Unifying Structure and Language Semantic for Efficient Contrastive Knowledge Graph Completion with Structured Entity Anchors

Anonymous ACL submission

Abstract

The goal of knowledge graph completion (KGC) is to predict missing links in a KG using trained facts that are already known. In recent, pre-trained language model (PLM) based methods that utilize both textual and structural information are emerging, but their performances lag behind state-of-the-art (SOTA) structure-based methods or some methods lose their inductive inference capabilities in the process of fusing structure embedding to text encoder. In this paper, we propose a novel method to effectively unify structure information and language semantics without losing the power of inductive reasoning. We adopt entity anchors and these anchors and textual description of KG elements are fed together into the PLM-based encoder to learn unified representations. In addition, the proposed method utilizes additional random negative samples which can be reused in the each mini-batch during contrastive learning to learn a generalized entity representations. We verify the effectiveness of the out proposed method through various experiments and analysis. The experimental results on standard benchmark widely used in link prediction task show that the proposed model outperforms existing the SOTA KGC models. Especially, our method show the largest performance improvement on FB15K-237, which is competitive to the SOTA of structure-based KGC methods.

1 Introduction

Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), and WordNet (Miller, 1995) are popular large-scale knowledge graphs (KGs) that contain human knowledge information as form of directed graph structure. In general, KGs are represented as a set of triple facts. A `fact` (or `triple`) consists of two entities and a relation, which is usually expressed in the form of `(head entity, relation, tail entity)`, which is also denoted as `(h, r, t)`. The most of the KGs are inherently incomplete, therefore, KGC techniques are used to infer the missing links from the known facts of KGs to enrich KGs.

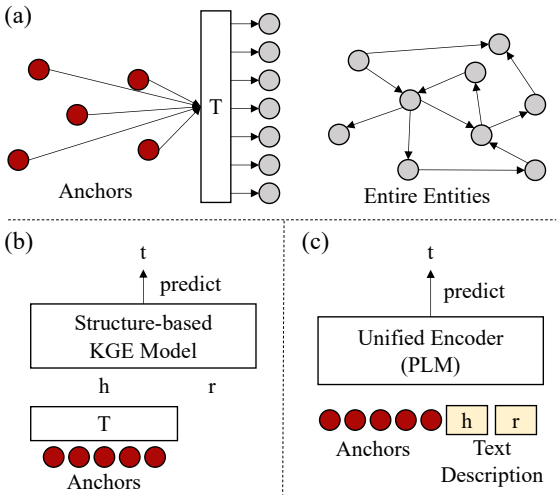

Figure 1: (a) Transformation from the fixed sized anchors to full entity embeddings. (b) The anchors are trained via structure-based KGE method. (c) The anchors and descriptions of h, r are fed into unified model to predict tail. Since all entities are encoded by their description and the fixed anchors, the model can represent unseen entities.

There are two main streams of KGC methods: structure based and textual description based methods. Traditional structure based methods use only the structure information of KG to learn the representation of entities and relations. These methods, such as (Bordes et al., 2013; Yang et al., 2014; Trouillon et al., 2016; Sun et al., 2019; Balažević et al., 2019), map each entity and relation into a low-dimensional vector. On the other hand, with development of PLM and large language model (LLM), textual description based KGC methods are emerging (Yao et al., 2019; Wang et al., 2021b; Choi et al., 2021; Wang et al., 2022a; Pan et al., 2023). These kind of methods express entities and relations by encoding their textual information. Many researches propose various methods

(Wang et al., 2021a; Shen et al., 2022; Wang et al., 2022b; Chen et al., 2023) to deal with the lack of structure information, but some methods (Wang et al., 2022b; Chen et al., 2023), in the process of using structure information, the inductive reasoning ability is lost which is the key advantage of textual description models.

In this paper, we propose a novel method that effectively integrates structural and textual information without losing inductive reasoning capabilities. Inspired by the anchor & transformation mechanism (Liang et al., 2020; Galkin et al., 2021), we design our model to utilize the decomposition of entity embeddings to represent every entity with a small number of anchors. These entity anchors are trained through the structure based KGC method and they are reused as input to the entity encoder, allowing them to be effectively fused with the language semantics. Since the entity anchors can be used exactly the same for unseen entities, they do not impede the model to making inductive inferences.

In addition, we find a limitation of in-batch contrastive learning in our experiments. SimKGC (Wang et al., 2022a) enables efficient contrastive learning through in-batch negative sampling, and our proposed model also follows the approach. However, we argue that in-batch negative sampling may introduce a bias that the negative samples follow the training data distribution. We address this issue by adding uniform random negative sampling strategy and we can take remarkable performance gains.

We evaluate the proposed method on FB15K-237, WN18RR, and Wikidata5M, the most widely used benchmark datasets. The proposed method outperforms SOTA text-based and structure-based methods on FB15K-237 and Wikidata5M. The result of the proposed method are comparable to the performance of SOTA on WN18RR. Moreover, we conduct various analysis to identify the contribution of each component of the model.

