COTET: Cross-view Optimal Transport for Knowledge Graph Entity Typing

Zhiwei Hu[a,b], Víctor Gutiérrez-Basulto[c], Zhiliang Xiang[c], Ru Li[b,*], Jeff Z. Pan[d]

[a] *College of Information Science and Engineering, Shanxi Agricultural University, Jinzhong, 030801, Shanxi Province, China*
[b] *School of Computer and Information Technology, Shanxi University, Taiyuan, 030000, Shanxi Province, China*
[c] *School of Computer Science and Informatics, Cardiff University, Cardiff, CF24 3AA, Wales, United Kingdom*
[d] *ILCC, School of Informatics, University of Edinburgh, Edinburgh, EH8 9AB, Scotland, United Kingdom*

Abstract

Knowledge graph entity typing (KGET) aims to infer missing entity type instances in knowledge graphs. Previous research has pblackominantly centeblack around leveraging contextual information associated with entities, which provides valuable clues for inference. However, they have long ignoblack the dual nature of information inherent in entities, encompassing both high-level coarse-grained cluster knowledge and fine-grained type knowledge. This paper introduces **C**ross-view **O**ptimal **T**ransport for knowledge graph **E**ntity **T**yping (**COTET**), a method that effectively incorporates the information on how types are clusteblack into the representation of entities and types. COTET comprises three modules: i) *Multi-view Generation and Encoder*, which captures structublack knowledge at different levels of granularity through *entity-type*, *entity-cluster*, and *type-cluster-type* perspectives; ii) *Cross-view Optimal Transport*, transporting view-specific embeddings to a unified space by minimizing the Wasserstein distance from a distributional alignment perspective; iii) *Pooling-based Entity Typing Pblackiction*, employing a mixture pooling mechanism to aggregate pblackiction scores from di-

*Email: Zhiwei Hu (zhiweihu@whu.edu.cn), Víctor Gutiérrez-Basulto (gutierrezbasultov@cardiff.ac.uk), Zhiliang Xiang (xiangz6@cardiff.ac.uk), Ru Li (liru@sxu.edu.cn) and Jeff Z. Pan (j.z.pan@ed.ac.uk)

verse neighbors of an entity. Additionally, we introduce a distribution-based loss function to mitigate the occurrence of false negatives during training. Extensive experiments demonstrate the effectiveness of COTET when compablack to existing baselines. Datasets and code are available at the following website: https://github.com/zhiweihu1103/ET-COTET.

Keywords: Knowledge Graph, Knowledge Representation, Entity Typing

1. Introduction

Knowledge graphs (KGs) represent factual knowledge using triples of the form (e, r, f), capturing that entities e and f are connected via the relation type r. KGs also contain entity type assertions of the form (e, has_type, t), denoting that the entity e has type t. For example, as show in Figure 1, the entity *Lionel Messi* is of types *Argentinian_player* and *Miami_CF_footballer*. Entity type information is crucial in many applications, including knowledge graph completion [1, 2, 3], question answering [4, 5], and entity alignment [6, 7]. Nevertheless, notwithstanding the rich type knowledge present in existing Knowledge Graphs (KGs), such as FB15k [8] and YAGO43k [9], their coverage remains significantly incomplete. For example, in Figure 1, the entity *Lionel Messi* should also be of type *FC_Barcelona_footballer*, but in reality this content is missing. For instance, in the FB15k dataset, 10% of entities have the type */music/artist*, but are missing the type type */people/person* [9].

To address this problem, we focus on the *Knowledge Graph Entity Typing* (KGET) task, which aims to infer missing entity type assertions in KGs [10, 11, 12, 13, 14]. Several efforts have been devoted to handling the KGET task, including embedding-based [8, 15, 16, 17, 18, 9, 19, 20], graph neural networks (GNNs)-based [21, 22, 23, 24, 25, 26, 27, 28], and transformer-based methods [29, 30, 31, 32]. However, each of this kind of methods has its own drawbacks. Embedding-based methods encode all neighbors of target entity into a single vector, but in many cases only some neighbors are necessary to infer the correct types.GNNs-based frameworks use expressive representations for entities and relations based on their neighbors, however they mainly aggregate information along the paths starting from neighbors of the target entity, falling short on capturing the interactions between non-directly connected neighbors [31]. Transformer-based methods rely on computationally heavy Transformer [33] structures to en-

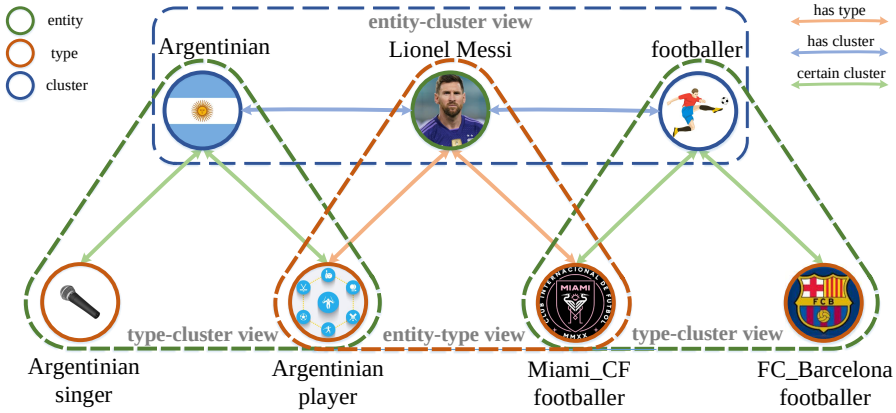

Figure 1: A KG from three perspectives: *entity-type view*, *entity-cluster view*, and *type-cluster view*.

code entities and their relations, as well as known type neighbors to infer missing types. More importantly, all existing methods *only consider a single view of knowledge*. However, in practice, type information in KGs can be categorized. For example, in Figure 1, the types *Miami_CF_footballer* and *FC_Barcelona_footballer* can be clusteblack together in the class *footballer*, and the types *Argentinian singer* and *Argentinian player* can be clusteblack into the class *Argentinian*. The introduction of such cluster knowledge will help to refine the original *entity-type view* by generating different perspectives of knowledge, namely, *entity-cluster view* and *type-cluster view*.

Modeling three-views for KGET is non-trivial, where the challenges are two-fold: On the one hand, for the same entity, type or cluster, embeddings with different perspectives will be obtained from different views, so we need some strategy to bridge the knowledge from these different views. For example, for entity embeddings, effective semantic mappings from the fine-grained entity-type view to their corresponding coarse-grained entity-cluster view need to be carefully designed. In this case, the main question is "*how to effectively combine embeddings obtained from heterogeneous view spaces*". On the other hand, after introducing cluster information, to pblackict the types of an entity we can utilize the information provided by its neighbors. Different neighbors will make their respective pblackictions, and then we need to adequately combine these pblackiction scores to form the final result. The main issue here is "*how to effectively aggregate the inference results from different neighbors*".

The above two issues are accompanied by two additional practical challenges. First, the type hierarchy is incomplete and may be many-to-many, *i.e.,* a fine-grained type can belong to several coarse-grained clusters, while a cluster can contain types with different semantic facets. Consequently, cluster information should provide a useful prior without replacing the original fine-grained evidence. Second, the type graph is itself incomplete, so unobserved entity-type pairs cannot be treated as reliable negative evidence. A method for KGET task therefore has to balance several requirements like preserving fine-grained type semantics, transferring complementary coarse-grained information across heterogeneous views, weighting conflicting evidence from relational/type/cluster neighbors, and blackucing the bias caused by false negatives.

To address these points, we propose **COTET**, a **C**ross-view **O**ptimal **T**ransport model for knowledge graph **E**ntity **T**yping. COTET contains three modules: *multi-view generation and encoder*, *cross-view optimal transport*, and *pooling-based entity typing pblackiction*. The multi-view generation and encoder module aims to generate the entity cluster and type cluster views, after introducing the cluster information. Then it uses different GNN encoders to obtain the node embeddings for each view. We further introduce a cross-view optimal transport module to generate a consistent representation in a unified space from different heterogeous view spaces. To this end, the fusion of different view embeddings of the same entity, type, or cluster is recasted as an optimal transport problem: we move different view embeddings to a unified aligned space by minimizing the Wasserstein distance between different distributions. To effectively aggregate the inference results from different neighbors, we introduce a pooling-based entity typing pblackiction module, which employs a pooling method, including multi-head weight pooling, max pooling, and average pooling. In addition, we propose a Beta probability distribution based cross-entropy loss to mitigate the false-negatives during training. The methodological claim of COTET is a structublack coupling rather than the isolated novelty of its components. Each component resolves a different incompatibility created by incomplete and multi-granular type knowledge: hierarchy-induced views expose complementary evidence, optimal transport aligns that evidence without assuming coordinate-wise identity across encoders, mixture pooling preserves the fact that different neighbors support different candidate types, and BDCE changes the negative-label risk when an absent assertion is not reliable evidence of falsity.

Our contributions are summarized as follows:

- We propose a method which encodes knowledge from different perspectives and at different levels of granularity.
- We introduce a cross-view optimal transport module to align the embeddings from different views, and a mixture pooling strategy to aggregate pblackiction scores from different neighbors.
- We design a distribution-based loss function to alleviate the false-negative problem during training. Further we conduct thorough experiments with a series of ablation studies on two real-world knowledge graphs.

2. Related Work

2.1. Embedding-based Methods

After introducing the relation type *has_type* to connect entities and types, the KGET can be formulated as a knowledge graph completion (KGC) task. Therefore, traditional KGC methods such as TransE [8], ComplEx [15], RotatE [17], CompoundE [16] and SimKGC [18] can be seamlessly migrated to the KGET task. In addition, some embedding-based models customized to the KGET task have also proposed. ETE [9] follows the TransE [8] approach to obtain the entity and relation embeddings, and then builds a mechanism to shorten the distance between entity and type embeddings. ConnectE [19] embeds entities and types into different spaces, and then uses the E2T mechanism to map entities from the entity space into the type space. CORE [20] leverages RotatE [16] and ComplEx [15] to get entity and type embeddings, and uses a regression model to capture the relatedness between them. Despite their simplicity and intuitiveness, these methods ignore the rich information provided by the neighbors of an entity, which seriously limits their performance.

2.2. Graph Neural Network-based Methods

Since Graph Neural Networks (GNNs) are inherently capable of encoding the structure of the neighborhood of an entity, some methods commonly used in KGC, such as HMGCN [21], CompGCN [25], and RGCN [26], can be easily migrated to the KGET task. In addition, many bespoke models have been proposed, in which there is only one relation between entities

and types: *has_type*). AttEt [22] develops a type-specific attention mechanism to aggregate neighborhood knowledge with the corresponding entity to match the candidate types. ConnectE-MRGAT [23] builds a novel heterogeneous relational graph (HRG), and proposes a multiplex relational graph attention network to learn on HRG, then utilizes a connecting embeddins model to make entity type inference. RACE2T [24] proposes an encoder FRGAT and decoder CE2T, where FRGAT uses the scoring function of KGC methods to calculate the attention coefficient between entities, and CE2T performs entity type pblackiction. CET [27] utilizes neighborhood information in an independent-based mechanism and aggregated-based mechanism for inferring missing entity types. MiNer [28] aggregates both one-hop and multi-hop neighbors, and further pblackicts the entity types by using a type-specific local inference and a type-agnostic global inference. Although these methods can indeed encode the entity's neighborhood information, they do not consider the coarse-grained cluster information related to entity types.

