KG-FIT: Knowledge Graph Fine-Tuning Upon Open-World Knowledge

Patrick Jiang

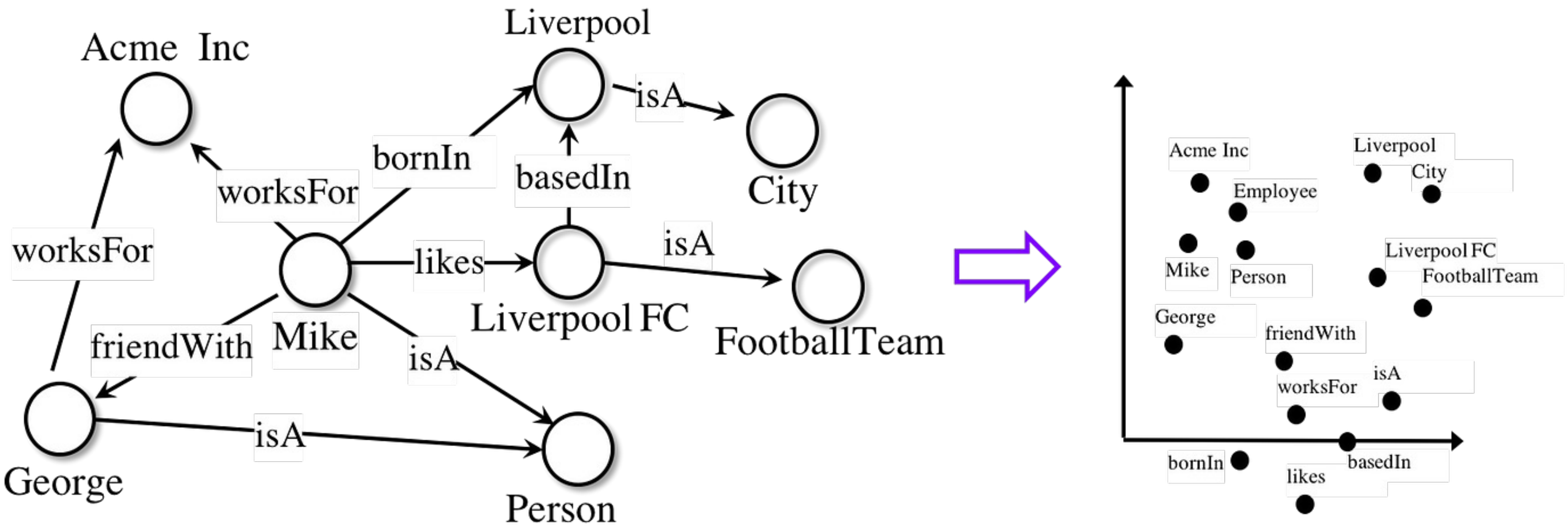
May 30th, 2024

Overview

- Background
- Related Works
- Problems
- Methodology
- Experiments & Findings
- Conclusions

Background

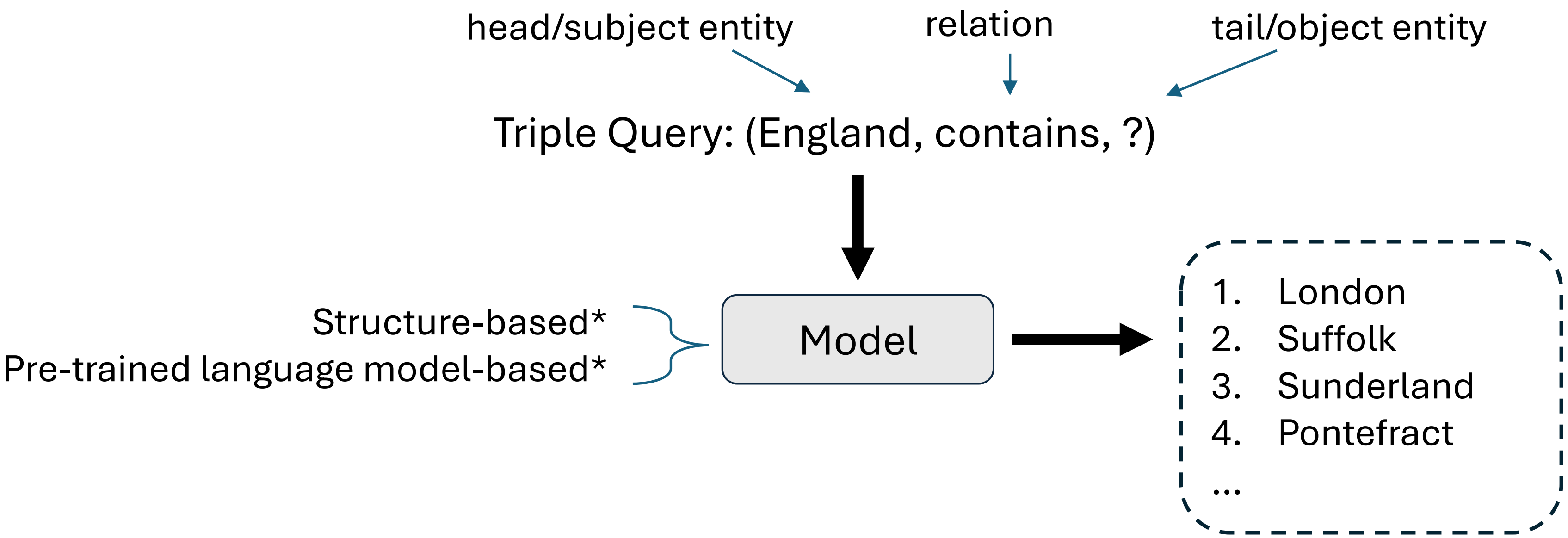
- **From Knowledge Graph (KG) to Knowledge Graph Embedding (KGE)**



