle h, r, t \rangle$ indicates a directional triple-wise relationship in the form of (head entity, relation, tail entity) called a fact in \mathcal{G} . In this paper, we focus on schema-based KGs. entities in KGs have additional information such as types and text descriptions. Also, each relation has name and relation-specific type information which restricts the types of entities in head and tail positions.

Most popular KGs, including Wikidata, DBpedia, and Freebase, are constructed based on the W3C Resource Description Framework (RDF)² for the Semantic Web. The data-representation framework allows expressing entity types (`rdf:type`). In addition, RDF Schema (RDFS)³ provides a data-modelling vocabulary such as hierarchical entity types (`rdfs:subClassOf`), and relation-specific type constraints on head entities (`rdfs:domain`) and tail entities (`rdfs:range`). Since each entity plays different roles in different scenarios, entities in a schema-based KGs intuitively belong to multiple semantic categories through hierarchical entity types.

Fig. 1 illustrates an example fact `<Harvard_University, State, Massachusetts>`. Harvard university has multiple entity types such as {University, EducationalInstitution, Agent, Organisation}. Likewise, massachusetts has entity types {Place, AdministrativeRegion, PopulatedPlace, Location, Region}. Thus, facts containing the relation `State` are only valid when the type of a tail entity is `PopulatedPlace` due to the `rdfs:range` constraints. Note that the type `Thing` of `rdfs:domain` indicates that there are no type constraints on head entities.

² <https://www.w3.org/TR/rdf11-concepts/>.

³ <https://www.w3.org/TR/rdf-schema/>.

4. Problem formulation

In this section, We first formally define a typical closed-world KGC problem for a fixed set of in-KG entities and relations. Each of the entities and relations is assigned a unique embedding vector which is learnable. The closed-world KGC problem is as follows:

Definition 1. Given an incomplete knowledge graph \mathcal{G} , **closed-world knowledge Graph Completion** aims to score test facts $F = \{\langle h, r, t \rangle | \langle h, r, t \rangle \notin \mathcal{G}, h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E} \}$ to predict their plausibility, where \mathcal{E} and \mathcal{R} denote entity and relation fixed sets which contain in-KG entities and relations, respectively.

An open-world KGC problem is extended from the conventional closed-world KGC problem, where out-of-KG entities are represented with their textual descriptions. Entity embeddings are generated using pre-trained word embeddings. Especially, the open-world KGC in our approach targets not only out-of-KG entities but also out-of-KG relations. The key idea behind the end-to-end inductive KGE (IKGE) model is how to generate the low-dimensional vector embedding of a target fact from word-level shared side information, which contains out-of-KG entity and relation as follows:

Definition 2. Given an incomplete knowledge graph \mathcal{G} , **Open-World Knowledge Graph Completion** aims to score test facts $F = \{\langle h, r, t \rangle | \langle h, r, t \rangle \notin \mathcal{G}, (h \in \mathcal{E}, r \in \mathcal{R}^*, t \in \mathcal{E}^*) \cup (h \in \mathcal{E}^*, r \in \mathcal{R}^*, t \in \mathcal{E}) \cup (h \in \mathcal{E}^*, r \in \mathcal{R}, t \in \mathcal{E}^*)\}$ to predict their plausibility, where \mathcal{E}^* and \mathcal{R}^* denote entity and relation supersets which additionally contain out-of-KG entities and relations, respectively.

Moreover, we incorporate graph-structured information into fact embeddings via aggregating fact feature information from multi-hop neighbors.

5. The IKGE model

As shown in Fig. 2, in the training phase, (a) given a sample KG, (b) we first extract fact feature information for every fact from word-level side information and construct a line graph where a node and an edge are a fact and a pair of adjacent edges, respectively. (c) After applying an attention-based GCN, a fact feature extractor for fact feature information extraction, agg-

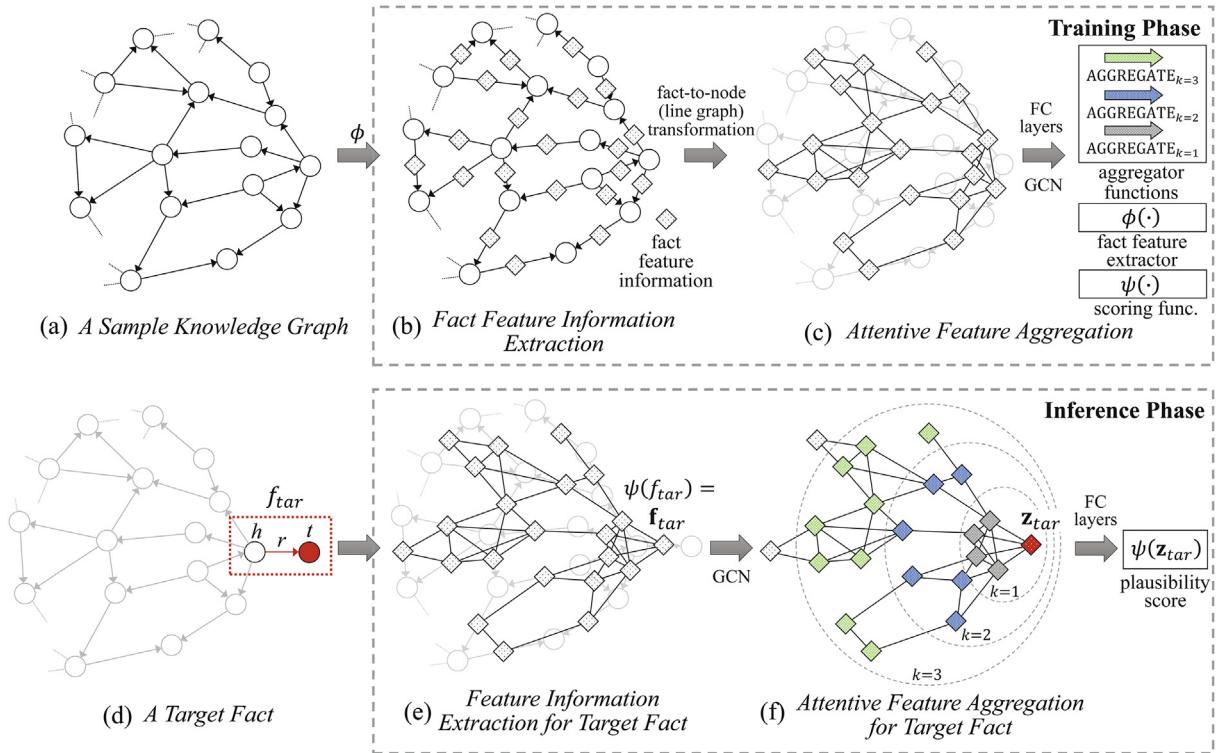


Fig. 2. The overall framework of the proposed IKGE model. In the training phase, a fact feature extractor for fact feature information extraction, aggregator functions for attentive feature aggregation, and FC layers for scoring facts, are trained. In the inference phase, the target fact's embedding \mathbf{z}_{tar} is inductively generated by hierarchically accumulating extracted feature information \mathbf{f}_{tar} of a target fact f_{tar} and multi-hop neighboring fact feature information until depth $k = 3$. Finally, we can score the target fact with FC layers to decide its plausibility score.