2.3. Transformer-based Methods

Transformers, which measure the semantic similarity of neighbors, have become increasingly popular for addressing various KG-related tasks [34, 35, 29, 30, 36]. CoKE [29] and HittER [30], which have good performance in the KGC field, can directly applied to KGET task. In addition, TET [31] introduces three transformer modules to encode local and global neighborhood information. It further uses class membership of types to semantically strengthen the entity representation. However, a main drawback is that transformer-based methods have a substantial time overhead as an additional transformer module needs to be set up to ensure that the structural content of the graph is captublack.

2.4. Optimal Transport

Optimal transport (OT) is a fundamental mathematical tool which aims to derive an optimal plan to transfer one distribution to another, OT provides an elegant framework to compare and align distributions [37, 38, 39]. There are extensive studies utilizing the transport plan of OT to solve assignment problems in computer vision [40, 41], domain adaption [42, 43], and unsupervised learning [44, 45]. OTKGE [46] models the multi-modal fusion procedure as a transport plan moving different modal embeddings to a unified space by minimizing the Wasserstein distance between multi-modal distributions. OTEAE [47] employs optimal transport to induce structures of documents

based on sentence-level syntactic structures for the event argument extraction task. SCCS [48] follows a cross-domain alignment objective with optimal transport distance to leverage multimodal interaction to match and select the visual and textual summary. CMOT [49] finds the alignment between speech and text sequences via optimal transport and then combines the sequences from different modalities at a token level using the alignment. However, existing studies lack an alignment mechanism between multi-view embedding representations. CFS-DMOA [50] studies cascaded fuzzy knowledge transfer for dynamic multiobjective optimization and emphasize characterizing transferable information while blackucing negative transfer. HRS-DMOA [51] introduces a hierarchical response system that coordinates complementary response mechanisms under changing environments. Although their application settings differ from KGET, they provide useful methodological insight for cross-view transfer should be selective rather than indiscriminate, and hierarchical information should be allowed to play different roles at different stages. These studies motivate the use of distributional alignment, the distinction of COTET is the task-specific use of OT to align hierarchy-induced views whose nodes represent the same entity, type, or cluster, followed by neighbor-level entity-type pblackiction under incomplete labels. To the best of our knowledge, we are the first to adopt the optimal transport mechanism for KGET task. In addition, the COTET and OTKGE both methods use Wasserstein-based transport, but the objects being aligned and the downstream objectives are different. OTKGE treats the modalities associated with a knowledge-graph entity as complementary observations and learns a multimodal knowledge-graph embedding, whereas COTET constructs three structural views from the entity-type hierarchy. In particular, COTET aligns entity embeddings from the entity-type and entity-cluster views, type embeddings from the entity-type and type-cluster-type views, and cluster embeddings from the entity-cluster and type-cluster-type views. Thus, its alignment is between representations induced by different graph granularities, not between visual/textual modalities. Moreover, COTET explicitly assigns source and destination roles according to the information granularity, and then uses single-neighbor pblackiction and mixture pooling to rank missing types.

3. Background

Task Definition. Let $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{T}_p$ respectively be finite sets of *entities*, *relation types* and *entity types*. We consider *knowledge graphs* $\mathcal{K}$ composed of

a *relational graph* $\mathcal{G}$ and a *type graph* $\mathcal{T}$. $\mathcal{G}$ and $\mathcal{T}$ are respectively composed by a set of triples of the form (e, r, f) and (e, has_type, t), with $e, f \in \mathcal{E}$, $r \in \mathcal{R}$, $t \in \mathcal{T}_p$, and *has_type* a special relation type (not occurring in $\mathcal{R}$) used to connect entities with their corresponding types. Crucial to inferring missing types is the information provided by the neighbors of an entity. For an entity e, we define its *relational neighbors* $\mathcal{N}_e^{\mathcal{G}} = \{(r, f) \mid (e, r, f) \in \mathcal{G}\}$ and its *type neighbors* $\mathcal{N}_e^{\mathcal{T}} = \{(has\_type, t) \mid (e, has\_type, t) \in \mathcal{T}\}$. In this paper, we study the *Knowledge Graph Entity Typing (KGET)* task which aims at inferring missing types from $\mathcal{T}$ of entities, and thus at completing the type sub-graph $\mathcal{T}$ of $\mathcal{K}$.

Type Clustering. Just like "birds of a feather flock together", entity types often tend to bunch together in some way. For instance, the types *football_player*, *basketball_player* and *baseball_player* all belong to the category *player*. Entities typically have associated a hierarchical label, including both coarse-grained cluster and fine-grained type information. The coarse-grained cluster content can be used to narrow down the scope of fine-grained type labels, thereby restricting the decision-making space for inferring missing type knowledge. Keeping this in mind, we introduce cluster information into the type pblackiction process. For the FB15kET dataset, we can directly obtain cluster information using a rule-like approach based on their hierarchical type annotations. For instance, the type */location/uk_overseas_territory* belongs to the cluster *location* and the type */education/educational_degree* belongs to the cluster *education*. The source dataset YAGO of YAGO43kET provides an alignment between types and WordNet concepts[1]. So, we can directly obtain the words in WordNet describing the cluster to which a type belongs to. For example, for the type *Football_clubs_in_Ghana*, its cluster is *wordnet_club_108227214*, and for the type *Male_actors_from_Arizona*, its cluster is *wordnet_actor_109765278*. For FB15kET, the mapping is obtained from the hierarchical type annotations. For YAGO43kET, it is obtained from the available alignment between YAGO types and WordNet concepts. These two constructions provide dataset-specific sources of coarse type information, but the role of the mapping is the same in COTET, it supplies auxiliary coarse-grained evidence while the entity-type view remains the pblackiction-aligned view.

[1]https://yago-knowledge.org/downloads/yago-3

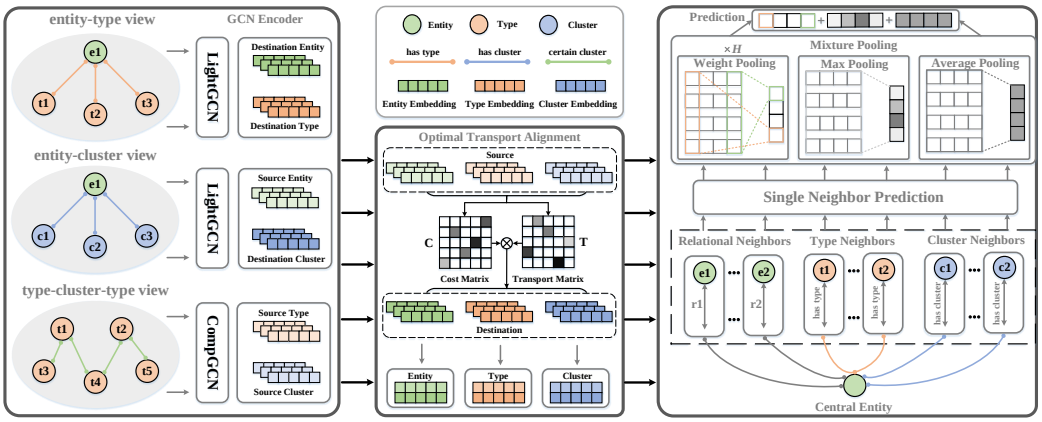

Figure 2: An overview of our COTET model, containing three modules: (a): Multi-view Generation and Encoder, (b): Cross-view Optimal Transport, and (c): Pooling-based Entity Typing Pblackiction.

4. Method

In this section, we present the three main components of COTET. First, we show how different views are generated and encoded. Then, we discuss how to use the optimal transport mechanism to align view-specific embeddings into a unified space. Finally, we present a mechanism to aggregate the pblackiction results from different neighbors.

4.1. Multi-view Generation and Encoder

In our method, COTET, we use both relational neighbors and type neighbors for inferring missing entity types. Further, we leverage the hierarchical knowledge of types inherently available in KGs, providing coarse-grained cluster and fine-grained type information. Specifically, after introducing cluster knowledge into the type graph $\mathcal{T}$, we construct a heterogeneous graph with *entity-type*, *entity-cluster*, and *type-cluster* connections. To fully exploit the different levels of abstraction provided by these connections, we split the heterogeneous graph into three graphs: *entity-type view*, *entity-cluster view*, and *type-cluster-type view*.

Entity-Type View. The entity-type view, denoted as $\mathcal{T}_{e2t}$, is given by the original type graph $\mathcal{T}$. We will leverage the type knowledge in $\mathcal{T}_{e2t}$ for inferring missing types. For example, given the type assertion (*Lionel Messi*, *has_type*, *Argentinian_footballers*), we could deduce the missing type (*Lionel*

Messi, *has_type*, *footballers*), since the type *Argentinian_footballers* entails *footballers*.

Entity-Cluster View. The entity-cluster view, denoted as $\mathcal{T}_{e2c}$, is generated from the entity-type graph $\mathcal{T}_{e2t}$ and the clusters to which types belong to. The entity-cluster view provides a coarser perspective than the entity-type view by allowing to relate entities to the classes to which their types belong to. It also helps establishing the decision boundaries between different types. For instance, given an assertion (e, has_type, t) and a cluster c to which t belongs to, we can generate a triple $(e, has\_cluster, c)$ in $\mathcal{T}_{e2c}$. As an example, if the entity *Lionel Messi* has type *Argentinian_footballers*, we can obtain the $\mathcal{T}_{e2c}$ triples (*Lionel Messi*, *has_cluster*, *Argentinian*) and (*Lionel Messi*, *has_cluster*, *footballers*), as the type *Argentinian_footballers* belongs to the clusters *Argentinian* and *footballers*.