KGE transforms entities and relations of a KG into continuous vector spaces, enabling efficient computation and facilitating tasks like link prediction, entity resolution, and recommendation.

Background

- **Link Prediction for Knowledge Discovery**

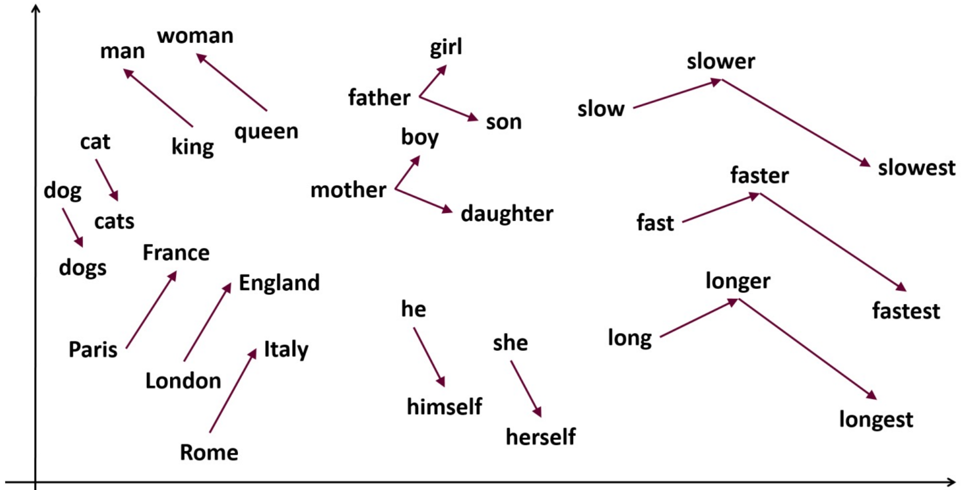


* We do not include rule-based or RL-based methods in the discussion as their performance is no longer competitive.

Related Works

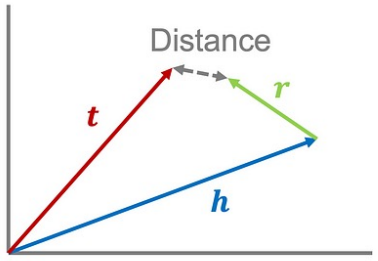
- **Structure-based Methods**

Map entities and relations into vector space



(conceptual figure)

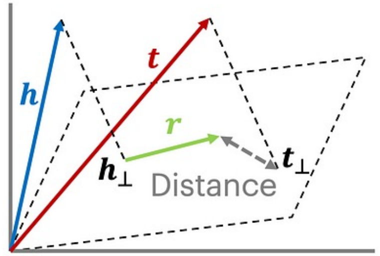
TransE



$f(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = -\|\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}\|_p$

$\boldsymbol{h},\boldsymbol{r},\boldsymbol{t} \in \mathbb{R}^d$

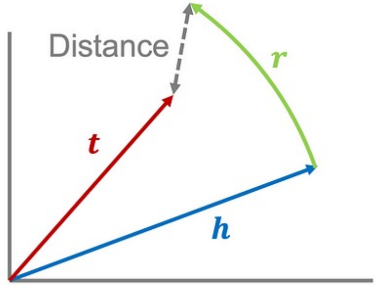
TransH



$f(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = -\|\boldsymbol{h}_\perp+\boldsymbol{r}-\boldsymbol{t}_\perp\|_p$