2 Related Work

2.1 Structure Based Graph Embedding Methods

Structure-based methods, such as (Bordes et al., 2013; Wang et al., 2014; Ji et al., 2015; Lin et al., 2015), aim to fit triples in a knowledge graph by translating the head entity's vector $\mathtt{h}$ to the tail entity's vector $\mathtt{t}$ through the relation vector $\mathtt{r}$. DistMult (Yang et al., 2014) employs a streamlined bilinear interaction process that incorporates relations as diagonal elements within relation-specific diagonal matrices. ComplEx (Trouillon et al., 2016) utilizes complex-valued vectors to map entities and relations, capturing intricate relational patterns through bi-linear interactions between entity vectors. RotatE (Sun et al., 2019) efficiently infers these patterns by mapping entities and relations to a complex vector space and framing relations as rotations from source to target entities.

2.2 Textual Description Based Graph Embedding Methods

KG-BERT (Yao et al., 2019) considers entities, relations, and triples as textual sequences, effectively reframing knowledge graph completion as a sequence classification challenge. MEM-KGC (Choi et al., 2021) leverages the principles of a masked language model to enhance the KGC task. StAR (Wang et al., 2021a) introduces a hybrid model that combines textual encoding and graph embedding paradigms to harness the synergistic advantages of both. While KGC methods using text descriptions have been criticized for having high computational costs with lower performance, SimKGC (Wang et al., 2022a) addresses this problem with efficient contrastive learning. It adopts a bi-encoder structure for training with InfoNCE (Oord et al., 2018) loss and incorporates three types of negatives.

2.3 Vocabulary Reduction

In the context of KG representation, conventional approaches involve assigning a unique embedding vector to each entity, leading to linear memory consumption growth and high computational costs when dealing with real-world KGs. Anchor & Transform (ANT) (Liang et al., 2020) learns a small set of anchor embeddings and a sparse transformation matrix. Notably, ANT represents the embeddings of entities as sparse linear combinations of these anchor embeddings, weighted according to the values within the transformation matrix. NodePiece (Galkin et al., 2021) is an anchor-based method that learns a fixed-size entity vocabulary, similar to subword tokenization in natural language processing (NLP). By constructing a vocabulary of subword/sub-entity units from anchor nodes in a graph with known relation types, NodePiece facilitates encoding and embedding for any entity, including previously unseen entities during training.

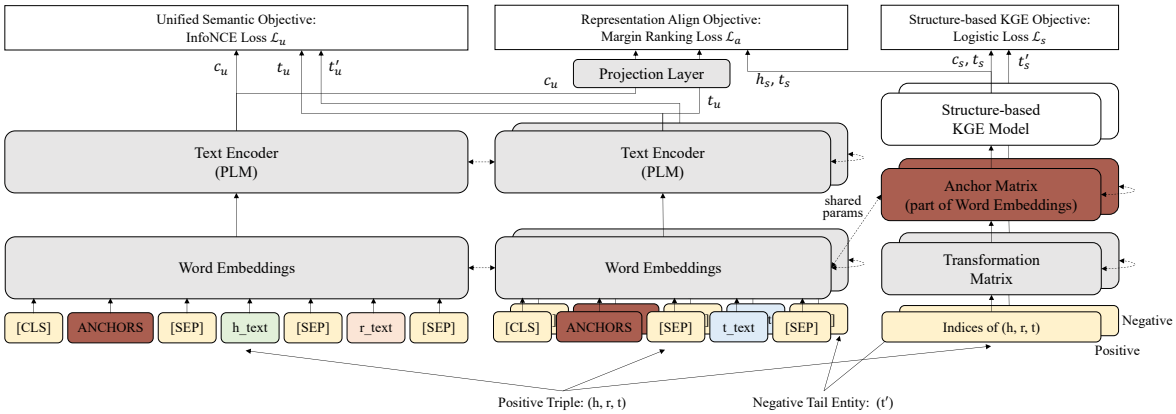

Figure 2: An overview of the SEA-KGC model architecture.

3 Method

3.1 Model Architecture

The proposed method, SEA-KGC, consists of two main part. The first part has PLM layer. It uses both entity anchors and textual descriptions of triple elements as input to learn a unified representation. The other part uses only entity anchors to learn the structure information of triples. This part has entity anchor matrix, transformation matrix and KGE scoring model. We only use the first part that has PLM to get unified representations for predicting entities. In the following sections, we describe in detail the training and inference of the proposed model.

3.1.1 Efficient Contrastive learning for Unified Representation

The first module utilizes the both of structure entity anchors and textual description of triple elements to generate unified representations. Here we use BERT (Devlin et al., 2018) as the PLM layer. All entities and relations are described into their name or description text. Each structure entity anchor is treated as special token which is part of word embeddings.