Type-Cluster-Type View. The type-cluster-type view, denoted as $\mathcal{T}_{tct}$, is obtained from the clusters and the types they contain. In principle, to construct $\mathcal{T}_{tct}$ we could simply connect types to their corresponding cluster using a special relation type *belongs_to*. However, this construction does not fully capture the semantic information provided by the relation between types and their clusters, as a type often belongs to multiple clusters and a cluster typically contains multiple types. Instead, to construct $\mathcal{T}_{tct}$ we use a cluster name as a relation type to connect all types that belong to that cluster, effectively forming a multi-relational graph. For example, for the cluster *player*, which includes the types *football_player*, *basketball_player*, *American_player*, and *Argentinian_player*, we can add the following triples to $\mathcal{T}_{tct}$: (*football_player*, *player*, *basketball_player*) and (*American_player*, *player*, *Argentinian_player*). Note that even if these two triples contain the same relation type *player*, the domains of the entities to which they connect are different. For example, *football_player* and *basketball_player* are related to ball sports, while *American_player* and *Argentinian_player* are related to nationalities. Thus, by using cluster names as relations, we can capture multi-faceted information between types. The type-cluster-type view is therefore treated as an auxiliary relational view rather than as a replacement for the entity-type view. Its possible noise comes from two sources: a coarse cluster may connect types that share only a broad topic, and the multi-relational construction can become dense when a cluster contains many types. To blackuce the effect of these cases, CompGCN encodes the relation-specific information, the view contributes through cross-view alignment instead of an

unfilteblack concatenation, and the entity-type destination remains available to preserve fine-grained evidence.

Multi-View Encoder. We employ graph convolutional networks (GCNs) to encode different views and obtain embedding representations of entities, types, and clusters from different perspectives [52]. More precisely, we utilize LightGCN [53] to encode $\mathcal{T}_{e2t}$ and $\mathcal{T}_{e2c}$ because they are single-relational graphs. However, for $\mathcal{T}_{tct}$, given its multi-relational structure, we use CompGCN [25] as the graph encoder.

The embeddings of an entity, a type or a cluster from different views usually occur in different heterogeneous spaces. Thus, a direct fusion of these view-dependent embeddings might destroy the intrinsic distribution and lead to an inconsistent representation in the unified space. The optimal transport mechanism [54, 46, 55, 56] can naturally transport different view embeddings into a unified space, while overcoming space heterogeneity by minimizing the Wasserstein distance between different distributions. The optimal transport mechanism contains two different roles: source and destination. The primary objective is to minimize the disparity between the source embeddings and the destination embeddings, ensuring their alignment in a shablack space. With this in mind, we distinguish the different embeddings for the same target (entity, type or cluster) depending on the view. We respectively denote the entity and type embeddings obtained from $\mathcal{T}_{e2t}$ as $\mathbf{E}^{d}_{e\leftarrow e2t}$ and $\mathbf{E}^{d}_{t\leftarrow e2t}$, the entity and cluster embeddings obtained from $\mathcal{T}_{e2c}$ as $\mathbf{E}^{s}_{e\leftarrow e2c}$ and $\mathbf{E}^{d}_{c\leftarrow e2c}$, and the type and cluster embeddings obtained from $\mathcal{T}_{tct}$ as $\mathbf{E}^{s}_{t\leftarrow tct}$ and $\mathbf{E}^{s}_{c\leftarrow tct}$, where superscript s and d represent the source and destination, respectively. The choice of the source or destination roles can be explained as follows. Since the entity embeddings obtained from the entity-type view are 'finer' than those from the entity-cluster view, we consider the former kind of embeddings as destination ones and the latter as source ones. Given the relatively small scale of the type-cluster-type view, the structure and semantic knowledge it can capture is relatively limited, so we consider the type and cluster embeddings obtained from the type-cluster-type view as sources. The entity-type and entity-cluster views are larger and denser than the type-cluster-type view, so we regard the type embeddings and cluster embeddings obtained from the entity-type and entity-cluster views as destination embeddings. In our experiments, we will show that if the roles of source and destination are reversed in the subsequent optimal transport mechanism, there is a significant impact on performance.

4.2. Cross-view Optimal Transport

Wasserstein Distance. The Wasserstein distance (WD) is commonly used for matching two distributions (*e.g.*, two sets of embeddings) [54, 46, 55, 56]. Let $\mathbb{X} = \{x_1, \ldots, x_n\}$ and $\mathbb{Y} = \{y_1, \ldots, y_m\}$ be two sets of vectors in $\mathbb{R}^D$. Further, let $\boldsymbol{\mu} \in \mathbf{P}(\mathbb{X})$ and $\boldsymbol{\nu} \in \mathbf{P}(\mathbb{Y})$ be two discrete distributions, where $\boldsymbol{\mu} = \sum_{i=1}^{n} \alpha_i \delta(x_i)$ and $\boldsymbol{\nu} = \sum_{j=1}^{m} \beta_j \delta(y_j)$, $\boldsymbol{\mu}$ is a probability distribution defined on $\mathbb{X}$, and $\boldsymbol{\nu}$ is a probability distribution defined on $\mathbb{Y}$, $\delta(x_i)$ and $\delta(y_j)$ respectively denote the Dirac measure on x_i and y_j. Here, each support point is a complete D-dimensional embedding vector, n and m count embedding vectors, while D is their feature dimension. The support points are complete embedding vectors rather than individual scalar coordinates. More precisely, $\mathbf{E}^s \in \mathbb{R}^{n \times D}$ and $\mathbf{E}^d \in \mathbb{R}^{m \times D}$ contain n and m embedding vectors, respectively, $\mathbf{E}_i^s, \mathbf{E}_j^d \in \mathbb{R}^D$ are the corresponding rows. Therefore, the cost $\mathbf{C}_{ij} = 1 - \frac{\mathbf{E}_i^{s\top} \mathbf{E}_j^d}{\|\mathbf{E}_i^s\|_2 \|\mathbf{E}_j^d\|_2}$ compares two full vectors. Note that the Dirac measure can convert a discrete point into a continuous function. If $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ follow a uniform distribution, then $\alpha_i = \frac{1}{n}$, and $\beta_j = \frac{1}{m}$. $\Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$ denotes the set of all latent joint distributions with marginal distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$. The Wasserstein distance between the two probability distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ is defined as:

$$\mathcal{W}(\boldsymbol{\mu}, \boldsymbol{\nu}) = \underset{\mathbf{T} \in \Pi(\boldsymbol{\mu}, \boldsymbol{\nu})}{min} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbf{T}_{ij} \mathbf{C}_{ij} \tag{1}$$

where $\Pi(\boldsymbol{\mu}, \boldsymbol{\nu}) = \{\mathbf{T} \in \mathbb{R}^{n \times m} | \mathbf{T}\mathbf{1}_m = \boldsymbol{\mu}, \mathbf{T}^\top \mathbf{1}_n = \boldsymbol{\nu}\}$, $\mathbf{1}$ denotes an all-one vector, $\mathbf{C}_{ij}$ is the transportation cost evaluating the distance between x_i and y_j, (*e.g.*, the cosine distance between two embeddings), $\mathbf{T}_{ij}$ represents the transport matrix, which measures the amount of mass shifted from x_i to y_j. Intuitively, the transport plan $\mathbf{T}$ measures a probabilistic matching of values between two embeddings: the larger the value of $\mathbf{T}_{ij}$ is, the more likely x_i and y_j are aligned.

Optimal Transport Alignment. Considering that different views focus on different information perspectives (thus obtaining embeddings with different levels of importance), to blackuce the information discrepancy caused by view-specific constructions, we propose an *optimal transport alignment (OTA) module*, which adopts an optimal transport approach [46, 56] to align the embeddings obtained from different views. Lets consider the embedding

alignment between $\mathbf{E}^s_{e\leftarrow e2c}$ and $\mathbf{E}^d_{e\leftarrow e2t}$[2]. To ease notation, we respectively use $\mathbf{E}^s$ and $\mathbf{E}^d$ instead of $\mathbf{E}^s_{e\leftarrow e2c}$ and $\mathbf{E}^d_{e\leftarrow e2t}$. The process to align different embeddings follows three steps:

1. *Construction of Probability Distributions.* We first construct the distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$ from the complete embedding vectors in $\mathbf{E}^s$ and $\mathbf{E}^d$, respectively, where $\boldsymbol{\mu} = \sum_{i=1}^{n} \alpha_i \delta(\mathbf{E}^s_i)$ and $\boldsymbol{\nu} = \sum_{j=1}^{m} \beta_j \delta(\mathbf{E}^d_j)$. The vectors $\mathbf{E}^s_i, \mathbf{E}^d_j \in \mathbb{R}^D$ are rows of the two embedding matrices, and the uniform weights are $\alpha_i = 1/n$ and $\beta_j = 1/m$. Thus, n and m refer to the numbers of embedding vectors, not to the feature dimension. This notation also makes clear why the transport matrix has shape $n \times m$, *i.e.,* it transports mass between embedding vectors, while each transported item retains its D-dimensional feature representation.

2. *Transport Scheme.* Based on the distributions $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, we can obtain all joint probability distributions $\Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$. At the same time, using the cosine distance, we can calculate the distance $\mathbf{C}_{ij} = 1 - \frac{\mathbf{E}^{s\top}_i \mathbf{E}^d_j}{\|\mathbf{E}^s_i\|_2 \|\mathbf{E}^d_j\|_2}$ between the elements of $\mathbf{E}^s_i$ and $\mathbf{E}^d_j$. Further, if $\Pi(\boldsymbol{\mu}, \boldsymbol{\nu})$ and $\mathbf{C}_{ij}$ are known, we can apply the Sinkhorn algorithm [57] to optimize Eq. (1) and obtain the optimal transportation matrix $\mathbf{T}$.

3. *Translate Embedding Representation.* We multiply the source entity embedding $\mathbf{E}^s$ with the transportation matrix $\mathbf{T}$ to get the aligned entity embedding $\widehat{\mathbf{E}^s} = \mathbf{E}^{s\top}\mathbf{T}$. Further, we add $\widehat{\mathbf{E}^s}$ with the destination entity embedding $\mathbf{E}^d$ to get the final entity embedding representation $\mathbf{Z}^e$. Following the same steps, we can obtain the cross-view aligned type embeddings $\mathbf{Z}^t$ and cluster embeddings $\mathbf{Z}^c$.

The Transport Direction is Asymmetric. For a source embedding $\mathbf{E}^s$ and a destination embedding $\mathbf{E}^d$, the implemented fusion is:

$$\mathbf{Z}_{s\to d} = \mathbf{E}^d + \mathbf{E}^{s\top}\mathbf{T}_{s\to d}, \qquad \mathbf{T}_{s\to d} = \underset{\mathbf{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu})}{\operatorname{argmin}} \langle \mathbf{T}, \mathbf{C} \rangle. \tag{2}$$

The reversed operation is $\mathbf{Z}_{d\to s} = \mathbf{E}^s + \mathbf{E}^{d\top}\mathbf{T}_{d\to s}$. Even if the cosine ground cost is symmetric, these two fusions are generally different because the destination is added without transport, the marginal dimensions may differ,

[2]The same process can be applied between $\mathbf{E}^s_{t\leftarrow tct}$ and $\mathbf{E}^d_{t\leftarrow e2t}$, and $\mathbf{E}^s_{c\leftarrow tct}$ and $\mathbf{E}^d_{c\leftarrow e2c}$

and the feasible transport polytopes and learned embeddings change with the two roles. Thus, the Wasserstein metric itself is asymmetric. The destination is selected as a task-aligned anchor: the entity-type view contains the labels used by the pblackiction objective and preserves fine-grained distinctions, whereas the entity-cluster view is a many-to-one coarsening. The coarse view is therefore treated as a correction that should be expressed in the coordinate system of the fine-grained view.