$\boldsymbol{h},\boldsymbol{r},\boldsymbol{t} \in \mathbb{R}^d$

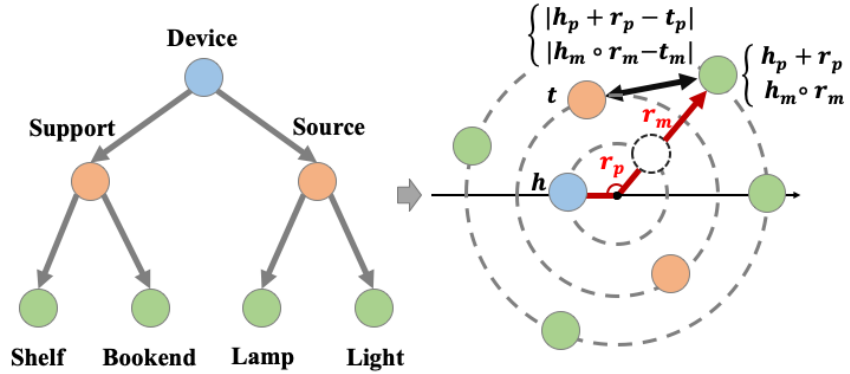
RotatE



$f(\boldsymbol{h},\boldsymbol{r},\boldsymbol{t}) = -\|\boldsymbol{h}\circ\boldsymbol{r}-\boldsymbol{t}\|_p$

$\boldsymbol{h},\boldsymbol{r},\boldsymbol{t} \in \mathbb{C}^d, |r_i| = 1\ \forall\, i$

HAKE

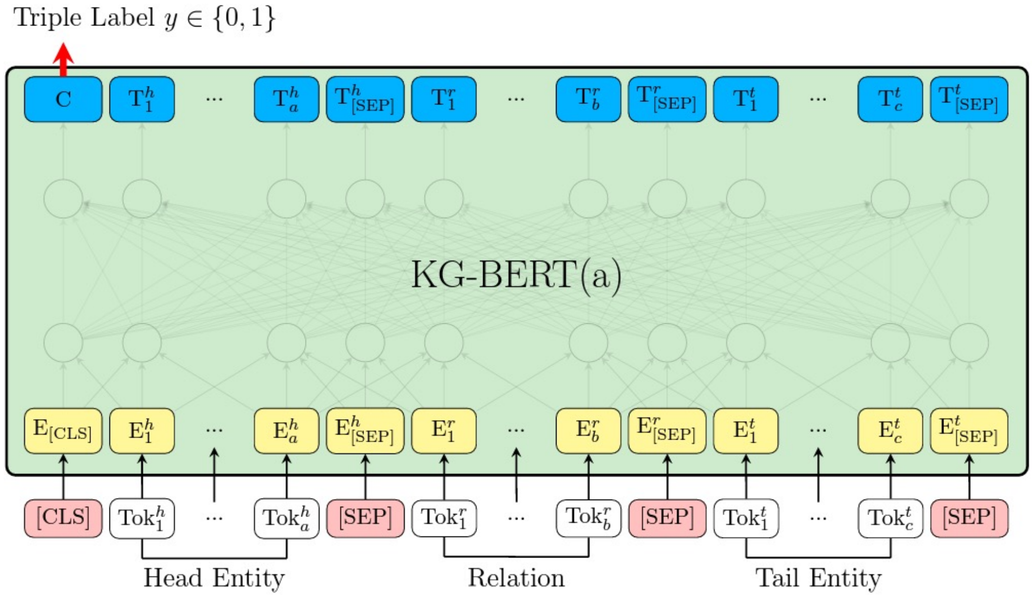


$-\|\mathbf{h}_m \circ \mathbf{r}_m - \mathbf{t}_m\|_2 - \lambda\|\sin((\mathbf{h}_p+\mathbf{r}_p-\mathbf{t}_p)/2)\|_1$ $\quad \mathbf{h}_m,\mathbf{t}_m \in \mathbb{R}^k, \mathbf{r}_m \in \mathbb{R}^k_+,$ $\mathbf{h}_p,\mathbf{r}_p,\mathbf{t}_p \in [0,2\pi)^k, , \lambda \in \mathbb{R}$