Given a triple, we split it into two parts `(h, r)` and `(t)` and encode them separately. This approach is more efficient than encoding the entire triple at once, as shown in several previous methods (Wang et al., 2021a, 2022a). To obtain the representation of head entity and relation, we make an input sequence by concatenating the entity anchors, text of head entity and text of relation with `'[SEP]'` as a separator. We take the hidden states of the last layer of the PLM. We apply mean pooling to get the context embedding c_u.

The tail entity representation t_u is created by encoding the anchors and text of tail with a separator. We use a single PLM that shares the same parameters for encoding head entity, relation and tail entity. The matching score between embedding c_u and t_u is defined by cosine similarity.

$$cos(c_u, t_u) = \frac{c_u \cdot t_u}{||c_u||\ ||t_u||} \quad (1)$$

Through the optimization, we make the score between the positive pair c_u, t_u high and the negative pair c_u, t'_u low.

3.1.2 Structure based Knowledge Embedding Learning

We perform additional optimization to ensure that entity anchors capture more structure information of KG. We adopt anchor & transformation mechanism (Liang et al., 2020) to represent every structure entity embedding as a linear combination of the entity anchors. The entire entity anchors are shared with each entity and each entity maintains an entity specific transformation weight vector. We optimize the structure entity embeddings using the structure based KGE method. Here, we use TransE (Bordes et al., 2013) scoring function. We can compute the triple score using the following formula with structure embedding h_s, r_s, and t_s.

$$KGE(h, r, t) = -||h_s + r_s - t_s|| \quad (2)$$

$$h_s = T[h_i] \cdot A, r_s = R[r_i], t_s = T[t_i] \cdot A \quad (3)$$

3.2 Initializing Structured Entity Anchors

Before training the SEA-KGC model, we should initialize structure entity anchors. There are two initializing methods. The first method is to initialize entity anchors randomly. The second method

is to initialize the entity anchors with a clustering technique. In the second method, we first represent each entity embedding as a vector encoded from its text description by BERT. We cluster all entities into a target number of anchors, set the centroid of each cluster as an anchor. We use K-Means algorithm to find centroids. We can reproduce all entity embeddings E by matrix multiplication of the transformation and anchor matrix, T and A. Therefore, initial transformation matrix can be calculated by multiplying the inverse of entity anchor matrix to entire embedding matrix. It can be formalized as,

$$E = TA \tag{4}$$

$$T = EA^{-1} \tag{5}$$

where $E \in \mathbb{R}^{V \times D}$, $T \in \mathbb{R}^{V \times N}$, and $A \in \mathbb{R}^{N \times D}$. V, N, and D are size of entity vocabulary, number of anchors, and size of hidden dimension, respectively.

3.3 Negative Sampling

We basically follow in-batch contrastive learning to train our model. This learning approach reuses the samples in the mini-batch as negative samples for each other. However, we find a problem with in-batch negative sampling during our experiments: the in-batch negatives follow the entity occurrence distribution in the training dataset (i.e., entities that occur more frequently in training are used as negatives more often). This problem may be an obstacle to learning a better representation, as some triples are repeatedly given only trivial negatives. To address this issue, we add randomness to negatives. We keep the in-batch negative samples $\mathcal{N}_{IB}$, and sample uniform random entities from the set of train entities and use them as additional negatives $\mathcal{N}_{UR}$. Thus, the set of negatives $\mathcal{N}(h, r)$ is,

$$\mathcal{N}(h, r) = \{t' | t' \in \mathcal{N}_{IB} \cup \mathcal{N}_{UR}\}. \tag{6}$$

Additional uniform random negative samples are also reused within the mini-batch. We observe remarkable performance improvements on various benchmark datasets by using these negative sampling strategy.

3.4 Training

We optimize our proposed model using the total of three losses. At first, we use InfoNCE loss (Oord et al., 2018) for loss $\mathcal{L}_u$. We use cosine similarity as the scoring function ϕ for triples, which is defined by Eq 1. The additive margin γ_c helps the correct triple to get higher score. The temperature τ controls the importance weights of hard negatives. We make τ as a learnable parameter to learn along with the model during training (Wang et al., 2022a). The loss L_u can be expressed as below.

$$\mathcal{L}_u = -\log \frac{e^{(\phi(h_u, r_u t_u) - \gamma_c)/\tau}}{e^{(\phi(h_u, r_u, t_u) - \gamma_c)/\tau} + \sum e^{\phi(h_u, r_u, t_i')/\tau}} \tag{7}$$