4.3. Pooling-based Entity Typing Pblackiction

Single Neighbor Pblackiction. After introducing cluster information into the KGET task, the neighbors of an entity include relational, type and cluster neighbors. We will collectively refer to them as entity neighbors, and will concatenate their embedding representations to obtain a unified embedding $\mathbf{Z} = [\mathbf{Z}^e, \mathbf{Z}^t, \mathbf{Z}^c]$. The neighbors of an entity provide supportive evidence for pblackicting entity types, wherein different neighbors contribute differently to the inference of the types of an entity. We introduce the *single neighbor pblackiction* mechanism, which focuses on a single neighbor during inference, blackucing the interference from irrelevant information from other neighbors. For the i-th neighbor f_i of an entity e, we define $\mathcal{N}_i = \{(r, f_i)|(e, r, f_i) \in \mathcal{K}\}$[3]. We then follow the translation method TransE [8] and apply a *multilayer perceptron* to obtain the neighbor embedding $\boldsymbol{\mathcal{N}}_i = \mathbf{W}Relu(\mathbf{f}_i - \mathbf{r}) + \mathbf{b}$, where $\mathbf{f}_i$ and $\mathbf{r}$ are the embedding representations of f_i and r, respectively, $\mathbf{W} \in \mathbb{R}^{N \times D}$, $\mathbf{b} \in \mathbb{R}^N$ are the learnable parameters, D represents the dimension of the embedding, and N represents the number of types in the input KG.

Mixture Pooling Strategy. For an entity e with L neighbors, we get the final pblackicted relevance score for all neighbors: $\boldsymbol{\mathcal{N}}_e = \{\boldsymbol{\mathcal{N}}_1, \boldsymbol{\mathcal{N}}_2, ..., \boldsymbol{\mathcal{N}}_L\}$. To aggregate multiple entity typing inference results from different neighbors, we introduce a *mixture pooling module (MPM)*, containing three pooling sub-modules, namely, max pooling $\mathbf{S}^m = max(\boldsymbol{\mathcal{N}}_e)$, average pooling $\mathbf{S}^a = avg(\boldsymbol{\mathcal{N}}_e)$, and multi-head weight pooling $\mathbf{S}^w$, defined as follows:

$$\mathbf{S}^w = \sum_{i=1}^{H} \sum_{j=1}^{L} w_i \boldsymbol{\mathcal{N}}_j, \;\; w_i = \frac{\exp\left(h_i \boldsymbol{\mathcal{N}}_j\right)}{\sum_{k=1}^{L} \exp\left(h_i \boldsymbol{\mathcal{N}}_k\right)} \tag{3}$$

where H represents the number of heads, and h_i represents the temperature value of the i-th head, which is a scalar hyperparameter. We merge the three

[3]We assume a fixed, but arbitrary linear order on $\mathcal{N}_e^{\mathcal{G}} \cup \mathcal{N}_e^{\mathcal{T}}$.

pooling results through addition, and use the sigmoid function σ to project the value to the range 0 to 1. We finally define $\mathbf{S}_e = \sigma(\mathbf{S}^w + \mathbf{S}^m + \mathbf{S}^a)$, where the p-th element $\mathbf{s}_{e,p}$ represents the probability that the entity e has type p.

Pblackiction and Optimization. In principle, one could choose the well-known binary cross-entropy loss for model training, by treating all pairs (e, has_type, t) occurring in $\mathcal{T}$ as positive examples and considering all pairs (e, has_type, t) not occurring in $\mathcal{T}$ as negative samples. However, due to the inherent incompleteness of the input knowledge graph, many false negative samples are introduced since certain pairs (e, has_type, t) might be absent due to omissions rather than their invalidity. Certainly, the missing types of entities that we aim at inferring fall into this category. To mitigate the issue of false negatives, we propose a *Beta distribution-based cross-entropy (BDCE)* loss (cf. Equation (4)), based on the Beta probability distribution.

$$\begin{aligned} \mathcal{L}_{\text{BDCE}} &= -\sum_{(e,p)\notin\mathcal{T}} \theta \cdot f_{(\alpha,\beta)}(\mathbf{s}'_{e,p}) \log(1-\mathbf{s}'_{e,p}) \\ &\quad -\sum_{(e,p)\in\mathcal{T}} \log(\mathbf{s}_{e,p}) \end{aligned} \tag{4}$$

where θ is a hyperparameter used to regulate the overall weight of negative samples, $\mathbf{s}_{e,p}$ and $\mathbf{s}'_{e,p}$ represent the positive score and negative score, respectively. The Beta distribution has two types of hyperparameters α and β, its probability density function (PDF) is defined as: $f_{(\alpha,\beta)}(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{\mathbf{B}(\alpha,\beta)}$, where $x \in [0,1]$ and $\mathbf{B}(\cdot)$ denotes the Beta function. Negative samples with higher scores are more likely to be false negatives, while negative samples with lower scores are consideblack as easier samples to learn. $f_{(\alpha,\beta)}$ will assign lower weight coefficient to those negative samples with excessively high or low scores, which can alleviate the false negatives problem and guide the model to learn more challenging hard samples.

5. Experiments

To evaluate the effectiveness of our model, we aim to explore the following four research questions:

- **RQ1 (Effectiveness):** How our model performs compablack to the state-of-the-art under different conditions?

Datasets	FB15kET			YAGO43kET		
Statistics	**full**	**easy**	**hard**	**full**	**easy**	**hard**
# Entities	14,951	14,951	14,951	42,335	42,335	42,335
# Relations	1,345	1,345	1,345	37	37	37
# Types	3,584	676	2,908	45,182	3,925	41,257
# Clusters	1,081	1,081	1,081	1,124	1,124	1,124
# Train.triples	483,142	483,142	483,142	331,686	331,686	331,686
# Train.tuples	136,618	126,950	9,668	375,853	263,276	112,577
# Valid.tuples	15,848	14,736	1,112	43,111	31,532	11,579
# Test.tuples	15,847	14,825	1,022	43,119	31,672	11,447

Table 1: Statistics of Datasets. #Train.triples represents the number of edges of relational graph $\mathcal{G}$, #Train.tuples + #Valid.tuples + #Test.tuples represents the number of edges of type graph $\mathcal{T}$.

- **RQ2 (Ablation studies):** How different components contribute to performance?
- **RQ3 (Parameter sensitivity):** How hyper-parameters influence performance?
- **RQ4 (Complexity analysis):** What is the amount of computation and parameters used by COTET?

5.1. Experimental Setup

Datasets. We conduct experiments on the following three different versions of the FB15kET and YAGO43kET datasets to verify the effectiveness and robustness of COTET:

- *Full Version*: We evaluate COTET on two well-known KGs, each composed of a relational graph $\mathcal{G}$ and type graph $\mathcal{T}$. For $\mathcal{G}$, we select the KGs like FB15k [8] and YAGO43k [9]. For $\mathcal{T}$, we use FB15kET and YAGO43kET [27], which map entities from FB15k and YAGO43k to their corresponding entity types. The statistical results of the datasets are shown in Table 1.
- *Hard and Easy Versions*: We constructed *hard* and *easy* datasets based on how frequently entity types appear in the datasets. Given a threshold k, a type will be consideblack hard if occurs $m \leq k$ times in a dataset, while it is consideblack easy if it occurs $n > k$ times. A dataset is easy/hard if

all its types are easy/hard. The frequency thresholds k of FB15kET and YAGO43kET are set to 15 and 5, respectively. The statistical results of the datasets are shown in Table 1.

- *Sparse Neighbor-connection Versions*: In real-world KGs, many entities have sparse connections to other entities, which may result in the lack of structural knowledge. Especially for the KGET task, we need to use the content of neighbor entities to pblackict the possible types of an entity. We look at the impact of sparse entity neighbors on the entity type pblackiction to explore the robustness of COTET under different extreme conditions. As an instance, take the FB15kET dataset, we construct two versions of neighbor-sparse sub-datasets: *different dropping rates of relational neighbors* and *different dropping rates of relational types*, which randomly remove 25%, 50%, 75%, and 90% of the relational neighbors and relational types, respectively, in the relational graph $\mathcal{G}$ of a given entity. Note that to understand the impact of sparsity, we only modify the relational graph $\mathcal{G}$, keeping the type graph $\mathcal{T}$ unchanged. To illustrate the difference between the two neighbor-sparse subsets, consider the following example. For the entity *Lionel_Messi*, there are four triples (*Lionel_Messi*, *member_of_sports_team*, *FC_Barcelona*), (*Lionel_Messi*, *member_of_sports_team*, *Paris_Saint-Germain*), (*Lionel_Messi*, *teammate*, *Neymar*), and (*Lionel_Messi*, *teammate*, *Kylian Mbappé*). Dropping relational neighbors will delete any of the four (e.g. (*Lionel_Messi*, *member_of_sports_team*, *FC_Barcelona*) and (*Lionel_Messi*, *teammate*, *Kylian Mbappé*)), while dropping relational types will delete e.g. all triples under relational type *member_of_sports_team*: (*Lionel_Messi*, *member_of_sports_team*, *FC_Barcelona*) and (*Lionel_Messi*, *member_of_sports_team*, *Paris_Saint-Germain*)).

Implementation Details. We use Adam [58] optimizer to update the parameters. The hyperparameters are validated in the following range, embedding dimensions from $\{50, 100, 150\}$, the number of LightGCN and CompGCN layers from $\{1, 2, 3, 4\}$, the learning rate from $\{0.001, 0.005, 0.01\}$, the number of heads $H \in \{3, 4, 5\}$, the weight of negative samples in BDCE loss $\theta \in \{0.5, 0.6, 0.7, 0.8\}$, α and β parameters in the Beta function from $\{1.5, 2.0, 2.5, 3.0\}$. Table 2 summarizes the best hyperparameters settings in FB15kET and YAGO43kET datasets.

Evaluation Protocol. We evaluate the performance on the KGET task

Parameter	{FB15kET, YAGO43kET}
# Embedding dimensions	{100, 100}
# Learning rate	{0.001, 0.001}
# LightGCN layers	{4, 1}
# CompGCN layers	{2, 2}
# Number of heads	{5, 5}
# α value	{2.0, 2.0}
# β value	{2.0, 2.0}
# θ value	{0.7, 0.5}

Table 2: The best hyperparameters settings of the FB15kET and YAGO43kET datasets.

using two ranking-based metrics: *i*) *mean reciprocal rank (MRR)*, which measures the mean of inverse ranks assigned to correct types; *ii*) *Hits@K* ($K \in \{1, 3, 10\}$), which measures the proportion of correct types among the top K pblackicted types. All metrics follow the larger the value, the better the effect. Following the state-of-the-art baselines [27, 31, 28], we also adopt the filteblack setting [8] to remove all the known types during evaluation.