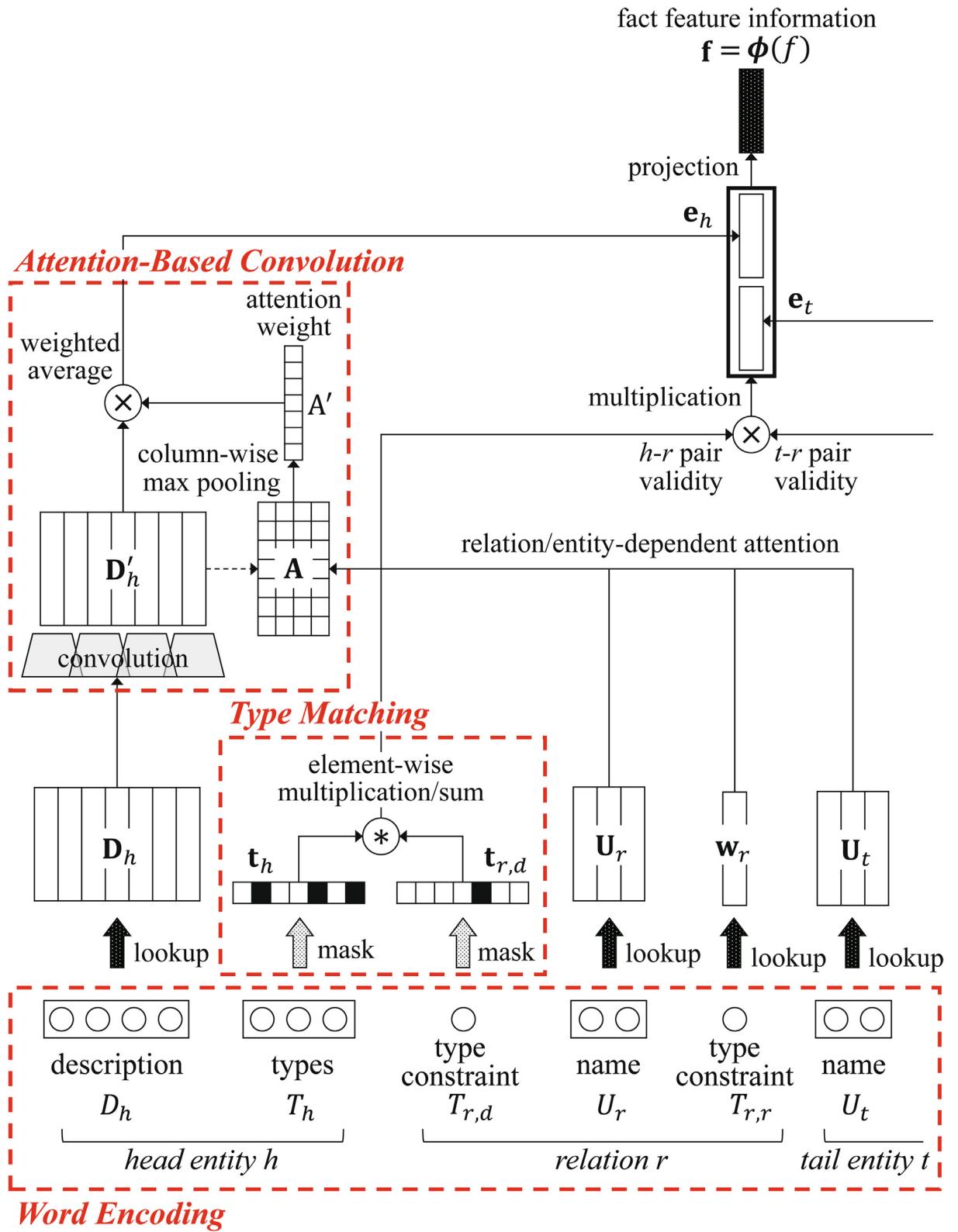


Fig. 3. Illustration of fact feature information extraction. For a fact f , the feature vectors \mathbf{e}_h and \mathbf{e}_t for head and tail entities are extracted from their textual descriptions with attention on relation name, type constraints, and the other entity name. Then, the fact feature information $\mathbf{f} = \phi(f)$ is generated by concatenating \mathbf{e}_h and \mathbf{e}_t , and by type matching.

gator functions for attentive feature aggregation, and fully-connected (FC) layers for scoring facts, are trained via supervised learning. In the inference phase, (d) given a target fact f_{tar} where r and t are out-of-KG, (e) we extract the feature information \mathbf{f}_{tar} of the target fact with the trained fact feature extractor. Then, (f) the multi-hop neighboring fact feature information is hierarchically accumulated with the trained aggregator functions. Finally, we can score the generated target fact's embedding \mathbf{z}_{tar} with the FC layers to determine the plausibility of the target fact f_{tar} . The proposed model IKGE for open-world KGC includes two major processes where we assume that its parameters have already been trained as follows:

- **Fact Feature Information Extraction.** For relation r and entity e , we denote relation-related (type constraints $T_{r,d}$ for domain and $T_{r,r}$ for range, and relation name U_r) and entity-related (description D_e , types T_e , and name U_e) word-level shared side information. For the h - r pair in a fact $\langle h, r, t \rangle$, words in D_h are attended by U_r , U_t , and $T_{r,r}$ with two-way attention mechanism. This process is the same for the t - r pair. In addition, we check the validities of the h - r and t - r pairs via type matching with entity types and type constraints. In summary, given a fact f with the side information, we extract fact feature information \mathbf{f} by a fact feature extractor $\phi(\cdot)$ consisting of attention-based sentence modeling and type matching (see Fig. 3).
- **Attentive Feature Aggregation.** After a fact-to-node (line graph) transformation, we apply an attention-based graph convolution network to recursively aggregate neighboring facts' feature information. For all fact f in a KG, we refer to a set of neighboring facts immediately connected with f as the neighborhood \mathcal{N}_f . We accumulate the facts' feature information $\mathbf{f}_i = \phi(f_i), \forall f_i \in \mathcal{N}_f$, into a single vector $\mathbf{h}_{\mathcal{N}_f}$ by an attentive feature aggregator function which is formulated based on attention mechanism. Then, the aggregated neighborhood vector $\mathbf{h}_{\mathcal{N}_f}$ is combined with the fact feature information \mathbf{f} of the fact f . The combined vector $\tilde{\mathbf{f}}$ is used as fact feature information for f at the next aggregator function. As shown in Fig. 2(c), the attentive feature aggregation process is iteratively performed from search depth 1 to k . At final depth k , a score function $\psi(\cdot)$ based on fully-connected layers assesses the fact f .

5.1. Fact feature information extraction

Given a fact consisting of head/tail entities and a relation, we rely on entity descriptions as side information to extract relation-specific head and tail entities' features. Then, we generate fact feature information from the entity features. To enhance the usefulness of the features, we further exploit the other entity name as well as relation-related information for attention. Especially, relation-specific type constraints are used to reduce the plausibility scope for test facts. As shown in Fig. 3, there are three main modules: word encoding, attention-based convolution, and type matching.

Most KGs are schema-based knowledge bases, where each entity (i.e., node) is represented by a globally unique identifier and appears only once. Similarly, relations also have unique identifiers, but they are much fewer than the entities and each of them appears multiple times as an edge in a KG. Since each relation is correlated with a very large number of head and tail entities, modeling relations is much more difficult than entities. For the reason, recent open-world KGC models focus on generalizing to out-of-KG entities, not out-of-KG relations. They entirely rely on entity descriptions to specify entities and generate entity embeddings from pre-trained word embeddings.

On the other hand, we not only utilize entity features extracted from descriptions but also obtain more informative features by attending to side information, i.e., relation name, relation-specific type constraints, and the other entity's name in a fact. Moreover, after the fact feature information extraction process, we aggregate the feature information of multi-hop neighboring facts to generate the final fact embedding of a target fact in an inductive manner. The attentive feature aggregation process can involve neighborhood graph-structured information in the final fact embedding.

5.1.1. Word encoding

As shown in Fig. 3, we utilize relation names, type constraints (i.e., `rdf:domain` and `rdf:range`), and entity names and types to model the correlation among a relation r and entities h and t . The relation and entity names are encoded on the level of words which are shared by the same vocabulary. Since the raw relation and entity names are a set of words and rarely exist in the vocabulary, we parse them into word sets which are shared with entity description words. Thus, the correlation between a relation and two entities in a fact can be captured from their names, types, and descriptions. The word-level modeling from shared information allows handling out-of-KG relation and entity by extracting fact feature information from the word-level shared side information. To do so, the relation and entity names are first split into words. Then, we perform lemmatization to extract the basic forms of the words, which exist in a dictionary.

Given a fact $\langle h, r, t \rangle$, each of the entities h and t as an entity e has a description $D_e = \{w_1, w_2, \dots, w_n\}$, types $T_e = \{w_1, w_2, \dots, w_m\}$, and a name $U_e = \{w_1, w_2, \dots, w_p\}$, which indicate the sets of symbolic word representations. Likewise, the relation r has name $U_r = \{w_1, w_2, \dots, w_k\}$, domain type constraint $T_{r,d} = \{w_d\}$, and range type constraint $T_{r,r} = \{w_r\}$. All the above words are represented by symbolic representations and shared through a same vocabulary $w_i \in \mathcal{W}$. We first encode each word w_i into d -dimensional vector representation \mathbf{w}_i in an embedding matrix $\mathbf{W}_{voca} \in \mathbb{R}^{d \times |\mathcal{W}|}$ for words, where $|\mathcal{W}|$ is the size of the word vocabulary. The word embedding matrix \mathbf{W}_{voca} can be initialized with pre-trained word vectors, such as GloVe,⁴ word2vec,⁵ and fastText⁶ vectors.