Second, we use sigmoid loss with self-adversarial sampling for loss $\mathcal{L}_s$. The loss shows good performance in many structure-based KGE models (Sun et al., 2019). We use TransE as KGE scoring function ϕ. The loss $\mathcal{L}_s$ make the entity anchors learn structure information directly from KG. The loss $\mathcal{L}_s$ can be formulated as,

$$\mathcal{L}_s = -\log \sigma(\gamma_k - \phi(h_s, r_s, t_s))$$
$$- \sum_{i=1}^{|\mathcal{N}|} p(h_s, r_s, t_s^{i'}) \log \sigma(\phi(h_s, r_s, t_s^{i'}) - \gamma_k) \tag{8}$$

$$p(h_s, r_s, t_s^{j'}) = \frac{\exp(\phi(h_s, r_s, t_s^{j'}))}{\sum_i \exp(\phi(h_s, r_s, t_s^{i'}))}. \tag{9}$$

Finally, we define a loss L_a that aligns the representations created by anchors with the representations created by text encoding in the structure embedding space. By doing so, the two groups of embeddings can be located in similar distribution. The unified tail representations t_u are passed into the projection layer $g(\cdot)$ and the representation $g(t_u)$ is now placed in the structure embedding space. We use a linear layer for $g(\cdot)$. We use MSE for the loss $\mathcal{L}_a^{mse}$ to make $g(t_u)$ and tail structure embedding t_s closer.

$$\mathcal{L}_a^{mse} = \frac{1}{n} \sum (g(t_u) - t_s)^2 \tag{10}$$

In addition, we define a loss $\mathcal{L}_a^{margin}$ using a margin ranking loss that makes the unified representation $g(t_u)$ closer to tail structure embedding t_s and farther from head structure embedding h_s. We use the Euclidean distance for the distance function. The loss $\mathcal{L}_a$ is the sum of $\mathcal{L}_a^{mse}$ and $\mathcal{L}_a^{margin}$.

$$\mathcal{L}_a^{margin} =$$
$$\max(d(g(t_u), h_s) - d(g(t_u), t_s) + \gamma_m,\ 0) \tag{11}$$

$$\mathcal{L}_a = \mathcal{L}_a^{mse} + \mathcal{L}_a^{margin} \tag{12}$$

We simply sum the all three losses into a final loss $\mathcal{L}$. We update the parameters of the proposed model during optimizing the loss $\mathcal{L}$.

$$\mathcal{L} = \mathcal{L}_u + \mathcal{L}_s + \mathcal{L}_a \tag{13}$$

3.5 Inference

When predicting the tail entity, we use the both matching score and alignment score of the unified representation to calculate a final score. And then we find the entity which has the highest final score. At first, the unified representation matching score can be calculated as,

$$score_u = \frac{c_u \cdot t_u}{||c_u||\ ||t_u||}. \tag{14}$$

The alignment score measures the similarity between the transformed unified representations. Since we use TransE as the structure KGE function, the similarity score is calculated by distance between $g(c_u)$ and $g(t_u)$ which can be formulated,

$$score_s = -\lambda \cdot ||g(c_u) - g(t_u)||_2^2. \tag{15}$$

The final score is sum of these two scores. After calculating the final score, we select the tail entity with the highest final score.

$$score = score_u + score_s \tag{16}$$

$$arg\max_{t_i}\ score(h, r, t_i),\ t_i \in \mathcal{E} \tag{17}$$

Additionally, we adopt the entity re-ranking method (Wang et al., 2022a). This strategy helps improve performance of text-based KGC methods by explicitly using the spatial locality of the KG. We increase the score of a candidate tail entity $\mathtt{t}$ by α if it is in the k-hop neighbors set of head entity $\mathtt{h}$ and decrease the score by β if it is the same as the head entity, based on the training graph.

$$arg\max_{t_i}\ score(h, r, t_i) + \alpha\mathbf{1}(t_i \in \mathcal{E}_k(h)) - \beta\mathbf{1}(t_i = h) \tag{18}$$

Since we do not use any entity-specific transformation vector, but only uses entity anchors and the text description of the entity to calculate the score, the proposed model can compute representations for any unseen entities. Therefore, the proposed method does not lose the inductive reasoning capability.

4 Experiments

To evaluate our model, we compare with various KGC methods on three datasets. We also perform various analyses to identify the contributions of each module, and pros and cons of SEA-KGC model. We implement our experiments using PyTorch (Paszke et al., 2019) framework and run them on 4 NVIDIA V100 GPUs.[1]

[1]The codes of our method will be updated soon.

Dataset	#entity	#relation	#train	#valid	#test
FB15K237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
Wikidata5M(Transductive)	4,594,485	822	20,614,279	5,163	5,163
Wikidata5M(Inductive)	4,579,609	822	20,496,514	6,699	6,894

Table 1: Statistics of the benchmark datasets.