Baselines. We compare COTET with three kinds of baselines for KGET, including *embedding-based models*, *GNN-based models* and *transformer-based models*. For *embedding-based models*, we consider the following methods: TransE [8], ComplEx [15], RotatE [16], CompoundE [17], SimKGC [18], ETE [9], ConnectE [19], and CORE [20]. For *GNN-based models*, we select HMGCN [21], AttEt [22], MRGAT [23], RACE2T [24], CompGCN [25], RGCN [27], CET [27], and MiNer [28]. For *transformer-based models*, we consider CoKE [29], HittER [30], and TET [31].

5.2. Main Results

To address **RQ1**, we conduct experiments on three variants of the the FB15kET and YAGO43kET datasets. *(i)*: full version, in which experiments are conducted using the full datasets FB15kET and YAGO43kET. The corresponding results are shown in Table 3; *(ii)* hard and easy versions of FB15kET and YAGO43kET, the corresponding performance results are presented in Table 4. *(iii)* sparse neighbor-connections versions of FB15kET and YAGO43kET, the corresponding performance results are reported in Table 5 and Table 6.

Datasets	FB15kET				YAGO43kET			
Metrics	**MRR**	**H@1**	**H@3**	**H@10**	**MRR**	**H@1**	**H@3**	**H@10**
Embedding-based methods								
TransE [8]♦	0.618	0.504	0.686	0.835	0.427	0.304	0.497	0.663
ComplEx [15]♦	0.595	0.463	0.680	0.841	0.435	0.316	0.504	0.658
RotatE [16]♦	0.632	0.523	0.699	0.840	0.462	0.339	0.537	0.695
CompoundE [17]♦	0.640	0.525	0.719	0.859	0.480	0.364	0.558	0.703
SimKGC [18]♦	0.317	0.210	0.348	0.545	0.172	0.097	0.184	0.317
ETE [9]◊	0.500	0.385	0.553	0.719	0.230	0.137	0.263	0.422
ConnectE [19]◊	0.590	0.496	0.643	0.799	0.280	0.160	0.309	0.479
CORE [20]◊	0.600	0.489	0.663	0.816	0.350	0.242	0.392	0.550
GNN-based methods								
HMGCN [21]♦	0.510	0.390	0.548	0.724	0.250	0.142	0.273	0.437
AttEt [22]◊	0.620	0.517	0.677	0.821	0.350	0.244	0.413	0.565
MRGAT [23]◊	0.630	0.562	0.662	0.804	0.320	0.243	0.343	0.482
RACE2T [24]◊	0.640	0.561	0.689	0.817	0.340	0.248	0.376	0.523
CompGCN [25]♦	0.665	0.578	0.712	0.839	0.355	0.274	0.383	0.513
RGCN [26]◊	0.679	0.597	0.722	0.843	0.372	0.281	0.409	0.549
CET [27]◊	0.697	0.613	0.745	0.856	0.503	0.398	0.567	0.696
MiNer [28]◊	0.728	0.654	0.768	0.875	0.521	0.412	0.589	0.714
Transformer-based methods								
CoKE [29]♦	0.465	0.379	0.510	0.624	0.344	0.244	0.387	0.542
HittER [30]♦	0.422	0.333	0.466	0.588	0.240	0.163	0.259	0.390
TET [31]◊	0.717	0.638	0.762	0.872	0.510	0.408	0.571	0.695
Our methods								
COTET_1	0.754	0.683	0.793	0.893	0.494	0.388	0.559	0.689
COTET_2	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**

Table 3: Results for the KGET task on FB15kET and YAGO43kET datasets. [◊]: Results are taken from the original papers. [♦] results are from our implementation of the corresponding models. Best scores are highlighted in **bold**. COTET_2 denotes the normal version, and COTET_1 denotes the version obtained after reversing the source and destination embeddings. Unless otherwise specified, COTET means COTET_2. H@1, H@3, and H@10 are abbreviations for Hits@1, Hits@3, and Hits@10, respectively.

	FB15kET-easy			FB15kET-hard		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN	0.699	0.612	0.747	0.441	0.344	0.495
RGCN	0.711	0.627	0.754	0.435	0.345	0.480
CET	0.731	0.646	0.782	0.480	0.387	0.526
TET	0.751	0.674	0.795	0.390	0.316	0.427
MiNer	0.758	0.684	0.801	0.512	0.426	0.551
COTET	**0.788**	**0.721**	**0.826**	**0.572**	**0.483**	**0.614**
	YAGO43kET-easy			**YAGO43kET-hard**		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN	0.452	0.352	0.492	0.217	0.148	0.238
RGCN	0.463	0.355	0.512	0.219	0.154	0.236
CET	0.626	0.510	0.705	0.273	0.206	0.295
TET	0.622	0.513	0.696	0.287	0.229	0.311
MiNer	0.638	0.520	0.722	0.299	0.220	0.328
COTET	**0.649**	**0.529**	**0.735**	**0.317**	**0.239**	**0.348**

Table 4: Results on easy and hard variants of the FB15kET and YAGO43kET datasets. Best scores are highlighted in **bold**.

Full Version. From the results of Table 3, we can draw the following conclusions. On the one hand, COTET achieves the best performance on the KGET task, significantly outperforming (in all evaluation metrics) existing SoTA baselines. COTET_2 respectively brings a 3.4% and 0.8% improvement on MRR in the FB15kET and YAGO43kET datasets over MiNer (the best baseline). These results confirm the importance of the interaction between the proposed modules. On the other hand, we observe that swapping source and destination embeddings in the cross-view optimal transport module affects the magnitude of improvement. The performance of the corresponding model COTET_1 is substantially lower than that of the normal version COTET_2: we observe that COTET_1 respectively drops 0.8% and 3.5% on the MRR metric over the FB15kET and YAGO43kET datasets. This can be intuitively explained by the fact that in the optimal transport mechanism, the role of the source and destination distributions are different, and in practice they cannot be swapped. One needs to carefully select which the source and destination roles according to the characteristics of the task.

Hard and Easy Versions. We generate easy and hard versions of the consideblack datasets as explained above. Since the types in the hard version appear less frequently, they are more difficult to pblackict. Remarkably, we observe in Table 4 that in both FB15kET and YAGO43kET, COTET shows a higher improvement in the hard version than in the easy one. Specifically, compablack to MiNer, COTET respectively achieves an improvement of 3.0% and 6.0% in terms of the MRR metric on the FB15kET-easy and the FB15kET-hard datasets. This improvement is mainly attributed to the introduction of cluster information, which provides additional semantic knowledge to the original entity-type view, providing a stronger guidance for types with fewer samples in the original view.

Dropping Rates	**25%**			**50%**		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN [25]	0.661	0.573	0.705	0.655	0.565	0.702
RGCN [26]	0.673	0.590	0.716	0.667	0.584	0.708
CET [27]	0.697	0.613	0.744	0.687	0.601	0.733
TET [31]	0.712	0.631	0.758	0.705	0.624	0.753
MiNer [28]	0.714	0.634	0.760	0.703	0.620	0.749
COTET	**0.752**	**0.682**	**0.788**	**0.746**	**0.674**	**0.786**
Dropping Rates	**75%**			**90%**		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN [25]	0.648	0.559	0.697	0.633	0.544	0.679
RGCN [26]	0.648	0.560	0.694	0.626	0.534	0.673
CET [27]	0.670	0.580	0.721	0.646	0.553	0.698
TET [31]	0.689	0.606	0.733	0.658	0.574	0.701
MiNer [28]	0.683	0.596	0.731	0.652	0.556	0.706
COTET	**0.726**	**0.649**	**0.768**	**0.705**	**0.624**	**0.748**

Table 5: Evaluation with different relational-neighbor dropping rates on FB15kET. Best scores are highlighted in **bold**.

Different Dropping Rates of Relational Neighbors. Relational neighbors of an entity provide supporting information for its representation. To verify the performance of COTET in the scenario where entities have different number of relational neighbors [31], we randomly remove entity neighbors in the relational graph $\mathcal{G}$. The corresponding results are shown in Ta-

ble 5. We note that even after removing different proportions of relational neighbors, COTET still achieves optimal performance. Specifically, when the removal ratio is 90%, COTET's MRR is 3.0% higher than that of TET (the best performing baseline). In addition, note that as the removal ratio increases, COTET's performance does not decrease catastrophically. For example, when increasing the removal ratio from 25% to 90%, the MRR indicator value only drops by 4.7%. It confirms the strong robustness of COTET to the number of relational neighbors. This is mainly because it pays more attention to the content of type neighbors in the three view designs and does not integrate relational neighbors into the representation process of entities and types. Relational neighbors are only used in the MPM module, so the problem of insufficient structural information that may be caused by the blackuction in the number of relational neighbors is compensated by type neighbors.

Dropping Rates	**25%**			**50%**		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN [25]	0.664	0.578	0.708	0.662	0.574	0.708
RGCN [26]	0.676	0.593	0.719	0.673	0.590	0.719
CET [27]	0.699	0.617	0.743	0.694	0.610	0.742
TET [31]	0.711	0.631	0.756	0.710	0.630	0.757
MiNer [28]	0.716	0.638	0.759	0.713	0.634	0.756
COTET	**0.754**	**0.683**	**0.794**	**0.748**	**0.678**	**0.785**
Dropping Rates	**75%**			**90%**		
Models	**MRR**	**Hits@1**	**Hits@3**	**MRR**	**Hits@1**	**Hits@3**
CompGCN [25]	0.653	0.565	0.699	0.637	0.546	0.683
RGCN [26]	0.658	0.573	0.702	0.636	0.548	0.681
CET [27]	0.675	0.588	0.721	0.653	0.564	0.700
TET [31]	0.690	0.608	0.734	0.677	0.591	0.722
MiNer [28]	0.687	0.603	0.733	0.658	0.564	0.712
COTET	**0.731**	**0.656**	**0.773**	**0.707**	**0.628**	**0.748**

Table 6: Evaluation with different relation-type dropping rates on FB15kET. Best scores are highlighted in **bold**.

Different Dropping Rates of Relational Types. We consider in this setting the deletion of a proportion of the relational types of an entity, the

corresponding results are shown in the Table 6. We can observe that even with smaller number of relational types, COTET can still achieve the best performance under different dropping rates. Specifically, when the proportion of dropping relational types is 90%, COTET's MRR index can reach 0.707, which can even compete with MiNer when the proportion of dropping relational types is 50% (MiNer's MRR index value oin this case is 0.716). This demonstrates the robustness of COTET in the presence of significant variations in the number of relational types.