5.1.2. Attention-based convolution

Recently, attention mechanisms [34,35] are commonly used in sentence modeling for neural machine translation [36], where an encoder first maps word vectors of an input sentence into a fixed-length sentence vector via recurrent neural networks (RNNs). Then, another RNN units decode the sentence vector into a word sequence. As other important attention models, two-way attention models [37,38] are widely studied for pair-wise ranking or classification such as question answering and interaction modeling. Inspired by the interaction modeling of the two-way attention, we mask entity description D_e with relation name U_r , relation-specific type constraint T_r , and the other entity name U_e by capturing their semantic textual similarity. In general, RNNs such as long short-term memory networks (LSTMs) [39] are employed for a word-to-word modeling to encode the *whole context* within the sentence, whereas convolutional neural networks (CNNs) focus on the *local context* to model the sentence [40]. Since we need to capture the local context features within entity descriptions rather than the whole context features, we employ the attention strategy with a CNN. Thus, the convolution operations generate more abstract features of higher granularity from the description.

As shown in Fig. 3, the attention-based convolution module models the entity description D_h for the head entity h . Firstly, the words in D_h are encoded into the entity description matrix $\mathbf{D}_h \in \mathbb{R}^{d \times n}$ via embedding lookup. In the same way, the representations $\mathbf{w}_r \in \mathbb{R}^{d \times 1}$ and $\mathbf{U}_r \in \mathbb{R}^{d \times k}$ are respectively generated for the range type constraint $T_{r,r}$ and relation name U_r . The tail entity's name U_t is also encoded into $\mathbf{U}_t \in \mathbb{R}^{d \times p}$. Then, we set $\mathbf{D}_{l,h}$ to be the output of two 1D convolutions over the description matrix \mathbf{D}_h . Following [37], attention weights are computed on the convolution output $\mathbf{D}_{l,h}$ according to the range type constraint $T_{r,r}$, relation name U_r , and tail entity's name U_t . To obtain the attention weights, we compute an attention score matrix \mathbf{A} which compares all columns of $\mathbf{D}_{l,h}$ with all columns of \mathbf{w}_r , \mathbf{U}_r , and \mathbf{U}_t as follows:

$$\mathbf{A} = \tanh((\mathbf{D}_{l,h})^T \mathbf{W}_a \text{cat}(\mathbf{w}_r, \mathbf{U}_r, \mathbf{U}_t)), \quad (1)$$

where $\mathbf{A} \in \mathbb{R}^{n \times (1+k+p)}$ and $\mathbf{W}_a \in \mathbb{R}^{d \times d}$ denote the attention score matrix and a weight matrix, respectively. $\text{cat}(\cdot, \cdot, \cdot)$ indicates a function for horizontally concatenating three inputs so that \mathbf{w}_r , \mathbf{U}_r , and \mathbf{U}_t are concatenated. Then, we apply a column-wise max pooling over \mathbf{A} as follows:

$$\mathbf{A}_{l,j} = \max_{1 \leq i \leq 1+k+p} \mathbf{A}_{ij} \quad (2)$$

Finally, the entity feature vector $\mathbf{e}_h \in \mathbb{R}^d$ is computed by a matrix multiplication (i.e., weighted average) between $\mathbf{D}_{l,h}$ and $\text{softmax}(\mathbf{A})$ as follows:

$$\mathbf{e}_h = \mathbf{D}_{l,h} \text{softmax}(\mathbf{A}). \quad (3)$$

The attention-based convolution process is performed once more for the tail entity's description to produce the entity feature vector \mathbf{e}_t . The two resulting feature vectors $\mathbf{e}_h \in \mathbb{R}^d$ and $\mathbf{e}_t \in \mathbb{R}^d$ are concatenated and projected onto a vector $\mathbf{f} \in \mathbb{R}^d$ to produce fact feature information of f as follows:

$$\mathbf{f} = \mathbf{W}_p \begin{bmatrix} \mathbf{e}_h \\ \mathbf{e}_t \end{bmatrix} + \mathbf{b}_p, \quad (4)$$

where $\mathbf{W}_p \in \mathbb{R}^{d \times 2d}$ and $\mathbf{b}_p \in \mathbb{R}^d$ are weight matrices.

Note that the weight matrices \mathbf{W}_{voca} , \mathbf{W}_{c1} , \mathbf{W}_{c2} , \mathbf{W}_a , \mathbf{W}_p , \mathbf{b}_p are shared in generating both \mathbf{e}_h and \mathbf{e}_t , where \mathbf{W}_{c1} and \mathbf{W}_{c2} are the kernels of the two convolutions.

5.1.3. Type matching

In addition, we also consider the entity and relation type information to determine the fact validity before the attentive feature aggregation. Fig. 3 illustrates the type matching for the h - r pair. Let entity types T_h for the entity h and domain type constraint $T_{r,d} = \{w_d\}$ be given. The entity types T_h have to contain the type constraint w_d .

To allow our model to be fully differentiable, we design the type matching process based on a matrix multiplication and a summation. Thus, model parameters can be trained in end-to-end learning with gradient descent. To exploit the type information, we first mask the domain type constraint $T_{r,d}$ and range type constraint $T_{r,r}$ as one-hot type vectors $\mathbf{t}_{r,d}, \mathbf{t}_{r,r} \in \mathbb{R}^{|\mathcal{W}|}$, respectively, where $|\mathcal{W}|$ is the total number of types. Likewise, the head entity types T_h and tail entity types T_t are masked as multi-hot type vectors $\mathbf{t}_h, \mathbf{t}_t \in \mathbb{R}^{|\mathcal{W}|}$, respectively. Then, we perform element-wise multiplications of the type vectors and a summation of all elements. Since the type matching for the h - r and t - r pairs has to be satisfied at the same time, we multiply the result values for the h - r and t - r pairs as follows:

$$\mathbf{f} \leftarrow \mathbf{f} \times \left(\sum_i^{|\mathcal{W}|} (\mathbf{t}_h * \mathbf{t}_{r,d})_i \times \sum_i^{|\mathcal{W}|} (\mathbf{t}_t * \mathbf{t}_{r,r})_i \right), \quad (5)$$

⁴ <https://nlp.stanford.edu/projects/glove/>.

⁵ <https://code.google.com/archive/p/word2vec/>.

⁶ <https://fasttext.cc/docs/en/support.html>.

where $*$ denotes the element-wise multiplication. $\mathbf{t}_{r,d}$ and $\mathbf{t}_{r,r}$ denote the masked one-hot type vectors for domain and range type constraints, respectively. Also, \mathbf{t}_h and \mathbf{t}_t denote the masked multi-hot type vectors for head and tail entities, respectively. If the type constraint is satisfied, the result values for the h - r pair and t - r pair validities will be 1, otherwise 0. Thus, the fact feature information \mathbf{f} becomes a zero vector and disappears if the type constraint is not satisfied.

5.2. Attentive feature aggregation

In order to evaluate the validity of a target fact $f_{tar} = \langle h, r, t \rangle$ which contains out-of-KG entity and relation, we need to generate a final fact embedding \mathbf{z}_{tar} of the fact f_{tar} , while containing key structural information of its multi-hop neighbors. In this section, the attentive feature aggregation process aims to compute \mathbf{z}_{tar} by hierarchically aggregating fact vectors within multi-hop neighbors of f_{tar} . Inspired by the idea of GraphSAGE [19], we employ the hierarchical aggregator functions to aggregate the multi-hop neighborhood feature information obtained from Section 5.1.

Moreover, we leverage attention mechanism [41–44] on graph-structured data (i.e., knowledge graphs) to focus on important neighbors' fact vectors. After the attentive feature aggregation, we can generate the final fact embedding \mathbf{z}_{tar} and evaluate it based on a simple scoring function to decide its plausibility. This process ensures that \mathbf{z}_{tar} can preserve the graph-structured information from the multi-hop neighborhoods so that we address the aforementioned challenge C2.

We set the search depth from the target fact as $k \in \{1, \dots, K\}$, where K denotes the maximum depth for aggregating features of the target fact's k -hop neighborhoods. For each search depth k , we build an attentive feature aggregator function denoted by AGGREGATE_k , which accumulates exactly 1-hop neighbors' fact features and then passes the aggregated neighborhood features to the next aggregator function AGGREGATE_{k+1} at depth $k + 1$. Thus, as the search depth is deeper, the target fact gains more and more neighborhood features while containing graph structural information.