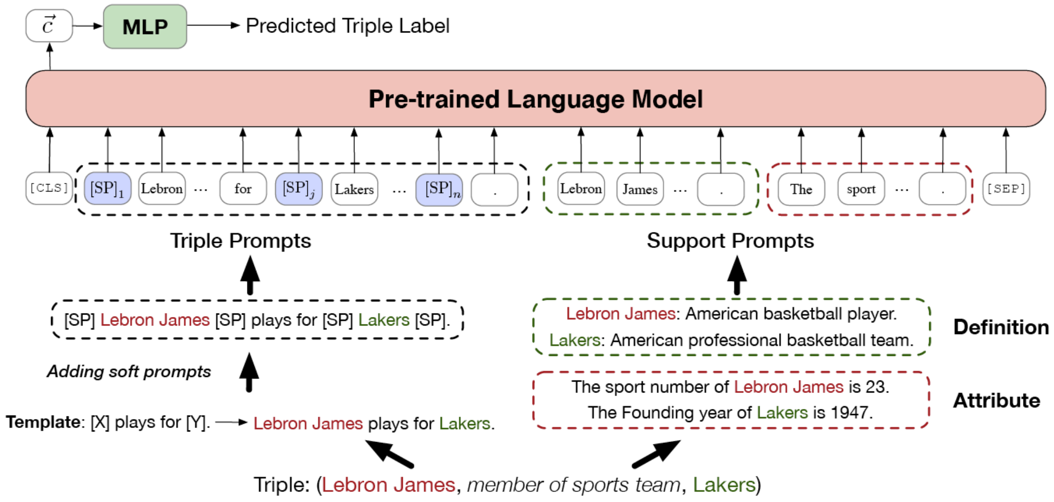
…

Related Works

- **PLM-based Methods**



KG-BERT: fine-tune a PLM with sliced triples



PKGC (ACL'22) / TagReal (ACL'23):
fine-tune a PLM with templated sliced triples

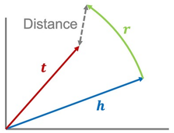
$$\mathcal{L} = -\log \frac{e^{(\phi(h,r,t)-\gamma)/\tau}}{e^{(\phi(h,r,t)-\gamma)/\tau} + \sum_{i=1}^{|\mathcal{N}|} e^{\phi(h,r,t_i')/\tau}}$$

SimKGC (ACL'22):
fine-tune a PLM with contrastive learning

⋮

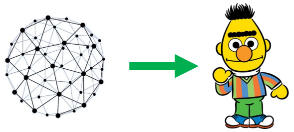
Problems

Local Knowledge



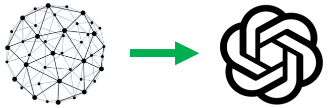
Structure-based Methods

Global Knowledge



Small-scale PLMs

Up-to-Date Global Knowledge



Fast-Iterating LLMs

Pros over PLMs/LLMs:

- Fast Training/Inference
- Low Resource
- Interpretable Embeddings
- Robustness to Sparse Data

Pros over Structure-based Methods:

- Abundant External Knowledge
- Handling Linguistic Ambiguity

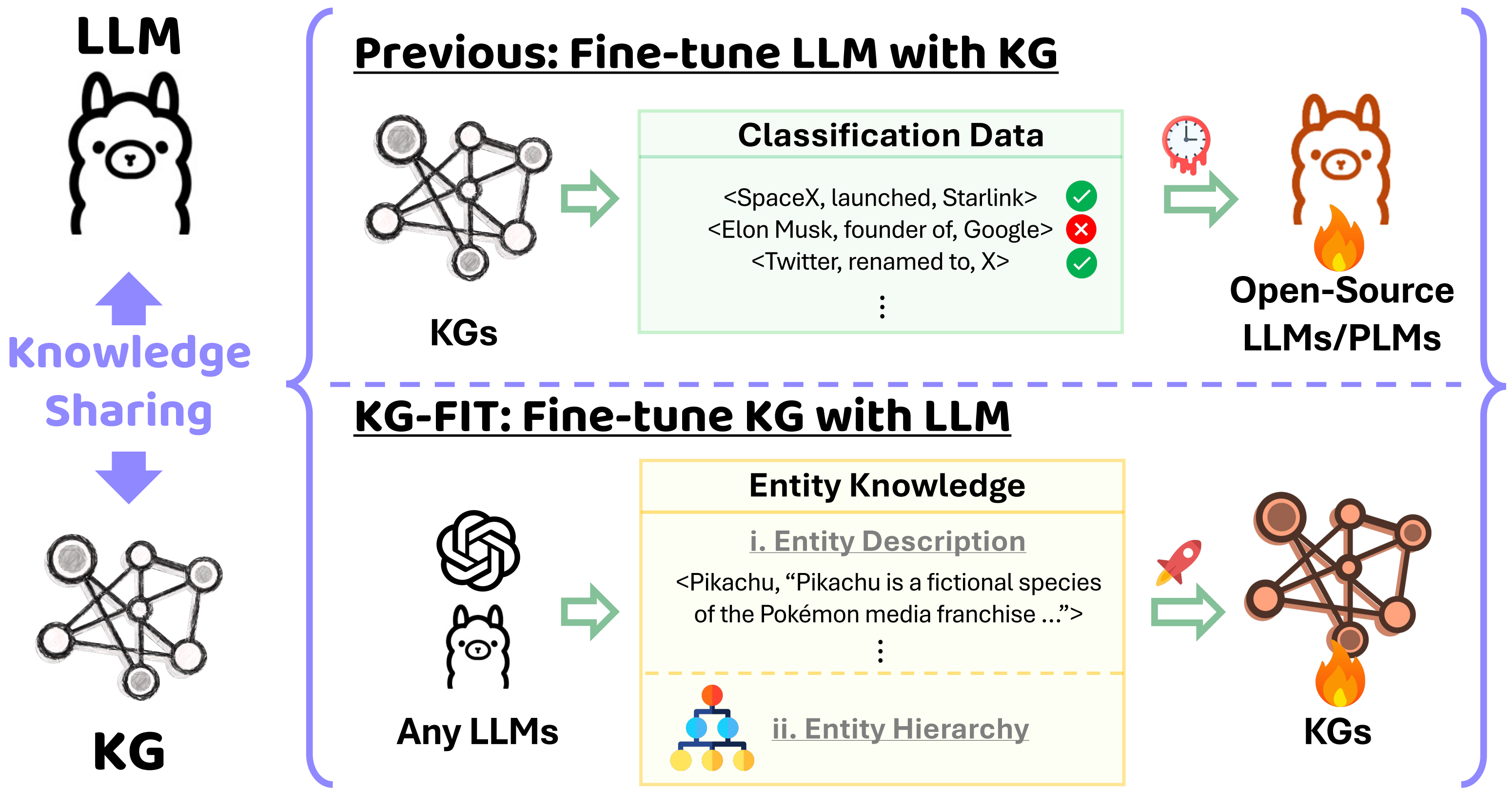
Pros over PLMs:

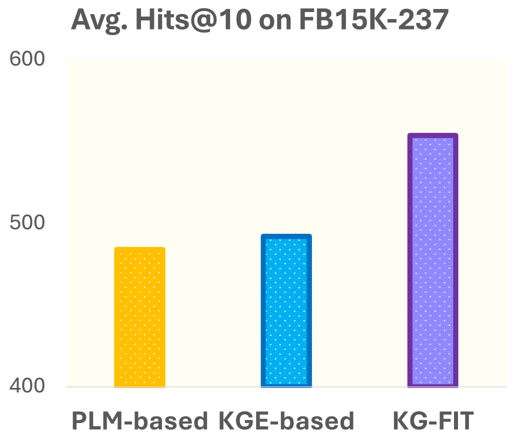
- Up-to-Date Knowledge
- More comprehensive understanding of entities

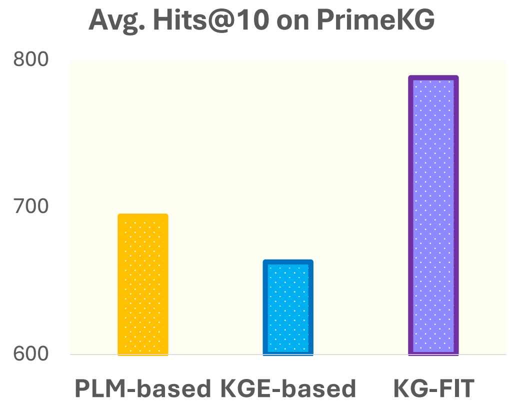
Can we combine them?