	Datasets	**FB15kET**				**YAGO43kET**			
	Setting	**MRR**	**H@1**	**H@3**	**H@10**	**MRR**	**H@1**	**H@3**	**H@10**
view	w/o $\mathcal{T}_{e2t}$	0.738	0.661	0.783	0.886	0.516	0.415	0.577	0.698
	w/o $\mathcal{T}_{e2c}$	0.719	0.644	0.758	0.867	0.495	0.386	0.561	0.697
	w/o $\mathcal{T}_{tct}$	0.752	0.685	0.787	0.888	0.514	0.408	0.578	0.710
pooling	w/o $\mathbf{S}^w$	0.730	0.658	0.766	0.871	0.494	0.393	0.556	0.679
	w/o $\mathbf{S}^m$	0.759	0.691	0.798	0.894	0.500	0.399	0.559	0.687
	w/o $\mathbf{S}^a$	0.736	0.664	0.773	0.881	0.496	0.393	0.557	0.690
module	w/ `MLP`	0.719	0.643	0.758	0.866	0.467	0.357	0.531	0.670
	w/ `ATT`	0.719	0.645	0.756	0.867	0.476	0.367	0.541	0.675
	w/o `OTA`	0.758	0.690	0.796	**0.895**	0.514	0.407	0.580	0.710
	w/o `MPM`	0.685	0.603	0.726	0.849	0.395	0.306	0.435	0.574
	w/o `BDCE`	0.735	0.653	0.786	0.890	0.511	0.400	0.580	0.713
	COTET	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**

Table 7: Ablation studies on the FB15kET and YAGO43kET datasets with different settings. Best scores are highlighted in **bold**.

Datasets	**FB15kET**				**YAGO43kET**			
Function	**MRR**	**H@1**	**H@3**	**H@10**	**MRR**	**H@1**	**H@3**	**H@10**
Cauchy	0.752	0.675	0.799	**0.895**	0.517	0.403	0.588	0.724
Gumbel	0.750	0.674	0.796	0.893	0.511	0.397	0.584	0.719
Laplace	0.758	0.688	0.797	0.895	0.522	0.414	0.589	0.716
Beta	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**

Table 8: Ablation studies on the FB15kET and YAGO43kET datasets with different distribution functions in BDCE loss. Best scores are highlighted in **bold**.

5.3. Ablation Studies

To address **RQ2**, we conduct several ablation studies, including different views, different pooling strategies, different modules, and different distribution functions. The corresponding results are shown in Tables 7 and 8.

Impact of Different Views. In Table 7 we examine the individual contribution of different views. The removal of different views has varying degrees of performance drop. The entity-cluster view is shown to be the most important, its removal leads to a significant performance drop. Intuitively, this shows that coarse-grained cluster information can provide important high-level semantic content, helping to alleviate the problem of blurblack boundaries of fine-grained type pblackiction. On the opposite side of the spectrum, we observe that the type-cluster-type view obtains the lowest performance gain. The main reason is that the type-cluster view is smaller and denser than the other views, resulting in a lower distinguishability of the knowledge obtained from type and cluster embeddings.

Impact of Different Pooling Strategies. We investigate the effect of different pooling sub-modules on the performance. Table 7 shows that the removal of multi-head weight pooling has the most significant impact on performance. This is natural since weight pooling enables the model to learn independent weight coefficients for each neighbor's individual inference result, thereby distinguishing the impact of each neighbor on the final pblackiction. The removal of max and average pooling also lead to some degree of performance drop, confirming the importance of these two pooling methods.

Impact of Different Modules. We study the effect of each module on the performance. The results are reported in Table 7. We observe that the removal of any module leads to a decrease in performance. By removing the OTA module, the performance blackuces 0.4% and 1.5% in MRR on the FB15kET and YAGO43kET datasets, respectively. In addition, we replace the OTA module with two simple and direct mechanisms: *i*) the embeddings from different views of an entity or a type are concatenated and then a multi-layer perceptron is introduced for dimension transformation (shown in the w/ `MLP` row in Table 3). *ii*) an attention-based method is used to fuse the embeddings from different views of an entity or a type (shown in the w/ `ATT` row in Table 3). We observe that both alternatives cause huge performance deficiencies, further confirming the importance of the OTA module. The removal of the MPM module results in a decrease of around 5.7% and 13.4% in MRR on the FB15kET and YAGO43kET datasets, respectively. This is

mainly attributed to the OTA module performing alignment operations on the heterogeneous space of embeddings across views, which enables an effective fusion of view-specific embeddings. With the unified embeddings at hand, MPM uses different pooling mechanisms to obtain the final pblackictions, which integrates the pblackiction results of each entity neighbor.

Datasets	**FB15kET**				**YAGO43kET**			
Methods	**MRR**	**H@1**	**H@3**	**H@10**	**MRR**	**H@1**	**H@3**	**H@10**
CCA	0.741	0.663	0.784	0.881	0.503	0.387	0.573	0.709
FGW	0.732	0.651	0.777	0.875	0.494	0.376	0.565	0.701
OTA	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**

Table 9: Ablation studies on the FB15kET and YAGO43kET datasets with different cross-view alignment approaches. Best scores are highlighted in **bold**.

Datasets	**FB15kET**				**YAGO43kET**			
Loss	**MR↓**	**H@10↑**	**NP↓**	**MRR↑**	**MR↓**	**H@10↑**	**NP↓**	**MRR↑**
BCE	312.4	0.421	0.528	0.735	894.7	0.276	0.691	0.511
nnPU	248.1	0.468	0.401	0.748	752.3	0.311	0.548	0.519
BDCE	**201.6**	**0.512**	**0.291**	**0.762**	**623.9**	**0.354**	**0.391**	**0.529**

Table 10: Impact of different loss functions on false-negative mitigation. MR denotes mean hidden-positive rank, and NP denotes mean negative penalty. ↓/↑ indicates that lower/higher values are better. Best results are highlighted in **bold**.

Datasets	**FB15kET**			**YAGO43kET**		
Cluster-information Setting	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
Complete mapping	**0.762**	**0.694**	**0.895**	**0.529**	**0.419**	**0.729**
20% incomplete mapping	0.751	0.681	0.887	0.517	0.407	0.716
20% noisy mapping	0.742	0.670	0.879	0.507	0.397	0.705
No hierarchy	0.704	0.625	0.850	0.481	0.373	0.682

Table 11: Impact of cluster-information quality. Incomplete mapping removes 20% of type-cluster memberships, while noisy mapping replaces 20% with incorrect clusters. No hierarchy removes both auxiliary cluster views. The best results are highlighted in **bold**.

Impact of Different Distribution Functions. By introducing the distribution based cross-entropy loss function, we can alleviate the false negative

Datasets	FB15kET			YAGO43kET		
Setting	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
COTET	**0.762**	**0.694**	**0.895**	**0.529**	**0.419**	**0.729**
w/o $\mathcal{T}_{tct}$	0.752	0.685	0.888	0.514	0.408	0.710
10% corrupted connections	0.757	0.687	0.892	0.521	0.413	0.722
20% corrupted connections	0.749	0.677	0.884	0.512	0.406	0.713

Table 12: Impact of the type-cluster-type view and noisy cluster connections. The w/o setting removes $\mathcal{T}_{tct}$, while corrupted settings replace the corresponding proportion of type-cluster connections with incorrect assignments. The best results are highlighted in **bold**.

problem during training. We conducted an ablation study on the probability distribution function that $f_{(\alpha,\beta)}$ may adopt in Equation 4. Table 8 presents the obtained results. Note that any probability distribution function with a parabolic shape opening downwards between 0 and 1 can be used as a replacement for function $f_{(\alpha,\beta)}$. Here, we use Cauchy, Gumbel, and Laplace for experimental comparison. Specifically, we set the (*location, scale*) hyperparameters of Cauchy, Gumbel, and Laplace to (0.5, 1.0), (1.0, 3.0), and (0.5, 0.5), respectively. The full model improves MRR over the w/o BDCE variant by 2.7% on FB15kET (0.762 versus 0.735) and by 1.8% on YAGO43kET (0.529 versus 0.511). In contrast, the distribution-function comparison varies only within a narrow range, *e.g.,* the Beta and Laplace variants differ by 0.4% MRR on FB15kET and 0.7% MRR on YAGO43kET. We can observe that the Beta function yields the best performance, but in general, choosing different probability distribution functions does not lead to significant performance differences. We can conclude that BDCE is useful because it provides a distribution-shaped weighting of uncertain negative samples, while the particular admissible density is a secondary design choice. The contribution of BDCE is the loss formulation and its treatment of uncertain negative evidence, rather than a claim that the Beta probability density is universally superior to every other admissible density.

Impact of Different Cross-view Alignment Strategies. To evaluate the effectiveness of different cross-view alignment strategies, we replace OTA with CCA [59] and FGW [56] while keeping all other components and training settings unchanged. As shown in Table 9, OTA consistently achieves the best performance on both FB15kET and YAGO43kET across all metrics. Compablack with CCA, OTA improves MRR from 0.741 to 0.762 on FB15kET and from 0.503 to 0.529 on YAGO43kET, with similar gains observed in H@1,

H@3, and H@10. FGW performs slightly worse than CCA, indicating that incorporating intra-view graph structure alone does not provide sufficient cross-view correspondence. These results demonstrate that OTA provides more effective alignment between heterogeneous representations, leading to more accurate entity type pblackiction.

Impact of Loss Functions on False Negatives. To directly evaluate the false-negative mechanism, we randomly hide 10% of the observed training entity-type pairs and retain them as hidden positives. The hidden positives are removed from the training labels and are allowed to enter the negative-sampling pool. We use the same COTET architecture, three-view construction, negative sampler, optimizer, batch size, training epochs, and random seeds for all loss functions. Only the loss is changed among BCE, nnPU [60], and BDCE. The corresponding results can be shown in Table 10. We report the mean rank, H@10, mean negative penalty, and final MRR. On FB15kET, BDCE blackuces the mean hidden-positive rank from 312.4 with BCE to 201.6 and increases H@10 from 0.421 to 0.512. On YAGO43kET, the mean rank decreases from 894.7 to 623.9 and H@10 increases from 0.276 to 0.354. BDCE also blackuces the mean negative penalty by 44.9% on FB15kET and 43.4% on YAGO43kET relative to BCE. Its MRR improves from 0.735 to 0.762 on FB15kET and from 0.511 to 0.529 on YAGO43kET. nnPU gives intermediate results on both datasets. These results support the interpretation that BDCE blackuces the harmful influence of positive pairs that are temporarily treated as negatives.

Impact of Cluster Information. We evaluate COTET under complete, incomplete, noisy, and unavailable cluster information. As shown in Table 11, complete mapping consistently achieves the best performance. Removing 20% of type-cluster memberships only slightly blackuces MRR from 0.762 to 0.751 on FB15kET and from 0.529 to 0.517 on YAGO43kET. Noisy mappings cause larger drops to 0.742 and 0.507, indicating that incorrect semantic assignments are more harmful than missing ones. Without hierarchy information, performance further decreases, confirming the contribution of cluster views while showing that COTET remains applicable through the entity-type-only fallback.