5.2.1. Aggregator function

To generate a final fact embedding of a target fact, IKGE hierarchically accumulates fact feature information of multi-hop neighbors along with the aggregation paths. The fact vectors are initialized by the initial fact feature extraction process. Each aggregation path connects from leaf entity to root entity. All aggregation paths constitute a form of a tree as shown in Fig. 2 (c). For a k -hop fact f_u at depth k , the aggregator function AGGREGATE_{k+1} takes neighboring fact vectors $\mathbf{N}(f_u) = \{\mathbf{f}_v, \forall f_v \in \mathcal{N}(f_u)\}$ as follows:

$$\mathbf{h}_{\mathcal{N}(f_u)}^{k+1} = \text{AGGREGATE}_{k+1}(\mathbf{N}(f_u)), \quad (6)$$

where $\mathcal{N}(f_u)$ denotes the variable-sized neighboring fact set of f_u . f_v denotes one neighbor of f_u . $\mathbf{h}_{\mathcal{N}(f_u)}^{k+1}$ is the aggregated neighborhood vector.

In the attentive feature aggregator function, we entirely rely on attention mechanism to focus on more important fact vectors in combining the neighborhood $\mathcal{N}(f_u)$ of f_u . In general, the important information of entities can be obtained via the Weisfeiler-Lehman algorithm [45] or centrality measures such as PageRank, degree centrality, and betweenness centrality, but we do not need the relative importance in aggregating graph-structured information from neighborhoods. Instead of the order information, we utilize content-based attention mechanism in the attentive feature aggregator function to obtain the learnable importance of fact vectors. Thus, we do not need to sample the neighboring facts $\mathcal{N}(f_u)$ to select relatively more important ones or prune relatively unnecessary ones. This makes the attentive feature aggregation of IKGE combine the variable-sized neighbors into the single fixed-length vector $\mathbf{h}_{\mathcal{N}(f_u)}^{k+1}$.

In our attention strategy, fact vectors $\mathbf{N}(f_u)$ and their parent fact vector \mathbf{f}_u are utilized to compute an attention weight a_v which indicates the relevance between each fact in $N(f_u)$ and f_u as follows:

$$\begin{aligned} a_v &= \text{softmax}_v(\text{AT_SCORE}(\mathbf{N}(f_u), \mathbf{f}_u)) \\ &= \frac{\exp(\text{att_score}_v(\mathbf{N}(f_u), \mathbf{f}_u))}{\sum_{i \in \mathcal{N}(f_u)} \exp(\text{att_score}_i(\mathbf{N}(f_u), \mathbf{f}_u))}, \end{aligned} \quad (7)$$

where the attention weight a_v denotes the importance of the v th neighboring fact in combining the neighborhood $\mathcal{N}(f_u)$ while considering the parent fact f_u . Attention weights are obtained from the attention scores for $\mathcal{N}(f_u)$ via the softmax function which ensures that the sum of $\{a_v, \forall f_v \in \mathcal{N}(f_u)\}$ is 1. Here, we can obtain each attention score for a fact vector $\mathbf{f}_v, \forall f_v \in \mathcal{N}(f_u)$, conditioned on its parent fact \mathbf{f}_u as follows:

$$\text{AT_SCORE}_v(\mathbf{f}_v, \mathbf{f}_u) = \mathbf{f}_v \mathbf{W}_a^{k+1} \mathbf{f}_u, \quad (8)$$

where \mathbf{W}_a^{k+1} denotes the parameter of the attentive feature aggregator function at depth $k + 1$. There are K sets of the parameter depending on the maximum search depth K . Each parameter set is shared for the aggregator function at the same depth. As a result, the fact vectors $\{\mathbf{f}_v, \forall f_v \in \mathcal{N}(f_u)\}$ are combined with the attention weight vector by a weighted sum as follows:

$$\mathbf{h}_{\mathcal{N}(f_u)}^{k+1} = \text{AGGREGATE}_{k+1}(\mathbf{N}(f_u)) = \tanh \left(\sum_{f_v \in \mathcal{N}(f_u)} a_v \mathbf{f}_v \right), \quad (9)$$

where $\mathbf{h}_{\mathcal{N}(f_u)}^{k+1}$ denotes the aggregated neighborhood vector, which indicates contextual information of the parent fact f_u . Then, the aggregated neighborhood vector $\mathbf{h}_{\mathcal{N}(f_u)}^{k+1}$ is simply combined with the initial fact embedding \mathbf{f}_u as follows:

$$\tilde{\mathbf{f}}_u = \mathbf{h}_{\mathcal{N}(f_u)}^{k+1} + \mathbf{f}_u, \quad (10)$$

where $\tilde{\mathbf{f}}_u$ denotes a context-aware fact vector of f_u . Along with a aggregation path, $\tilde{\mathbf{f}}_u$ is injected into the next aggregator function AGGREGATE_k at search depth k as follows:

$$\mathbf{f}_u \leftarrow \tilde{\mathbf{f}}_u. \quad (11)$$

Note that the context-aware fact vector $\tilde{\mathbf{f}}_u$ is considered as the fact vector of f_u to be aggregated with other fact vectors at the aggregator function AGGREGATE_k . This process is performed repeatedly for the search depth $k \in \{1, \dots, K\}$, where K is the maximum depth, until the final fact embedding $\tilde{\mathbf{f}}_{tar}$ (i.e., context-aware fact vector $\tilde{\mathbf{f}}_{tar}$) of target fact f_{tar} is generated.

As shown in Fig. 2(d), after the final attentive feature aggregator function AGGREGATE_1 at search depth $k = 1$, the initial fact embedding \mathbf{f}_{tar} is combined with the aggregated neighborhood vector $\mathbf{h}_{\mathcal{N}_{tar}}^1$ to generate the context-aware fact feature $\tilde{\mathbf{f}}_{tar}$ which indicates the final embedding of the target fact. Finally, the score function $\psi(\mathbf{z})$ based on two fully connected layers and a sigmoid function assesses all fact vectors in training KG to predict their plausibility scores as follows:

$$\psi(\mathbf{z}) = \text{sigmoid}(\mathbf{W}_{f2} \text{ReLU}(\mathbf{W}_{f1}\mathbf{z} + \mathbf{b}_{f2}) + \mathbf{b}_{f2}). \quad (12)$$

5.2.2. Training

To effectively train the model (i.e., aggregator functions AGGREGATE_k , fact feature extractor $\phi(\cdot)$, and scoring function $\psi(\cdot)$), we employ negative sampling which randomly corrupts head and tail entities of each positive fact. With the negative and positive facts, binary cross-entropy loss is utilized to evaluate correct facts higher than the corrupted facts as follows:

$$\mathcal{L} = \sum_{(h,r,t,y) \in \mathcal{T}} y \log(\psi(\mathbf{z})) + (1 - y) \log(1 - \psi(\mathbf{z})), \quad (13)$$

where y is 1 for positive facts $\langle h, r, t \rangle$, otherwise, y is 0 for negative ones.

5.2.3. Inference

Given a target fact f_{tar} , the inference phase applies previously-learned aggregator functions AGGREGATE_k , fact feature extractor $\phi(\cdot)$, and scoring function $\psi(\cdot)$ to the target fact. Specifically, by using the fact feature extractor, fact feature information \mathbf{f}_{tar} of f_{tar} is extracted from its entity descriptions, type information, and names as described in Section 5.1. Note that the fact feature information of other facts in a training KG is already extracted at the training phase.

After the attentive feature aggregator function AGGREGATE_K at final search depth $k = K$, the initial fact feature information \mathbf{f}_{tar}^1 has fact feature information of every neighbor within K -hop. The final representation output $\tilde{\mathbf{f}}_{tar}^K$ at depth K is denoted by \mathbf{z}_{tar} , which indicates the final fact vector of the target fact. Finally, we evaluate the fact vector \mathbf{z}_{tar} with the scoring function $\psi(\mathbf{z})$ which consists of two fully connected layers and a sigmoid function. The plausibility score represents whether f_{tar} is valid or not. In open-world scenarios, a target fact includes out-of-KG entities and relation except for one entity (i.e. at least one entity in the target fact should be included). In a closed-world assumption, a target fact includes all in-KG entities and a relation.

5.2.4. Complexity analysis

We perform the space and time complexity of the proposed IKGE model. First, there are learnable parameters $\{\mathbf{W}_{c1}, \mathbf{W}_{c2}\} \in \mathbb{R}^{d \times k \times d}, \mathbf{W}_a \in \mathbb{R}^{d \times d}, \mathbf{W}_p \in \mathbb{R}^{d \times 2d}, \mathbf{b}_p \in \mathbb{R}^d$ in the fact feature information extraction. Also, $\mathbf{w}_a^k \in \mathbb{R}^{2d}, \forall k \in \{1, \dots, K\}$ for aggregator functions in the attentive feature aggregation process. Thus, the space complexity is summarized as $O(kd^2 + Kd)$, where k , d , and K denote the width of convolution kernel, embedding size, and the number of aggregation layers, respectively.