Methodology: KG-FIT

- Abstractive Overview

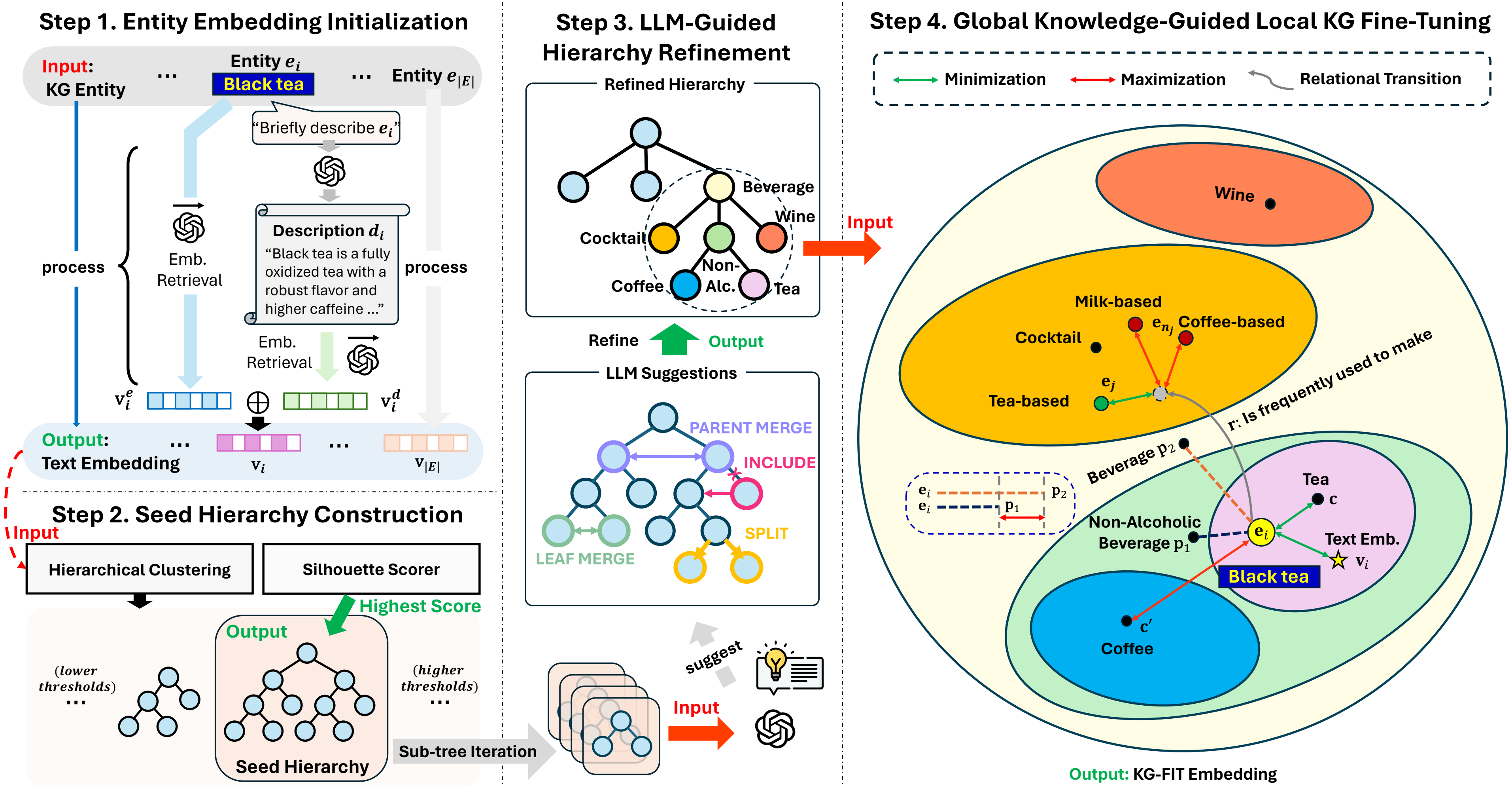






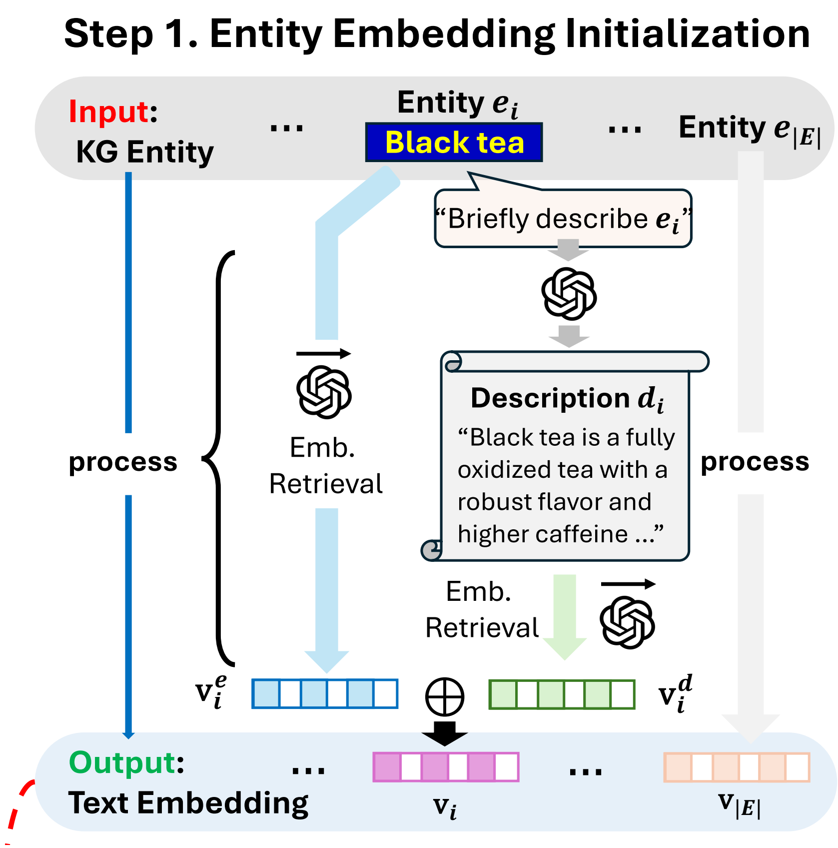
Methodology: KG-FIT

- **KG-FIT Framework**



Methodology: KG-FIT

- **Illustration – Step 1**



(1) We generate descriptions of all the entities within the KG using an LLM.

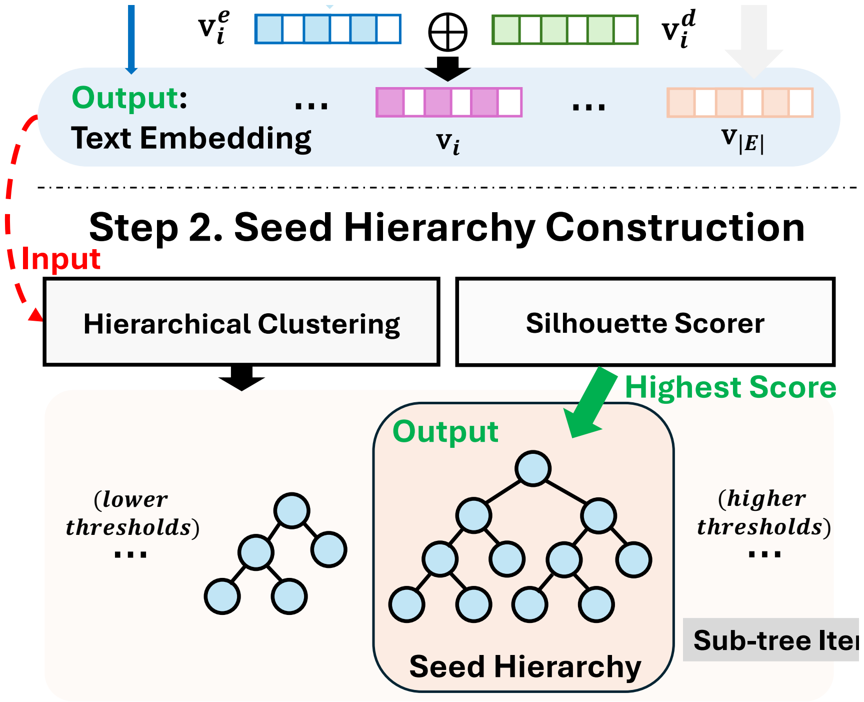
(2) We concatenate the embeddings of the entity name and the entity description as the initial entity embedding.

$$\mathbf{v}_i = [\mathbf{v}_i^e; \mathbf{v}_i^d]$$

KG-FIT is "<u>**K**</u>nowledge <u>**G**</u>raph <u>**FI**</u>ne-<u>**T**</u>uning" as we are using LLM's pre-trained text embedding as the starting point.

Methodology: KG-FIT

- **Illustration – Step 2**



(1) We apply agglomerative clustering to the text embedding of all the entities in the KG.