4.1 Experiment Settings

4.1.1 Datasets and Evaluation Metrics

WN18RR (Dettmers et al., 2018) and FB15K-237 (Toutanova and Chen, 2015) are subsets of WN18 (Bordes et al., 2013; Miller, 1995) and FB15K (Bordes et al., 2013; Bollacker et al., 2008), respectively. These datasets solve the problem of data leakage by removing inverse relations. We use the textual descriptions of entities in WN18RR and FB15K-237, which are published by KG-BERT (Yao et al., 2019). Wikidata5M (Wang et al., 2021b) consists of two settings of the dataset: transductive and inductive. The detailed statistics of the datasets are shown in Table 1. We use the most widely used metrics for model evaluation: mean reciprocal rank (MRR), and hits at N (Hit@N). All evaluation metrics are computed under the filtered setting introduced in (Bordes et al., 2013).

4.1.2 Hyperparameters

We adopt BERT-Base (Devlin et al., 2018) for PLM layer in the SEA-KGC. We use AdamW (Loshchilov and Hutter, 2017) with cosine learning rate decaying to optimize the experimental models. Some hyper-parameters validated in the following range, learning rate $\in$ {1e-5, 3e-5, 5e-5}, and β for re-ranking $\in \{0.0, 0.1, 0.2\}$. We set the other hyper-parameters with same values regardless of datasets. We fix the batch size to 512. The number of entity anchor is 10. The maximum input token length is 60, included entity anchors. The temperature τ and the additive margin γ_i for infoNCE loss are 0.05 and 0.02, respectively. We set α to 0.05 for entity re-ranking. The margin of KGE scoring function γ_k and margin ranking loss γ_m are 9.0 and 1.0, respectively. The weight of the alignment score λ is set to 0.01. More details are described in Appendix A.1.

4.2 Main Results

We compare the performance of SEA-KGC with various text-based KGC methods and some popular structure-based methods. Our results are statistically significant under t-test with p-value 0.05. As shown in Table 2 and Table 3, SEA-KGC

Method	FB15K-237				WN18RR			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
Structure-based Methods								
TransE (Bordes et al., 2013)†	27.9	19.8	37.6	44.1	24.3	4.3	44.1	53.2
DistMult (Yang et al., 2014)†	28.1	19.9	30.1	44.6	44.4	41.2	47.0	50.4
ComplEx (Trouillon et al., 2016)†	27.8	19.4	29.7	45.0	44.9	40.9	46.9	53.0
RotatE (Sun et al., 2019)†	33.8	24.1	37.5	53.3	47.6	42.8	49.2	57.1
TuckER (Balažević et al., 2019)†	35.8	26.6	39.4	54.4	47.0	44.3	48.2	52.6
NePTuNe (Sonkar et al., 2021)	36.6	27.2	**40.4**	54.7	49.1	45.5	50.7	55.7
Text-based Methods								
KG-BERT (Yao et al., 2019)†	-	-	-	42.0	21.6	4.1	30.2	52.4
StAR (Wang et al., 2021a)	29.6	20.5	32.2	48.2	40.1	24.3	49.1	70.9
KGT5 (Saxena et al., 2022)‡	27.6	21.0	-	41.4	50.8	48.7	-	54.4
KG-S2S (Chen et al., 2022)	33.6	25.7	37.3	49.8	57.4	53.1	59.5	66.1
SimKGC (Wang et al., 2022a)	33.6	24.9	36.2	51.1	**66.6**	**58.7**	**71.7**	**80.0**
LMKE (Wang et al., 2022b)	30.6	21.8	33.1	48.4	61.9	52.3	67.1	78.9
LASS-RoBERTa (Shen et al., 2022)	-	-	-	53.3	-	-	-	78.6
CSProm-KG (Chen et al., 2023)	35.8	26.9	39.3	53.8	57.5	52.2	59.6	67.8
SEA-KGC	**36.7**	**27.5**	40.1	**55.3**	65.3	57.7	69.6	79.5

Table 2: Main results for FB15K-237 and WN18RR. Results of [†] are taken from (Wang et al., 2021a), results of [‡] are taken from (Chen et al., 2023), and the other results are taken from the corresponding papers. Bold numbers represent the best and underlined numbers represent the second best.

Method	Wikidata5M-Transductive				Wikidata5M-Inductive			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
Structure-based Methods								
TransE (Bordes et al., 2013)††	25.3	17.0	31.1	39.2	-	-	-	-
RotatE (Sun et al., 2019)††	29.0	23.4	32.2	39.0	-	-	-	-
Text-based Methods								
DKRL (Xie et al., 2016)††	16.0	12.0	18.1	22.9	23.1	5.9	32.0	54.6
KEPLER (Wang et al., 2021b)	21.0	17.3	22.4	27.7	40.2	22.2	51.4	73.0
BLP-SimpleE (Daza et al., 2021)	-	-	-	-	49.3	28.9	63.9	86.6
SimKGC (Wang et al., 2022a)	35.8	31.3	37.6	44.1	71.4	60.9	78.5	91.7
KGT5 (Saxena et al., 2022)	30.0	26.7	31.8	36.5	-	-	-	-
CSProm-KG (Chen et al., 2023)	38.0	**34.3**	39.9	44.6	-	-	-	-
SEA-KGC	**38.6**	33.8	**40.4**	**47.4**	**73.0**	**62.5**	**80.7**	**92.3**