The time complexity of the fact feature information extraction may be expressed as $O(|E|nkd^2)$, where $|E|$ is the number of edges (i.e., facts), n is the length of input description, and k is the width of the convolution kernel. Specifically, the time complexities of subprocess are $O(2|E|nkd^2)$ for \mathbf{D}_e , $O(|E|nd^2)$ for attention scores \mathbf{A} , $O(|E|nd)$ for attention weights \mathbf{Av} , and so on. In the attentive feature aggregation process, the overall time complexity is expected to be $O(K(|V|d^2 + |E|d))$ which is proven by [44], where $|V|$ denotes the number of nodes and K denotes the number of aggregation layers.

Most conventional KGC models for an open-world setting do not exploit neighborhood information. Although the complexity of the proposed model increases, we combine the fact feature information within K-hop neighbors, resulting in high prediction performance.

6. Experiments

6.1. Experimental setup

The trainable parameters of fact feature information extraction and attentive feature aggregation, excluding pre-trained word embeddings, were tuned by the stochastic gradient descent with shuffled mini-batches and AdamW update rule with the initial learning rate of 0.01. In addition, a cosine annealing learning rate scheduler was adopted. AdamW is an adaptive gradient-based optimizer to adaptively adjust the learning rate for each parameter. It resolves the problem of Adagrad where the learning rate radically decreases or increases. All experiments were implemented in Python using the open-source library PyTorch which provides GPU-accelerated deep learning tools on Intel Core i7-10700 CPU and NVIDIA GeForce GTX 3080 GPU.

6.1.1. Real-world datasets

To evaluate the KGC task including link (entity and relation) prediction and triple classification sub-tasks, there are commonly used real-world KG datasets under the closed-world assumption: FB15k [26] and FB13 [46] which are constructed from Freebase [5]. To simulate open-world KGC scenarios, FB20k [15] and DBpedia50k/DBpedia500k [14] are proposed, which contain test facts with out-of-KG entities. FB20k is built upon FB15k by adding new entities with entity descriptions. DBpedia50k/DBpedia500k are constructed from DBpedia, where randomly sampled entities and triples are used for training triples, and the remaining triples are used for test triples.

In a transductive setup, all entities and relations in test sets must be in-KG like the existing KG datasets. On the other hand, since IKGE supports out-of-KG entities and relations, test sets for an inductive setup contain at least one out-of-KG entity or relation. To do this, we randomly selected out-of-KG relations based on uniform sampling over all in-KG relations. As a result, the test set is split into 8 fact types: O-O-O, O-O-X, O-X-O, X-O-O, O-X-X, X-X-O, X-O-X, and X-X-X, where O and X indicate in-KG and out-of-KG, respectively. Note that, since we assume that a test fact involves at least one in-KG entity, we do not focus on the X-X-X type. [Table 1](#) illustrates the statistics of five real-world datasets used in our experiments.

Although the datasets contain substantial facts with long textual descriptions, the proposed model IKGE requires some side information to support open-world KGC, such as relation-related (names and type constraints) and entity-related (descriptions and types) side information. We newly augmented FB20k, DBpedia50k, and DBpedia500k as FB20k+, DBpedia50k+, and DBpedia500k+ for both inductive and transductive setups.

6.1.2. Baselines

State-of-the-art KGC methods in an open-world setting generally leverage pre-trained word embeddings from textual descriptions to capture correlations among head entity, relation, and tail entity. Embeddings of head and tail entities are generated from their textual descriptions. Thus, these models evaluate the validity of a given triple involving in-KG as well as out-of-KG entities. The open-world KGC methods compared with IKGE are as follows:

- **Target Filtering Baseline:** is introduced by [14] to compare with ConMask. The approach only evaluates test facts $\langle ?, r, t \rangle$ and $\langle t, r, ? \rangle$ where target entity candidate t has been connected with relation r in the training set, otherwise the test facts are filtered.

Table 1

Statistics of five real-world datasets.

		Dataset				
		FB15k	FB20k+	FB13	DBpedia50k+	DBpedia500k+
#Entites	In-KG	14,904	14,904	75,043	49,900	517,475
	Out-of-KG	–	5,019	–	5,699	49,882
#Relations	In-KG	1,341	1,341	13	654	654
	Out-of-KG	–	200	–	96	96
#Train		472,860	472,860	316,232	32,388	3,102,677
#Valid		48,991	48,991	5,908	399	10,000
#Test	O-O-O	57,803	51,280	23,733	2,001	296,225
	O-O-X	–	9,543	–	4,238	286,763
	X-O-O	–	15,995	–	2,862	373,368
	O-X-O	–	6,523	–	321	52,274
	O-X-X	–	2,043	–	814	50,605
	X-X-O	–	2,758	–	473	65,888

- **Semantic Averaging:** is introduced by [14] to compare with ConMask. The approach is a simple version of ConMask, which only uses a part of feature extraction modules.
- **DKRL (2-layer CNN)** [15]: proposes two versions of description-based entity embedding generators, a 2-layer CNN encoder and a CBOW encoder. Among them, we adopt the 2-layer CNN encoder which shows better performance.
- **ConMask** [14]: further employs a multi-layer CNN with relation-dependent attention to extract entity features from descriptions. Moreover, the extracted entity features are combined with features simply extracted from a relation name.
- **Cmplx-OWE-300** [47]: first learns KGC embeddings with existing transductive KGC models under the closed-world assumption. Then, they learn a transformation function which transforms from description-based entity embeddings (300 dimensions) to KGC-based entity embeddings. Thus, KGC-purposed entity embeddings can be generated from entity descriptions and the learned transformation function.

6.1.3. Hyperparameters

For both inductive and transductive setups, we decided hyperparameters with a grid search on validation datasets. We used two convolution layers, filter width of 3, dropout rate of 0.25, and L2 constraint of 0.001 in the relation-specific feature extraction. In the attentive feature aggregation, we considered the 2-hop neighboring facts (i.e., maximum search depth $K = 3$) for FB15k/FB13 and 3-hop neighboring facts for DBpedia50k+/DBpedia500k+. Fully connected layers for scoring the target fact's vector consists of 2 layers with 512, 256 dimensions. All words in the vocabulary \mathcal{W} were initialized with the pre-trained 300-dimensional Wikipedia2Vec embeddings trained by [48]. We did not train the word embeddings. For words not included in the pre-trained vectors, we initialized them with Kaiming initialization using a uniform distribution, which is also known as He initialization.

6.1.4. Performance measures

In entity and relation predictions, test facts' plausibility scores are computed under closed-world and open-world assumptions. Conventional closed-world KGC models (e.g., TransE, TransR) do not generate entity and relation embeddings from side information, so they cannot handle out-of-KG entities. All open-world KGC models (e.g., IKGE, DKRL, ConMask, and Cmplx-OWE-300) obviously perform closed-world entity prediction with test facts of the O-O-O pattern. In the open-world setting, head entity prediction evaluates test facts of the O-O-X pattern, and tail entity prediction evaluates test facts of the X-O-O pattern. In other words, for each test fact (X -O-? or ?-O-X) involving an open-world entity (X) and a target entity (?), open-world KGC models rank all known target entities by scoring the test facts. Since all the conventional open-world KGC models do not support out-of-KG relations, they limitedly evaluate O-O-X and X-O-O patterns. Note that, although the original datasets (DBpedia50k and DBpedia500k) include all the eight patterns, they only utilize O-O-X and X-O-O patterns for open-world scenarios and an O-O-O pattern for closed-world scenarios. On the other hand, the proposed IKGE can score test facts involving out-of-KG relations. Therefore, IKGE can perform extended entity prediction, head entity prediction for the O-O-X, O-X-X, and O-X-O patterns, and tail entity prediction for the X-O-O, X-X-O, and O-X-O patterns. Furthermore, IKGE is able to perform relation prediction for the O-O-X, X-O-O, and X-O-X patterns by ranking all in-KG relations $r \in \mathcal{R}$ for each test fact $\langle h, r, t \rangle$. Then, the target entities and relations' ranks are evaluated with the commonly used evaluation metrics: mean rank (MR), mean reciprocal rank (MRR), and HITS@k (as H@k in %). MR and MRR measure the average of the correct entities/relations' ranks and the correct entities/relations' reciprocal rank values, respectively. HITS@k is the average proportion of valid entities whose prediction score ranks are lower than k (10 for entity prediction or 1 for relation prediction). Lower MR, higher MRR and HITS@k indicate better performance. Following the previous open-world KGC approaches, the "Filtered" evaluation setting firstly reported in [26] was adopted, not "Raw". The "Filtered" setting ignores any explicit facts in KG datasets before predicting rank scores.