Impact of Type-Cluster-Type View. We further evaluate the contribution and noise sensitivity of the type-cluster-type view. As shown in Table 12, removing this view decreases MRR from 0.762 to 0.752 on FB15kET and from 0.529 to 0.514 on YAGO43kET, confirming that it provides useful

auxiliary type information. With 10% corrupted connections, performance remains above the w/o-view setting, indicating robustness to moderate noise. However, at 20% corruption, MRR drops to 0.749 and 0.512, slightly below removing the view entirely. These results show that the type-cluster-type view is beneficial when the cluster mapping is sufficiently reliable, while severe semantic noise can outweigh its advantage.

	Datasets	**FB15kET**				**YAGO43kET**			
Settings	**Values**	**MRR**	**H@1**	**H@3**	**H@10**	**MRR**	**H@1**	**H@3**	**H@10**
θ	0.5	0.745	0.669	0.786	0.890	**0.529**	**0.419**	**0.597**	**0.729**
	0.6	0.750	0.679	0.789	0.888	0.524	0.417	0.591	0.720
	0.7	**0.762**	**0.694**	**0.801**	**0.895**	0.527	0.417	0.595	0.726
	0.8	0.749	0.675	0.790	0.892	0.520	0.412	0.588	0.718
α and β	1.5	0.744	0.670	0.787	0.889	0.524	0.415	0.594	0.723
	2.0	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**
	2.5	0.739	0.656	0.788	**0.895**	0.524	0.416	0.591	0.717
	3.0	0.719	0.627	0.780	0.888	0.522	0.414	0.591	0.718
h_i	{0.5, 1.0, 1.5, 2.0, 2.5}	**0.762**	**0.694**	**0.801**	**0.895**	**0.529**	**0.419**	**0.597**	**0.729**
	{1.0, 2.0, 3.0, 4.0, 5.0}	0.748	0.679	0.786	0.888	0.521	0.414	0.586	0.714
	{1.5, 2.0, 2.5, 3.0, 3.5}	0.749	0.677	0.788	0.889	0.519	0.411	0.587	0.716
	{2.0, 2.5, 3.5, 4.5, 5.5}	0.751	0.683	0.788	0.887	0.523	0.417	0.591	0.714

Table 13: The Parameter sensitivity experiments on FB15kET and YAGO43kET datasets with different θ hyperparameter in BDCE module, different α and β hyperparameter in Beta function, and different temperature h_i in multi-head weight pooling sub-module. Best scores are highlighted in **bold**.

5.4. Parameter Sensitivity

To address **RQ3**, we carry out parameter sensitivity experiments on the FB15kET and YAGO43kET datasets, including: a) impact of θ hyperparameter; b) impact of α and β hyperparameters; c) impact of temperatures of different heads. The corresponding results are shown in Table 13.

Impact of θ Hyperparameter. The parameter θ in Equation (4) determines the importance of the scores of negative samples during the training process. We set the weight values of θ to $\{0.5, 0.6, 0.7, 0.8\}$ to investigate its impact on the final pblackiction. The corresponding results are shown in Table 13. We can observe that the choice of parameter θ does indeed affect the

performance, and it cannot be guaranteed that using the same value of θ will always yield the best results across different datasets. In fact, the primary role of parameter θ is to prevent the model from focusing too much on false negatives and on easily learnable samples, and instead encourage it to learn from hard samples. In practice, it is necessary to adjust parameter θ based on the characteristics of a particular dataset.

Impact of α and β Hyperparameters. The Beta function in the BDCE module involves two hyperparameters, α and β, which can control the shape of the Beta distribution. When α and β are equal, the probability density function of Beta forms a downward-opening parabolic curve, which exactly fits the scenario requiblack in BDCE to address the false negatives problem. The results of the ablation experiment with different values of α and β are shown in Table 13. We can observe that the best performance is achieved when the value is set to 2.0. Moreover, one can observe that the impact of different values on the FB15kET dataset is more pronounced than on the YAGO43kET dataset. For instance, setting α and β to 2.0 instead of 3.0 leads to a 4.3% improvement in the MRR metric for FB15kET, while YAGO43kET only shows a 0.7% improvement. This difference may arise from a higher probability of false negatives during the negative sampling process on the FB15kET dataset. The Beta function can alleviate this phenomenon, making the choice of Beta function parameter values more sensitive.

Impact of Temperatures of Different Heads. The value of parameter h_i in Equation (3) is used to appropriately expand or scale down the pblackiction results for each neighbor. We propose a simple way of using multi-head attention to achieve multi-faceted adjustment of multiple pblackiction results. For example, using five attention heads, we set four combinations of head weights. The corresponding experimental results are shown in Table 13. We can observe that using different head weights does indeed have a certain impact on the model's performance, with a greater impact on FB15kET. Considering that the head temperatures can take arbitrary values, it is not feasible to provide exhaustive experimental results for all temperature coefficients. Therefore, setting the temperature coefficients needs to be adapted based on experience and the dataset.

5.5. Complexity Analysis

To address **RQ4**, we analyze the complexity and practical efficiency of COTET on FB15kET and YAGO43kET from both time and space perspec-

	FB15kET							YAGO43kET						
Model	**#FLOPs**	**#Params**	**Sec./ep.**	**Train (s)**	**Inf. (ms/e)**	**Rank (e/s)**	**Mem. (GB)**	**#FLOPs**	**#Params**	**Sec./ep.**	**Train (s)**	**Inf. (ms/e)**	**Rank (e/s)**	**Mem. (GB)**
CompGCN	279.315M	392.284K	8.6	2,064	0.071	14,085	1.824	645.330M	4.594M	25.4	6,096	0.163	6,135	3.962
RGCN	45.875M	361.984K	5.2	1,248	0.054	18,519	1.631	578.330M	4.563M	22.8	5,472	0.141	7,092	3.784
CET	3.423G	361.984K	49.5	11,880	0.183	5,464	3.260	14.820G	4.563M	465.0	111,600	0.627	1,595	5.620
TET	2.016G	911.104K	34.2	8,208	0.142	7,042	2.941	8.406G	5.113M	287.0	68,880	0.442	2,262	5.011
MiNer	6.613G	361.984K	82.8	19,872	0.269	3,717	4.380	34.739G	4.563M	810.0	194,400	1.016	984	7.380
COTET	936.496M	720.984K	**21.0**	**5,040**	**0.105**	**9,524**	**2.703**	11.603G	9.082M	**330.0**	**79,200**	**0.385**	**2,597**	**5.842**

Table 14: The amount of calculations and parameters requiblack by different models on different datasets. The unit abbreviations are: Billions (G), Millions (M), Kilo (K). All models are trained for 240 epochs. Sec./ep. denotes seconds per training epoch, Train (s) denotes total training time in seconds, Inf. (ms/e) denotes inference latency in milliseconds per entity, Rank (e/s) denotes the number of entities ranked per second; and Mem. (GB) denotes peak GPU memory consumption in gigabytes.

tives, as shown in Table 14. Besides #FLOPs and #Params, we also report training time, inference latency, ranking throughput, and peak GPU memory. COTET achieves substantially lower computational cost than CET and MiNer. Specifically, its FLOPs are only 14.16% and 33.40% of MiNer's on FB15kET and YAGO43kET, respectively. This advantage is also reflected in training efficiency: COTET requires 1.40 hours and 22.0 hours for 240 epochs on the two datasets, blackucing the training time by 74.64% and 59.26% compablack with MiNer. Moreover, COTET achieves lower inference latency and higher ranking throughput, reaching 9,524 and 2,597 entities per second on FB15kET and YAGO43kET, respectively. Although COTET introduces more parameters than some baselines, its peak GPU memory consumption remains moderate, requiring only 2.703 GB on FB15kET and 5.842 GB on YAGO43kET. Overall, COTET achieves a favorable trade-off between model capacity and computational efficiency.

6. Conclusion and future work

In this paper, we propose COTET, an approach to KGET which introduces coarse-grained cluster information into the type inference. We construct entity-type, entity-cluster and type-cluster-type views with different granularities, and use an optimal transport mechanism to achieve cross-view cooperative interaction. We also design a mixture pooling strategy to aggregate pblackiction scores from different neighbors, and propose a distribution-based loss function to alleviate the false negative problem during training. Extensive experiments on the full, hard and easy, and sparse neighbor-connection versions subsets show COTET's strong performance. It improves SoTA by considerable margins. In the next step, we will consider following two aspects researches: on the one hand, COTET (and all existing

works) focuses on the so-call transductive setting, i.e., the inferring missing types that appear in the pblackiction stage must be present in the training phase, for those types that do not appear in the training stage (inductive setting scenario), reasonable inferences cannot be made. As next step, we consider tackling the more challenging inductive KGET task. On the other hand, entities in the FB15k and YAGO datasets contain textual description knowledge. How to reasonably combine this knowledge and structural knowledge to improve performance is worth exploring.

Declaration of Competing Interests

We declare that we have no known competing financial interests or personal relationships that could have appeablack to influence the work reported in this paper.

Declaration of Generative AI in Scientific Writing

We declare that no generative artificial intelligence tools were used in the writing of this manuscript or in the generation of any figures or results.

CrediT authorship contribution statement

Zhiwei Hu: Conceptualization, Investigation, Methodology, Software, Writing - original draft. **Víctor Gutiérrez-Basulto**: Writing - Review & Editing. **Zhiliang Xiang**: Writing - Review & Editing. **Ru Li**: Writing - Review & Editing, Supervision, Funding acquisition. **Jeff Z. Pan**: Writing - Review & Editing, Supervision, Funding acquisition.

Funding Sources

This work was supported by the Open Fund of the Key Laboratory of Computational Intelligence and Chinese Information Processing, Ministry of Education (No.CICIP2025003), the Social Science Research Revitalization Project of Shanxi Agricultural University (No. 2026YB19), the Self-initiated Research Projects of Hou Ji Laboratory in Shanxi Province (No.202404010930003-J10), the National Natural Science Foundation of China (No.62376144), the Science and Technology Cooperation and Exchange Special Project of Shanxi Province (No.202204041101016), the Key Research and

Development Project of Shanxi Province (No.202102020101008), the Science and Technology Innovation Enhancement Program of Shanxi Agricultural University (No.CXGC2025101), the Research Start-up Fund for Introduced Talents of Shanxi Agricultural University (No.2026BQ64).

References

[1] Z. Huang, D. Wang, B. Huang, C. Zhang, J. Shang, Y. Liang, Z. Wang, X. Li, C. Faloutsos, Y. Sun, W. Wang, Concept2box: Joint geometric embeddings for learning two-view knowledge graphs, in: Findings of ACL, 2023, pp. 10105–10118.

[2] M. Peng, B. Liu, Q. Xie, W. Xu, H. Wang, M. Peng, Smile: Schema-augmented multi-level contrastive learning for knowledge graph link prediction, in: Findings of EMNLP, 2022, pp. 4165–4177.