Table 3: Main results for Wikidata5M both settings. Results of [††] are taken from (Wang et al., 2022a). We follow the evaluation protocol used in (Wang et al., 2022a).

outperforms state-of-the-art methods on FB15K-237, Wikidata5M(Transductive & Inductive). Our model also shows competitive performance on WN18RR. Especially, SEA-KGC outperforms not only text-based methods but also structure-based methods on FB15K-237. Generally, structure-based methods perform better than text-based method on FB15K-237 which has more dense connectivities and requires complex structural information. To the best of our knowledge, the SEA-KGC is the first text-based method which outperforms SOTA structure based methods on FB15K-237 benchmark.

4.3 Ablation Studies

4.3.1 Impact of Structure KGE Functions

As presented in section 2.1, there are various structure-based knowledge graph embedding methods that can be adopted to the structural entity embeddings in SEA-KGC. To evaluate the influence of different KGE functions, we conduct experiments in which we substitute the TransE KGE scoring function with alternative KGE functions, including DistMult, ComplEx, and RotatE. Resultingly, it is evident that different KGE functions do not make a substantial impact on the results. In Table 4, we can observe that the original SEA-KGC shows more favorable performance on FB15K-237 when com-

pared to models employing other KGE functions. Figure 3 also illustrates the strong MRR performance of the model using the TransE KGE function on both FB15K-237 and WN18RR datasets.

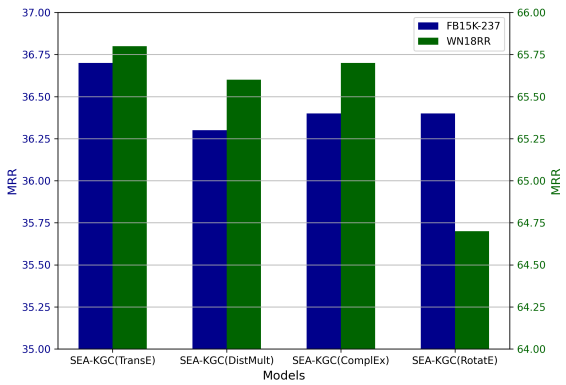

Figure 3: MRR of SEA-KGC with various KGE function on FB15K-237 and WN18RR

4.3.2 Effect of Entity Anchors

In this section, we analyze the effect of an entity anchor's initializing method on the model performance. As shown in Table 4, when learning with anchors initialized to random vectors, the performance decreases slightly compared to original SEA-KGC. In particular, the performance gap is bigger in Hit@1 and MRR. Therefore, finding representative centroids through the clustering and using them as anchors is helpful in learning structured information of entities.

We also perform ablation on the number of entity anchors. We compare each performance of the models when the number of entity anchor is 3, 5, 10, 20, and 50, respectively. As shown in Figure 4, as the number of anchors increases, model performance also increases. However, when the number of anchors exceeds 10, performance is saturated.

4.3.3 Effect of Negative Sampling

We train SEA-KGC using in-batch contrastive learning with additional uniform random negatives. To check the effectiveness of additional uniform random negatives, we expreiment the training of SEA-KGC without uniform random negatives. Table 4 shows the results of negative sampling ablation. The performance of SEA-KGC without uniform random negatives drops by $\sim 4\%$ in all metrics compared to the original model. This result shows the most dramatic difference of all ablation experiments. We also experimentally verify the

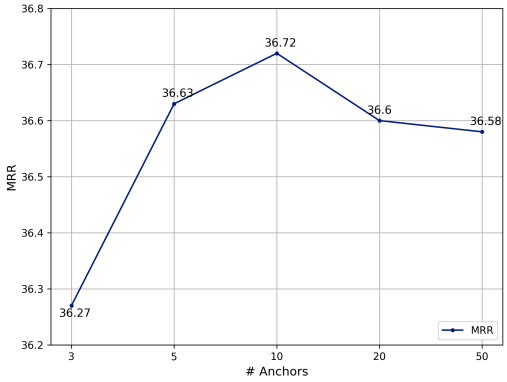

Figure 4: MRR on FB15K-237 according to the change of the number of anchors. The performance is slightly decreasing when the number of anchors is larger than 10.

limitations of in-batch negative learning and the effectiveness of uniform negative sampling.