6.2. Experimental results

In this section, experimental results are reported on two main tasks based on the plausibility scores of facts, entity and relation prediction (i.e., link prediction) in Tables 2–4, and triple classification in Table 5. Furthermore, to study the relative importance of IKGE components, variant versions of IKGE from the point of ablation studies are further evaluated in the same experimental setup. For open-world KGC, IKGE without attention-based convolution (called IKGE_{No_ATT}), IKGE without type matching (called IKGE_{No_TM}), and IKGE without attentive feature aggregation (called IKGE_{No_AFA}) are evaluated as shown in Section 6.2.1 and Section 6.2.2. For closed-world KGC, IKGE without fact feature information extraction (called IKGE_{No_FFIE}) is evaluated as shown in Section 6.2.3 and Section 6.2.4. Instead of referring directly to other models' experimental results, comparative evaluations are conducted with their released implementations for open-world KGC models. The experimental results include extended experiments for more fact patterns than previous models. The OpenKE framework⁷ is used for most closed-world KGC models, which collects well-known KGC models under the closed-world assumption. For KGC models not in the OpenKE framework, all the models are evaluated with the same experimental setup stated in their papers. As described in Section 6.1.2, Cmplx-OWE-300 first obtains KGC-based embeddings in preprocessing and then trains a transformation function. Thus, the training time of Cmplx-OWE-300 is longer than other models, except for the proposed IKGE. On the other hand, once

⁷ OpenKE framework: <https://github.com/thunlp/OpenKE>.

Table 2

Head entity prediction on FB20k+, DBpedia50k+, and DBpedia500k+ in an open-world setting (O-O-X, O-X-X, and O-X-O patterns). Best results are in boldface.

	FB20K+			DBpedia50k+			DBpedia500k+		
	MR	H@10	MRR	MR	H@10	MRR	MR	H@10	MRR
Target Filtering Baseline	2243	0.05	0.08	605	0.07	0.07	20667	0.01	0.01
Semantic Averaging	1967	0.07	0.13	513	0.10	0.11	10245	0.04	0.09
DKRL (2-layer CNN)	1790	0.16	0.20	490	0.14	0.16	13776	0.07	0.13
ConMask	698	0.28	0.32	213	0.29	0.32	5983	0.14	0.30
Cmplx-OWE-300	592	0.25	0.35	206	0.35	0.31	4843	0.19	0.29
IKGE _{No_ATT}	1223	0.19	0.25	268	0.22	0.20	7421	0.11	0.18
IKGE _{No_TM}	583	0.28	0.34	198	0.33	0.32	4857	0.21	0.33
IKGE _{No_AFA}	654	0.26	0.32	324	0.28	0.27	6877	0.18	0.30
IKGE	545	0.30	0.39	185	0.38	0.34	3978	0.23	0.35

Table 3

Tail entity prediction on FB20k+, DBpedia50k+, and DBpedia500k+ in an open-world setting (X-O-O, X-X-O, and O-X-O patterns). Best results are in boldface.

	FB20K+			DBpedia50k+			DBpedia500k+		
	MR	H@10	MRR	MR	H@10	MRR	MR	H@10	MRR
Target Filtering Baseline	2848	0.04	0.05	121	0.23	0.11	3494	0.02	0.01
Semantic Averaging	2538	0.06	0.07	105	0.26	0.15	1539	0.16	0.12
DKRL (2-layer CNN)	1235	0.13	0.19	82	0.38	0.22	2732	0.08	0.09
ConMask	548	0.24	0.28	43	0.58	0.52	683	0.32	0.36
Cmplx-OWE-300	484	0.23	0.42	39	0.60	0.49	797	0.36	0.37
IKGE _{No_ATT}	892	0.17	0.21	62	0.39	0.31	1249	0.21	0.19
IKGE _{No_TM}	422	0.25	0.32	38	0.75	0.56	538	0.41	0.35
IKGE _{No_AFA}	621	0.20	0.27	44	0.61	0.52	653	0.35	0.34
IKGE	407	0.30	0.40	31	0.78	0.61	463	0.43	0.36

Table 4

Head and Tail Entity prediction on FB20k+, DBpedia50k+, and DBpedia500k+ in an open-world setting (X-O-O and O-O-X patterns). Best results are in boldface.

	FB20K+			DBpedia50k+			DBpedia500k+		
	MR	H@10	MRR	MR	H@10	MRR	MR	H@10	MRR
Target Filtering Baseline	2345	0.05	0.06	333	0.16	0.10	10423	0.02	0.02
Semantic Averaging	2122	0.08	0.12	271	0.19	0.15	5515	0.10	0.11
DKRL (2-layer CNN)	1412	0.18	0.22	260	0.27	0.21	8013	0.08	0.12
ConMask	601	0.30	0.31	104	0.44	0.44	3161	0.24	0.33
Cmplx-OWE-300	532	0.31	0.42	119	0.49	0.42	2672	0.28	0.35
IKGE _{No_ATT}	1018	0.19	0.24	145	0.32	0.27	4051	0.19	0.19
IKGE _{No_TM}	493	0.27	0.33	113	0.52	0.45	2396	0.33	0.34
IKGE _{No_AFA}	614	0.24	0.31	167	0.45	0.41	3511	0.28	0.32
IKGE	463	0.34	0.42	104	0.54	0.52	2136	0.38	0.37

Table 5

Relation prediction on FB20k+, DBpedia50k+, and DBpedia500k+ in an open-world setting (O-O-X, X-O-O, and X-O-X patterns). Best results are in boldface.

	FB20K+			DBpedia50k+			DBpedia500k+		
	MR	H@1	MRR	MR	H@1	MRR	MR	H@1	MRR
Semantic Averaging	421	0.07	0.09	213	0.04	0.11	285	0.03	0.07
DKRL (2-layer CNN)	262	0.15	0.16	130	0.09	0.12	153	0.11	0.10
ConMask	142	0.23	0.31	95	0.29	0.25	103	0.23	0.28
Cmplx-OWE-300	159	0.24	0.29	94	0.27	0.27	114	0.25	0.30
IKGE _{No_ATT}	242	0.19	0.21	125	0.14	0.20	131	0.14	0.22
IKGE _{No_TM}	136	0.27	0.34	88	0.31	0.30	92	0.23	0.33
IKGE _{No_AFA}	146	0.25	0.33	91	0.29	0.28	98	0.22	0.32
IKGE	135	0.29	0.36	85	0.32	0.31	83	0.24	0.34

the training is completed, the inference times of Cmplx-OWE-300 and DKRL are similar and fastest among other models. The reason is that they simply score test facts via a translation-based scoring function in TransE. The proposed IKGE took the longest training and inference time than all other models. Since IKGE and ConMask have similar processing leveraging pre-trained word

embeddings from descriptions, the time for fact feature information in IKGE and the time for the whole scoring process in ConMask are similar. However, IKGE additionally employs an attentive feature aggregation module which takes into account the entire KG structure. This makes training and inference times take a long time.

6.2.1. Open-world entity prediction

To evaluate open-world entity prediction, all open-world KGC methods predict head entities for the O-O-X, O-X-X, and O-X-O patterns and tail entities for the X-O-O, X-X-O, and O-X-O patterns, including out-of-KG entities and relations. Following ConMask [14,47], target filtering is adopted for all open-world KGC methods, which evaluates only the candidate entities whose relation-entity combinations exist in the training KG. It reduces the computation complexity and helps to evaluate only reasonable facts. ConMask, Cmplx-OWE-300, and DKRL assumed that descriptions and names of all in-KG and out-of-KG entities are given. In the same manner, we utilize the entity descriptions and names as well as relation-related side information for fact feature information extraction and neighborhood information for attentive feature aggregation. Since IKGE can identify the facts with out-of-KG relations by using the abundant relation-related side information, it is able to handle the out-of-KG relations regarding head/tail entity prediction for O-X-X, O-X-O, and X-X-O patterns.