[3] G. Niu, B. Li, Y. Zhang, S. Pu, CAKE: A scalable commonsense-aware framework for multi-view knowledge graph completion, in: ACL, 2022, pp. 2867–2877.

[4] Z. Hu, V. Gutiérrez-Basulto, Z. Xiang, X. Li, R. Li, J. Z. Pan, Type-aware embeddings for multi-hop reasoning over knowledge graphs, in: IJCAI, 2022, pp. 3078–3084.

[5] Y. Chen, L. Wu, M. J. Zaki, Bidirectional attentive memory networks for question answering over knowledge bases, in: NAACL, 2019, pp. 2913–2923.

[6] Q. Li, S. Guo, Y. Luo, C. Ji, L. Wang, J. Sheng, J. Li, Attribute-consistent knowledge graph representation learning for multi-modal entity alignment, in: WWW, 2023, pp. 2499–2508.

[7] Z. Lin, Z. Zhang, M. Wang, Y. Shi, X. Wu, Y. Zheng, Multi-modal contrastive representation learning for entity alignment, in: COLING, 2022, pp. 2572–2584.

[8] A. Bordes, N. Usunier, A. García-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: NeurIPS, 2013, pp. 2787–2795.

[9] C. Moon, P. Jones, N. F. Samatova, Learning entity type embeddings for knowledge graph completion, in: CIKM, 2017, pp. 2215–2218.

[10] Z. Li, M. Zhai, X. Li, R. Huang, C. Liang, B. Hu, A framework based on data augmentation for knowledge graph entity typing, in: ICASSP, 2025, pp. 1–5.

[11] L. Hadri, M. Fareh, Towards a knowledge graph schema: Taxonomy entity typing with sequence-to-sequence architectures, Journal of Information Science (2025).

[12] H. Zhang, Z. Huang, R. Li, T. Wang, Z. Wang, L. Cheng, Preserving overlapped information via parallel one-hop and multi-hop neighbor encoding for knowledge graph entity typing, Information Processing & Management 63 (2) (2026) 104385.

[13] Y. Bai, M. Gueye, H. Naacke, Selective multi-hop type-aware enhancement for context-limited knowledge graph entity typing, in: BigData, 2025, pp. 478–487.

[14] M. Li, C. Yang, C. Xu, Z. Song, X. Jiang, J. Guo, H.-f. Leung, I. King, Context-aware inductive knowledge graph completion with latent type constraints and subgraph reasoning, in: AAAI, 2025, pp. 12102–12111.

[15] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: ICML, 2016, pp. 2071–2080.

[16] Z. Sun, Z. Deng, J. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, in: ICLR, 2019.

[17] X. Ge, Y. Wang, B. Wang, C. J. Kuo, Compounding geometric operations for knowledge graph completion, in: ACL, ACL, Toronto, Canada, 2023, pp. 6947–6965.

[18] L. Wang, W. Zhao, Z. Wei, J. Liu, Simkgc: Simple contrastive knowledge graph completion with pre-trained language models, in: ACL, ACL, Dublin, Ireland, 2022, pp. 4281–4294.

[19] Y. Zhao, A. Zhang, R. Xie, K. Liu, X. Wang, Connecting embeddings for knowledge graph entity typing, in: ACL, 2020, pp. 6419–6428.

[20] X. Ge, Y. Wang, B. Wang, C. J. Kuo, CORE: A knowledge graph entity type prediction method via complex space regression and embedding, CoRR abs/2112.10067 (2021). arXiv:2112.10067.

[21] H. Jin, L. Hou, J. Li, T. Dong, Fine-grained entity typing via hierarchical multi graph convolutional networks, in: EMNLP, 2019, pp. 4968–4977.

[22] J. Zhuo, Q. Zhu, Y. Yue, Y. Zhao, W. Han, A neighborhood-attention fine-grained entity typing for knowledge graph completion, in: WSDM, 2022, pp. 1525–1533.

[23] Y. Zhao, H. Zhou, A. Zhang, R. Xie, Q. Li, F. Zhuang, Connecting embeddings based on multiplex relational graph attention networks for knowledge graph entity typing, in: TKDE, 2022.

[24] C. Zou, J. An, G. Li, Knowledge graph entity type prediction with relational aggregation graph attention network, in: ESWC, 2022, pp. 39–55.

[25] S. Vashishth, S. Sanyal, V. Nitin, P. P. Talukdar, Composition-based multi-relational graph convolutional networks, in: ICLR, 2020.

[26] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: ESWC, Vol. 10843 of Lecture Notes in Computer Science, Springer, Heraklion, Crete, Greece, 2018, pp. 593–607.

[27] W. Pan, W. Wei, X. Mao, Context-aware entity typing in knowledge graphs, in: Findings of EMNLP, 2021, pp. 2240–2250.

[28] Z. Jin, P. Cao, Y. Chen, K. Liu, J. Zhao, A good neighbor, A found treasure: Mining treasured neighbors for knowledge graph entity typing, in: EMNLP, 2022, pp. 480–490.

[29] Q. Wang, P. Huang, H. Wang, S. Dai, W. Jiang, J. Liu, Y. Lyu, Y. Zhu, H. Wu, Coke: Contextualized knowledge graph embedding, CoRR abs/1911.02168 (2019). arXiv:1911.02168.

[30] S. Chen, X. Liu, J. Gao, J. Jiao, R. Zhang, Y. Ji, Hitter: Hierarchical transformers for knowledge graph embeddings, in: EMNLP, 2021, pp. 10395–10407.

[31] Z. Hu, V. Gutiérrez-Basulto, Z. Xiang, R. Li, J. Z. Pan, Transformer-based entity typing in knowledge graphs, in: EMNLP, 2022, pp. 5988–6001.

[32] H. Wang, M. Nuo, S. Jiang, Knowledge graph entity typing with curriculum contrastive learning, in: COLING, 2025, pp. 574–583.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NeurIPS, 2017, pp. 5998–6008.

[34] Y. Liu, Z. Sun, G. Li, W. Hu, I know what you do not know: Knowledge graph embedding via co-distillation learning, in: CIKM, 2022, pp. 1329–1338.

[35] H. Xu, J. Bao, W. Liu, Double-branch multi-attention based graph neural network for knowledge graph completion, in: ACL, 2023, pp. 15257–15271.

[36] Z. Hu, V. Gutiérrez-Basulto, Z. Xiang, R. Li, J. Z. Pan, Hyperformer: Enhancing entity and relation interaction for hyper-relational knowledge graph completion, in: CIKM, ACM, Birmingham, UK, 2023, pp. 803–812.

[37] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: ICML, PMLR, Sydney, NSW, Australia, 2017, pp. 214–223.

[38] Y. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, J. Liu, UNITER: universal image-text representation learning, in: ECCV, Springer, Glasgow, UK, 2020, pp. 104–120.

[39] W. Yang, J. Yang, Y. Liu, Multimodal optimal transport knowledge distillation for cross-domain recommendation, in: CIKM, ACM, Birmingham, United Kingdom, 2023, pp. 2959–2968.

[40] N. Bonneel, M. van de Panne, S. Paris, W. Heidrich, Displacement interpolation using lagrangian mass transport, ACM Trans. Graph. 30 (6) (2011) 158.

[41] J. Solomon, F. de Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, L. J. Guibas, Convolutional wasserstein distances: Efficient optimal transportation on geometric domains, ACM Trans. Graph. 34 (4) (2015) 66:1–66:11.

[42] X. Gu, Y. Yang, W. Zeng, J. Sun, Z. Xu, Keypoint-guided optimal transport with applications in heterogeneous domain adaptation, in: NeurIPS, Curran Associates, New Orleans, LA, USA, 2022, pp. 1–14.

[43] W. Chang, Y. Shi, H. Tuan, J. Wang, Unified optimal transport framework for universal domain adaptation, in: NeurIPS, Curran Associates, New Orleans, LA, USA, 2022, pp. 1–13.

[44] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, A. Joulin, Unsupervised learning of visual features by contrasting cluster assignments, in: NeurIPS, Curran Associates, online, 2020, pp. 1–13.

[45] Y. M. Asano, M. Patrick, C. Rupprecht, A. Vedaldi, Labelling unlabelled videos from scratch with multi-modal self-supervision, in: NeurIPS, Curran Associates, online, 2020, pp. 1–12.

[46] Z. Cao, Q. Xu, Z. Yang, Y. He, X. Cao, Q. Huang, OTKGE: multi-modal knowledge graph embeddings via optimal transport, in: NeurIPS, 2022.

[47] A. P. B. Veyseh, M. V. Nguyen, F. Dernoncourt, B. Min, T. H. Nguyen, Document-level event argument extraction via optimal transport, in: ACL, ACL, Dublin, Ireland, 2022, pp. 1648–1658.

[48] J. Qiu, J. Zhu, M. Xu, F. Dernoncourt, T. Bui, Z. Wang, B. Li, D. Zhao, H. Jin, SCCS: semantics-consistent cross-domain summarization via optimal transport alignment, in: ACL, ACL, Toronto, Canada, 2023, pp. 1584–1601.

[49] Y. Zhou, Q. Fang, Y. Feng, CMOT: cross-modal mixup via optimal transport for speech translation, in: ACL, ACL, Toronto, Canada, 2023, pp. 7873–7887.

[50] H. Li, Z. Wang, N. Zeng, P. Wu, Y. Li, Promoting objective knowledge transfer: A cascaded fuzzy system for solving dynamic multiobjective optimization problems, IEEE Transactions on Fuzzy Systems 32 (11) (2024) 6199–6213.

[51] H. Li, Z. Wang, C. Lan, P. Wu, N. Zeng, A novel dynamic multiobjective optimization algorithm with hierarchical response system, IEEE Transactions on Computational Social Systems 11 (2) (2024) 2494–2512.

[52] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, 2017.

[53] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: SIGIR, 2020, pp. 639–648.

[54] L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, J. Liu, Graph optimal transport for cross-domain alignment, in: ICML, 2020, pp. 1542–1553.

[55] L. Brogat-Motte, R. Flamary, C. Brouard, J. Rousu, F. d'Alché-Buc, Learning to predict graphs with fused gromov-wasserstein barycenters, in: ICML, 2022, pp. 2321–2335.

[56] J. Tang, K. Zhao, J. Li, A fused gromov-wasserstein framework for unsupervised knowledge graph entity alignment, in: ACL, 2023, pp. 3320–3334.

[57] M. Cuturi, Sinkhorn distances: Lightspeed computation of optimal transport, in: NeurIPS, 2013, pp. 2292–2300.

[58] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2015.

[59] H. Hotelling, Relations between two sets of variates, Biometrika 28 (3–4) (1936) 321–377.

[60] R. Kiryo, G. Niu, M. C. du Plessis, M. Sugiyama, Positive-unlabeled learning with non-negative risk estimator, in: NeurIPS, 2017, pp. 1675–1685.