4.3.4 Various Losses Ablation

The proposed model is designed to optimize total of three different losses. We perform ablation on the loss to check the effect of each loss. First, we remove structure and alignment losses and train the SEA-KGC using only unified semantic loss. We observe that removing structure loss causes performance degradation, by 0.2 MRR and 0.3 Hit@10. This performance gap can be explained by the reason that entity anchors become meaningless tokens because they do not properly express structural information. We also observe a performance drop (i.e., 0.2 MRR and 0.1 Hit@10) when only the alignment loss is removed from SEA-KGC. This experiment verifies that explicit alignment is necessary, although unified representation has implicit alignment inherent.

Model	MRR	Hit@1	Hit@10
SEA-KGC	36.7	27.5	55.3
SEA-KGC w/ DistMult	36.3	27.1	55.0
SEA-KGC w/ Complex	36.4	27.1	54.9
SEA-KGC w/ RotatE	36.4	27.1	55.2
SEA-KGC w/ random initialized anchors	36.1	26.6	55.0
SEA-KGC w/o additional negatives	32.8	23.6	51.8
SEA-KGC w/o structure and alignment losses	36.5	27.3	55.0
SEA-KGC w/o alignment loss	36.5	27.1	55.2

Table 4: The results of ablation on FB15K-237.

4.4 Case Study

In this section, we give qualitative analysis by comparing a baseline model and our SEA-KGC. We select SimKGC (Wang et al., 2022a) as a baseline

Input (head and relation):
(Charlotte Gainsbourg [*is an Anglo-French actress and singer...*], /person/nationality)
Prediction (Baseline):
P1: Normandy [*is a geographical region of France...*]
P2: Brittany [*is a cultural region in the north-west of France...*]
Prediction (SEA-KGC):
P1: **England [*is a part of the United Kingdom...*]**
P2: United Kingdom [*The Uinted Kingdom of Great Britain and Northern Ireland...*]
Input (head and relation):
(Theodore Roosevelt [*was an American author, ... and politician, ...*], /government_position_held/basic_title)
Prediction (Baseline):
P1: Member of Congress [*is a person who has been appointed or elected and inducted into some official body...*]
P2: Secretary of State [*is a commonly used title for a senior or mid-level post in governments...*]
Prediction (SEA-KGC):
P1: **President [*is a leader of an organization, company, community, club...*]**
P2: Secretary of State [*is a commonly used title for a senior or mid-level post in governments...*]

Table 5: Case study on FB15K-237. Texts in brackets are description for entities. Bold text represents the ground truth entity.

Input (head and relation):
(Geography [*study of the earth's surface...*], hypernym)
Prediction (Baseline):
P1: **Earth Science [*any of the sciences that deal with the earth...*]**
P2: Subject Field [*a branch of knowledge...*]
Prediction (SEA-KGC):
P1: Scientific Discipline [*a particular branch of scientific knowledge...*]
P2: Subject Field [*a branch of knowledge...*]
Input (head and relation):
(Mucus [*protective secretion of the mucus*], hypernym)
Prediction (Baseline):
P1: **Secretion [*a functionally specialised substance...*]**
P2: Internal Secretion [*the secretion of an endocrine gland...*]
Prediction (SEA-KGC):
P1: Liquid Body Substance [*the liquid parts of the body*]
P2: **Secretion [*a functionally specialised substance...*]**

Table 6: Error analysis on WN18RR. Texts in brackets are description for entities. Bold text represents the ground truth entity.

model, because the model has simple architecture based on PLM but shows competitive performance on various benchmarks.

We show some examples to illustrate how SEA-KGC outperforms the baseline model. Table 5 shows the predictions of each model for some test examples in FB15K-237. In the first case, the baseline model ranks two different provinces in France as the predictions. This could be caused by the keyword *"French"* in the description of head entity. However, SEA-KGC returns *"England"* as tail entity correctly. In the second case, SEA-KGC selects correct tail entity, *"President"*, as answer from many similar government positions in KG using the trained structure information while the baseline model misses the answer. This is probably because the description of the head entity contains a variety of occupation and political career information, making the baseline model confused to find the correct answer based on textual similarity. From these results, we can figure out that SEA-KGC can capture both text and structure information, while the the baseline focuses excessively on textual information.

4.5 Error Analysis

We perform error case analysis to confirm the limitations of SEA-KGC. We find that the most frequent errors of the proposed method on WN18RR test dataset are caused by *hypernym* relation. In our experiments, more than 60% of the wrong predictions are linked to *hypernym* relation. For more detailed analysis, we again compare the baseline model and SEA-KGC. We find that several error cases in which the baseline model predicts correct answer, while the SEA-KGC does not. First, SEA-KGC tends to predict *hypernym*s in a broader sense. As shown in Table 6, it seems hard to say that the top-2 tail entity predictions returned by SEA-KGC are completely wrong. Second, we find that the text semantics often overlap in the description of the head entity and answer tail entity. In the first example, *"earth"* in the head description and *"earth"* in the tail completely overlap. In the second example, *"secretion"* in the head description and *"secretion"* in the tail also overlap. Therefore, in these cases, the baseline model that mainly focuses on text similarity provides better answers. On the other hand, SEA-KGC, which considers structure information, predicts a more diverse and wide hierarchical range of *hypernym*s. Through error analysis, we confirm that some errors may occur in SEA-KGC due to above reasons.