The state-of-the-art open-world KGC approaches (ConMask, DKRL, and Cmplx-OWE-300) do not consider out-of-KG relations. To support the out-of-KG relations regarding head/tail entity prediction for O-X-X, O-X-O, and X-X-O patterns, we differently handled the out-of-KG relations for each KGC approach except for ConMask and Semantic Averaging. For Semantic Averaging and ConMask, their original versions were used. The original versions simply average pre-trained word embeddings from relation names to identify the validity among the head entity, relation, and tail entity. Note that, although they can score facts involving out-of-KG relations, they did not conduct and report the head/tail entity prediction for O-X-X, O-X-O, and X-X-O patterns. Inspired by them, for DKRL and Cmplx-OWE-300, pre-trained word embeddings from relation names are averaged to obtain relation embeddings firstly. Uniquely assigned relation embeddings in DKRL (original version) are replaced with the word-embedding-based relation embeddings. The original version of Cmplx-OWE-300 learns a transformation function from KGC-based pre-trained entity embeddings to word-embedding-based entity embeddings. We further employed a trainable relation-specific transformation function, which transforms KGC-based pre-trained relation embeddings to word-embedding-based relation embeddings (from relation names). For Target Filtering Baseline, the same process in ConMask is applied, which assigns random scores to test facts that pass the target filtering. Thus, IKGE can be compared with state-of-the-art approaches in an open-world setting with out-of-KG relations.

Link prediction performance for head and tail entities is presented in [Table 2](#) and [Table 3](#), respectively. Among the Target Filtering Baseline [14], Semantic Averaging [14], DKRL [15], ConMask [14], and Cmplx-OWE-300 [47] approaches, the results of Target Filtering Baseline and Semantic Averaging were always worse than all other models. These models are the simplified models that randomly assign to test facts or combine pre-trained word embeddings from descriptions and names with averaging. In contrast, ConMask generates entity embeddings by attention-based context masking and CNN-based target fusion operation. Thus, Semantic Averaging cannot capture important word semantics among input sentences. Although DKRL, ConMask, and IKGE exploited entity descriptions to generate entity vectors, the result of DKRL is significantly worst. The reason is that DKRL does not model the entity descriptions according to relation-related information. On the other hand, both ConMask and IKGE utilize relation names and entity descriptions. In addition, [Table 4](#) shows the head and tail prediction performance for out-of-KG entities only (i.e., head and tail prediction for O-O-X and X-O-O patterns, respectively), whereas [Table 2](#) and [Table 3](#) target O-O-X, O-X-X, and O-X-O patterns and O-O-X, O-X-X, and O-X-O patterns, respectively. The overall performance in [Table 4](#), including out-of-KG relations, is higher than the average performance in [Tables 2 and 3](#). Since relation-related information is insufficient compared to entities, entity prediction on out-of-KG relations is more complicated than out-of-KG entities.

Among IKGE_{No_ATT}, IKGE_{No_TM}, and IKGE_{No_AFA}, IKGE_{No_ATT} shows the lowest performance. This means that the attention-based convolution module is more important than the other modules, because the module extracts initial fact features from entity descriptions attended by relation-related side information. Similar to DKRL, ConMask, and Cmplx-OWE-300, the extracted features from descriptions play a very important role to capture the correlation among head entity, relation, and tail entity. Note that the attention-based convolution module is different from the previous open-world KGC models, because the module performs entity/relation-related attention with relation-related side information. Thus, IKGE_{No_AFA} performs better than the previous open-world KGC models. The impact of the type matching module is relatively lower than the impact of the attentive feature aggregation module. However, both modules are not important compared to the attention-based convolution module. IKGE_{No_TM} outperforms all other variants of IKGE and shows similar performance to IKGE. The type matching module provides some performance improvement, whereas the attentive feature aggregation module provides a significant performance improvement. In summary, the attention-based convolution module is the most important component in IKGE.

6.2.2. Open-world relation prediction

To evaluate open-world relation prediction, all open-world KGC methods predict relations for the O-O-X, X-O-O, and X-O-X patterns, including out-of-KG entities. The target filtering was not applied (also Target Filtering Baselines) because the challenging X-O-X pattern has no relation-entity combination in the training set. As shown in [Table 5](#), we observe that IKGE shows the best performance for relation prediction across all three datasets by a substantial margin. Among previous open-world KGC models, ConMask shows better performance than Cmplx-OWE-300. Cmplx-OWE-300 mainly targets the trans-

formation from description-based entity embeddings to KGC-based entity embeddings, leading to loss of relation-related semantic information to identify a given fact (head entity, relation, and tail entity). We also find that IKGE significantly outperforms IKGE_{No_AFA} across all datasets, especially FB20K+ having more neighborhood information. Since attentive feature aggregation accumulates information from neighboring facts, IKGE can more easily determine the validity of target facts than IKGE_{No_AFA}. As like open-world entity prediction, the attention-based convolution and attentive feature aggregation modules are important components in IKGE by observing the results of IKGE_{No_ATT} and IKGE_{No_AFA}.

6.2.3. Closed-world entity prediction

IKGE is also compared with other well-known closed-world KGC models targeting only in-KG entities and relations, as shown in Table 6. To evaluate closed-world KGC, we first assume that all test facts' entities and relations have previously appeared in training. As shown in Section 6.1.1, the pattern of the test facts is an O-O-O pattern, which only involves in-KG entities and relations. Since the closed-world entity prediction is less challenging than the open-world entity/relation prediction, the proposed IKGE (inductive version) can be directly compared against existing transductive KGC models under the closed-world assumption. Unlike the open-world setup, the closed-world KGC tasks are performed with entity prediction on the original datasets (i.e., FB15k, DBpedia50k, and DBpedia500k) similar to [14].

Moreover, IKGE_{No_FFI} is introduced, a variant version of IKGE, which no longer employs a complex encoder to generate relation-specific representations of facts from pre-trained word embeddings with relation-related side information. IKGE_{No_FFI} cannot generalize the facts involving out-of-KG entities and relations. Instead, a simple encoder of IKGE_{No_FFI} first assigns a unique embedding to each entity and relation and then generates initial fact vectors with simple RNNs. The unique embeddings are optimized similarly to conventional closed-world KGC models. IKGE_{No_FFI} is a topology-based model which mainly incorporates the graph structure information into the aggregated features by utilizing multi-hop neighborhoods. In summary, IKGE inductively generates initial fact vectors from pre-trained word embeddings (via fact feature information extraction), whereas IKGE_{No_FFI} assigns randomly initialized vectors to entities and relations like the existing closed-world KGC models. We find that description-based KGC models IKGE, DKRL, ConMask, and Cmplx-OWE-300 mostly outperform closed-world methods for mean rank. These models especially show improvements on the DBpedia-derived datasets which are sparse graphs. The Transductive models TransE and TransR mainly rely on independent triples rather than neighboring facts, making it more difficult to predict fact validity due to less interconnected facts. Although IKGE_{No_FFI} is also a transductive model, IKGE_{No_FFI} outperforms by a small margin TransE and TransR. This demonstrates that further exploiting neighborhood information provides a performance improvement. The proposed IKGE shows better performance than all other models significantly. We believe that this is due to neighborhood information accumulated with the attentive feature aggregation modules. .

6.2.4. Closed-world triple classification

We further analyzed the triple classification performance of the compared conventional KGC models under closed-world assumption, as shown in Table 7. As described in [46], the goal of the triple classification task is determining whether a given fact is correct or not (i.e., binary classification). With a relation-specific threshold, test facts are classified into positive (1) or negative (0). The evaluation results are directly taken from [49]. From the results on FB13, the full version of IKGE outperforms all other models. Unlike other models, IKGE, IKGE_{No_FFI} and TransE-NMM additionally utilize neighborhood information (i.e., neighboring facts) to determine the validity of a given fact. These models perform better in most cases than the other models by leveraging neighborhood information. As aforementioned, since IKGE_{No_FFI} accumulates entity and relation embeddings in multi-hop neighborhood as contextual information via the attentive feature aggregation module in IKGE, IKGE_{No_FFI} can preserve the graph structure for more informative fact embeddings. TransE-NMM [50] is also one of the state-of-the-art topology-based KGC models utilizing neighborhood information. However, they consider only 1-hop neighbors as local structural information and simply combine them. Therefore, TransE-NMM showed relatively low performance compared to IKGE_{No_FFI}. We also observed that IKGE_{No_FFI} shows higher classification performance than other comparisons

Table 6

Head and Tail Entity prediction on FB15k, DBpedia50k, and DBpedia500k in a closed-world setting (O-O-O pattern). Best results are in boldface.

	FB15k				DBpedia50k				DBpedia500k			
	Head		Tail		Head		Tail		Head		Tail	
	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10	MR	H@10
TransE	189	0.68	92	0.75	2854	0.37	734	0.68	10034	0.15	2472	0.45
TransR	186	0.71	87	0.77	2689	0.39	718	0.67	–	–	–	–
DKRL (2-layer CNN)	175	0.70	83	0.60	1867	0.40	262	0.65	2523	0.17	1845	0.19
ConMask	116	0.62	80	0.62	1063	0.41	141	0.72	1512	0.21	1568	0.20
Cmplx-OWE-300	121	0.64	81	0.61	1043	0.39	187	0.73	1745	0.22	1499	0.27
IKGE _{No_FFI}	156	0.64	88	0.65	1743	0.38	538	0.69	5983	0.16	2195	0.24
IKGE	113	0.70	81	0.75	971	0.42	135	0.74	1486	0.24	1479	0.33

Table 7

Triple classification on FB13 in a closed-world setting (O-O-O pattern). Best results are in boldface.

Accuracy (%)	
TransR	82.5
TransD	89.1
TransE	87.6
TransE-comp	87.6
NTN	87.2
Bilinear-comp	86.1
KG2E	85.3
TransH	83.3
TransE-NMM	88.6
IKGE _{No_FFIE}	89.1
IKGE	90.3

except for IKGE. This demonstrates the importance of the attentive feature aggregation module, rather than simply combining neighborhood information in TransE-NMM. .

7. Conclusions

In this paper, we proposed an inductive knowledge graph embedding (IKGE) for the open-world KGC task based on attentively aggregating feature information from neighboring facts. IKGE generalizes to facts including out-of-KG entities and relations by inductively generating fact embeddings from entity descriptions. Since different words in entity descriptions usually have different informativeness for representing facts, the model attends to relation-related side information. Also, the model exploits relation-specific type constraints to confine unseen relations in an end-to-end manner. Moreover, the proposed attentive feature aggregation hierarchically accumulates the features of multi-hop neighbors. Therefore, IKGE preserves global structure information, leading to more accurate KGC in open-world scenarios. Experimental results show that IKGE outperforms existing approaches in both transductive and inductive setups by successfully aggregating neighborhood features.

In the future, we will introduce generative models such as VAEs and GANs which leverage side information of entities and relations to adjust biases towards seen and unseen facts. We will discuss how to generate extra facts in the training phase by exploiting extensive external knowledge.

CRediT authorship contribution statement

Byungkook Oh: Conceptualization, Methodology, Software, Writing – original draft. **Seungmin Seo:** Software, Formal analysis. **Jimin Hwang:** Data curation, Investigation. **Dongho Lee:** Writing – review & editing. **Kyong-Ho Lee:** Supervision, Project administration.

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.

Acknowledgement

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP; Ministry of Science, ICT & Future Planning) (No. NRF-2019R1A2B5B01070555).

References

- [1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, et al, DBpedia—a large-scale, multilingual knowledge base extracted from wikipedia, Semantic Web 6 (2) (2015) 167–195.
- [2] F.M. Suchanek, G. Kasneci, G. Weikum, YAGO, a core of semantic knowledge, in: Proceedings of the 16th international conference on World Wide Web, 2007, pp. 697–706.
- [3] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, T.M. Mitchell, Toward an architecture for never-ending language learning, in: Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.
- [4] D. Vrandečić, M. Krötzsch, Wikidata: a free collaborative knowledge base, Commun. ACM 57 (10) (2014) 78–85.
- [5] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, 2008, pp. 1247–1250.
- [6] G.A. Miller, WordNet: a lexical database for english, Commun. ACM 38 (11) (1995) 39–41.
- [7] S. Shin, K. Lee, Processing knowledge graph-based complex questions through question decomposition and recomposition, Inf. Sci. 523 (2020) 234–244.

- [8] X. Guo, W. Lin, Y. Li, Z. Liu, L. Yang, S. Zhao, Z. Zhu, DKEN: deep knowledge-enhanced network for recommender systems, *Inf. Sci.* 540 (2020) 263–277.
- [9] Z. Lin, L. Feng, R. Yin, C. Xu, C.K. Kwok, GLIMG: global and local item graphs for top-n recommender systems, *Inf. Sci.* 580 (2021) 1–14.
- [10] R. Zhang, Y. Mao, W. Zhao, Knowledge graphs completion via probabilistic reasoning, *Inf. Sci.* 521 (2020) 144–159.
- [11] N. Shanavas, H. Wang, Z. Lin, G.I. Hawe, Ontology-based enriched concept graphs for medical document classification, *Inf. Sci.* 525 (2020) 172–181.
- [12] D. Wang, H. Fan, J. Liu, Learning with joint cross-document information via multi-task learning for named entity recognition, *Inf. Sci.* 579 (2021) 454–467.
- [13] E. Amador-Domínguez, E. Serrano, D. Manrique, P. Hohenecker, T. Lukasiewicz, An ontology-based deep learning approach for triple classification with out-of-knowledge-base entities, *Inf. Sci.* 564 (2021) 85–102.
- [14] B. Shi, T. Weninger, Open-world knowledge graph completion, *AAAI* (2018).
- [15] R. Xie, Z. Liu, J. Jia, H. Luan, M. Sun, Representation learning of knowledge graphs with entity descriptions, *AAAI* (2016) 2659–2665.
- [16] M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs, *Proc. IEEE* 104 (1) (2016) 11–33.
- [17] A. Neelakantan, B. Roth, A. McCallum, Compositional vector space models for knowledge base completion, *ACL* (2015) 156–166.
- [18] S. Mazumder, N. Ma, B. Liu, Towards a continuous knowledge learning engine for chatbots, arXiv preprint arXiv:1802.06024.
- [19] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, *NIPS* (2017) 1024–1034.
- [20] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, arXiv preprint arXiv:1709.05584.
- [21] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, *IEEE Trans. Knowl. Data Eng.* 29 (12) (2017) 2724–2743.
- [22] T. Hayashi, H. Fujita, Cluster-based zero-shot learning for multivariate data, *J. Ambient Intell. Humanized Comput.*
- [23] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online learning of social representations, *SIGKDD* (2014) 701–710.
- [24] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: *KDD*, 2016, pp. 855–864.
- [25] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, *WWW* (2015) 1067–1077.
- [26] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, *NIPS* (2013) 2787–2795.
- [27] B. Yang, W.T. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in: *ICLR*, 2015..
- [28] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: *ICML*, 2016, pp. 2071–2080..
- [29] B. Oh, S. Seo, K.H. Lee, Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods, in: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ACM, 2018, pp. 257–266.
- [30] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: *ICML*, 2016, pp. 2014–2023..
- [31] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: *ICLR*, 2014..
- [32] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, *ICLR* (2017).
- [33] M. Schlichtkrull, T.N. Kipf, P. Bloem, R.V.D. Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, *ESWC* (2018) 593–607.
- [34] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: *ICLR*, 2015..
- [35] M.T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in: *EMNLP*, 2015..
- [36] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, *NIPS* (2014) 3104–3112.
- [37] C. dos Santos, M. Tan, B. Xiang, B. Zhou, Attentive pooling networks, arXiv preprint arXiv:1602.03609.
- [38] W. Yin, H. Schütze, B. Xiang, B. Zhou, Abcn: Attention-based convolutional neural network for modeling sentence pairs, *TACL* 4 (2016) 259–272.
- [39] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Comput.* 9 (8) (1997) 1735–1780.
- [40] Y. Kim, Convolutional neural networks for sentence classification, *EMNLP* (2014) 1746–1751.
- [41] Z. Lin, M. Feng, C.N. dos Santos, M. Yu, B. Xiang, B. Zhou, Y. Bengio, A structured self-attentive sentence embedding, *ICLR* (2017).
- [42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, *NIPS* (2017) 5998–6008.
- [43] J. Cheng, L. Dong, M. Lapata, Long short-term memory-networks for machine reading, in: *EMNLP*, 2016..
- [44] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: *ICLR*, 2017..
- [45] B.L. Douglas, The weisfeiler-lehman method and graph isomorphism testing, arXiv preprint arXiv:1101.5211..
- [46] R. Socher, D. Chen, C.D. Manning, A. Ng, Reasoning with neural tensor networks for knowledge base completion, *NIPS* (2013) 926–934.
- [47] H. Shah, J. Villmow, A. Ulges, U. Schwanecke, F. Shafait, An open-world extension to knowledge graph completion models, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 3044–3051..
- [48] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, Y. Matsumoto, Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Association for Computational Linguistics, 2020, pp. 23–30..
- [49] D.Q. Nguyen, An overview of embedding models of entities and relationships for knowledge base completion, arXiv preprint arXiv:1703.08098..
- [50] D.Q. Nguyen, K. Sirts, L. Qu, M. Johnson, Neighborhood mixture model for knowledge base completion, *CoNLL* (2016) 40–50.