5 Conclusion

In this paper, we propose a novel method SEA-KGC that effectively unifies text and structure information using structured entity anchors. We use entity anchors in the proposed model to utilize structural information effectively while the model still enables inductive reasoning. Additionally, we figure out the problem with in-batch negative sampling and alleviate the problem by introducing additional uniform random negatives. In our experiments on various benchmark datasets, and SEA-KGC outperforms SOTA KGC methods on FB15K-237 and Wikidata5M and shows competitive results on WB18RR.

For future work, we plan to adopt larger language models to our method and try to develop a model that creates powerful synergy between KG information and text semantics. Another direction is negative sampling strategy. We plan to study effective hard negative sampling methods that can also be used in in-batch contrastive learning.

6 Limitations

We utilize PLM encoder in SEA-KGC to integrate textual and structural information. As a result, the increase in computational cost of the model is much bigger than the increase in performance on open-domain KG(e.g. FB15K-237) compared to structured based methods. SEA-KGC has a huge performance improvement on KG where word semantics is important (e.g. WN18RR), the cost cannot be ignored. Additionally, the input sequence length for PLM is fixedly increased by the number of entity anchors. It causes more computational resources for training and inference. In our experiments, a small number of anchors was sufficient, but it can be changed when training much larger KGs. In the future, we plan to overcome these issues for SEA-KGC.

References

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. 2022. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. *arXiv preprint arXiv:2209.07299*.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023. Dipping plms sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. *arXiv preprint arXiv:2307.01709*.

Bonggeun Choi, Daesik Jang, and Youngjoong Ko. 2021. Mem-kgc: Masked entity model for knowledge graph completion with pre-trained language model. *IEEE Access*, 9:132025–132032.

Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*, pages 798–808.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L Hamilton. 2021. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. *arXiv preprint arXiv:2106.12144*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696.

Paul Pu Liang, Manzil Zaheer, Yuan Wang, and Amr Ahmed. 2020. Anchor & transform: Learning sparse embeddings for large vocabularies. *arXiv preprint arXiv:2003.08197*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2023. Unifying large language models and knowledge graphs: A roadmap. *arXiv preprint arXiv:2306.08302*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. *arXiv preprint arXiv:2203.10321*.

Jianhao Shen, Chenguang Wang, Linyuan Gong, and Dawn Song. 2022. Joint language semantic and structure embedding for knowledge graph completion. *arXiv preprint arXiv:2209.08721*.

Shashank Sonkar, Arzoo Katiyar, and Richard G Baraniuk. 2021. Neptune: Neural powered tucker network for knowledge graph completion. *arXiv preprint arXiv:2104.07824*.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021a. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, pages 1737–1748.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022a. Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. *arXiv preprint arXiv:2203.02167*.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Xintao Wang, Qianyu He, Jiaqing Liang, and Yanghua Xiao. 2022b. Language models as knowledge embeddings. *arXiv preprint arXiv:2206.12617*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

A Appendix

A.1 Details on Hyperparameters and Settings

Hyperparameter	Value
batch size	512
# of anchors	10
# of max token	60
gradient clip	10
warm-up steps	800
weight decay	1e-4
τ	0.05
γ_i	0.02
γ_k	9.0
γ_m	1.0
λ	0.01

Table 7: Common hyperparameter settings for all benchmarks

Hyperparameter	**FB15K-237**	**WN18RR**	**Wikidata5M-Trans.**	**Wikidata5M-Ind.**
lr	5e-5	1e-5	3e-5	3e-5
α	0.05	0.05	0.0	0.0
β	0.0	0.0	0.0	0.2

Table 8: The best hyperparameter settings for each benchmark

We list the best hyperparameters of SEA-KGC on the benchmarks. Table 7 shows the hyperparameters shared across all the benchmarks. Table 8 summarizes the best hyperparameters on each benchmark.

We train SEA-KGC model 50 epochs on WN18RR and 10 epochs on FB15K-237. We train only 1 epoch on Wikidata5M (both settings). Training each epoch takes about 24 minutes for FB15K-237, 8 minutes for WN18RR, and 22 hours for Wikidata5M (both settings).

We use open-source *transformers*[2] library to implement the PLM layer in SEA-KGC. We initialize the PLM layer with pre-trained $BERT_{Base}$ (Devlin et al., 2018) weights.

We adopt the inverse relation strategy from SimKGC (Wang et al., 2022a) to predict head entity given tail and relation. For inverse relation r^{-1}, same as SimKGC, we add a prefix word *"inverse"* to the description of r.

[2] https://github.com/huggingface/transformers