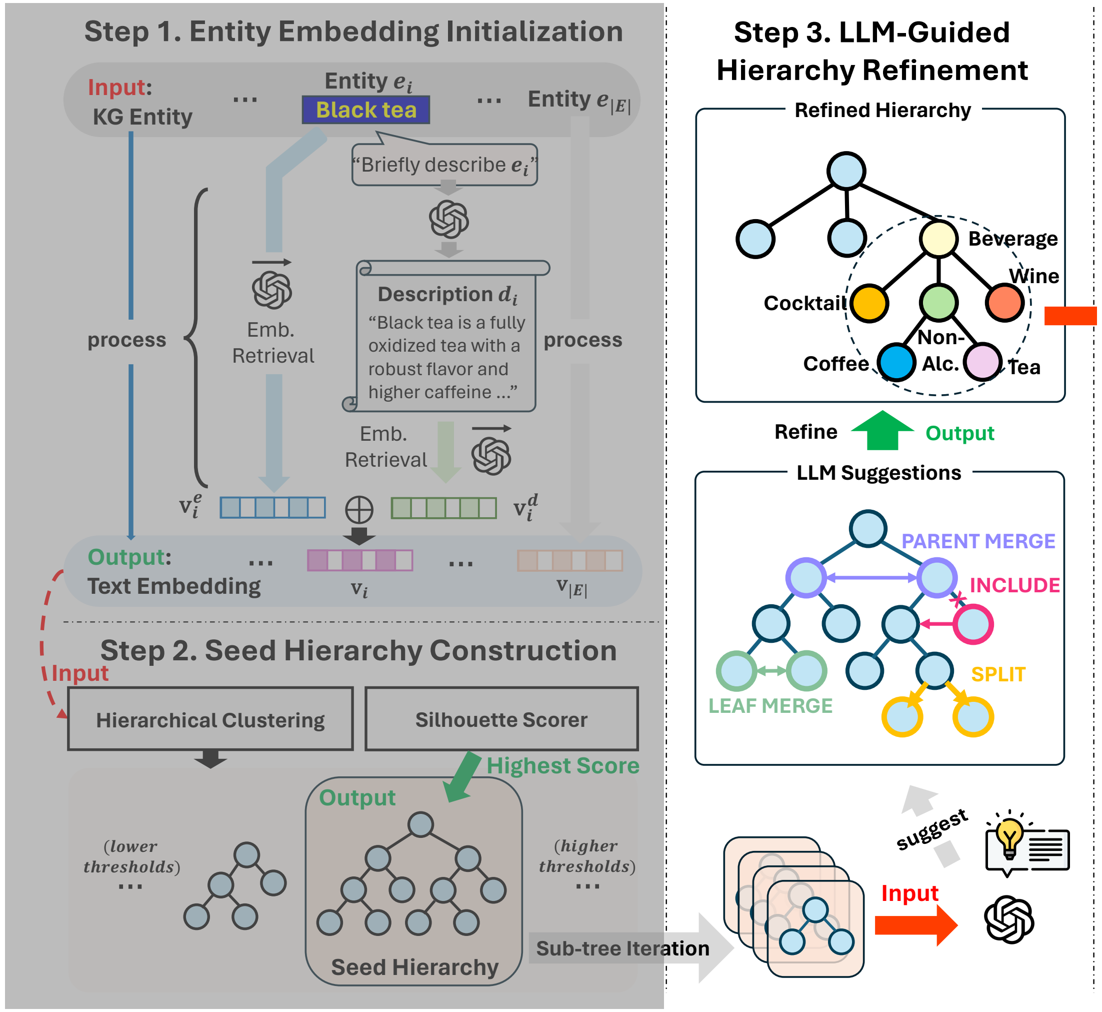
(2) We use silhouette score S^* to select the optimal hierarchy among those with different distance thresholds, as the seed hierarchy $\mathcal{H}_{\text{seed}}$

$$\tau_{\text{optim}} = \arg\max_{\tau \in [\tau_{\min}, \tau_{\max}]} S^*(\mathbf{V}, \text{labels}_\tau)$$

* The silhouette score is a metric used to evaluate the quality of clustering by measuring how similar an object is to its own cluster compared to other clusters, providing a succinct and effective assessment of the separation and cohesion of the clusters.

Methodology: KG-FIT

- **Illustration – Step 3**



The seed hierarchy is a binary tree, which may not optimally represent real-world entity knowledge.
Thus, we refine it with LLM's understanding of entities.

(1) For each leaf cluster, we prompt the LLM to split it into subclusters if needed, resulting in $\mathcal{H}_{\text{split}}$

$$C_{\text{split}} = \text{LLM}(\mathcal{P}_{\text{SPLIT}}(C_{\text{original}})), \quad C_{\text{original}} \to C_{\text{split}} = \{C_1, C_2, \ldots, C_k\}$$

(2) For each sub-tree (parent-child triple) in $\mathcal{H}_{\text{split}}$, we refine it through a series of actions:

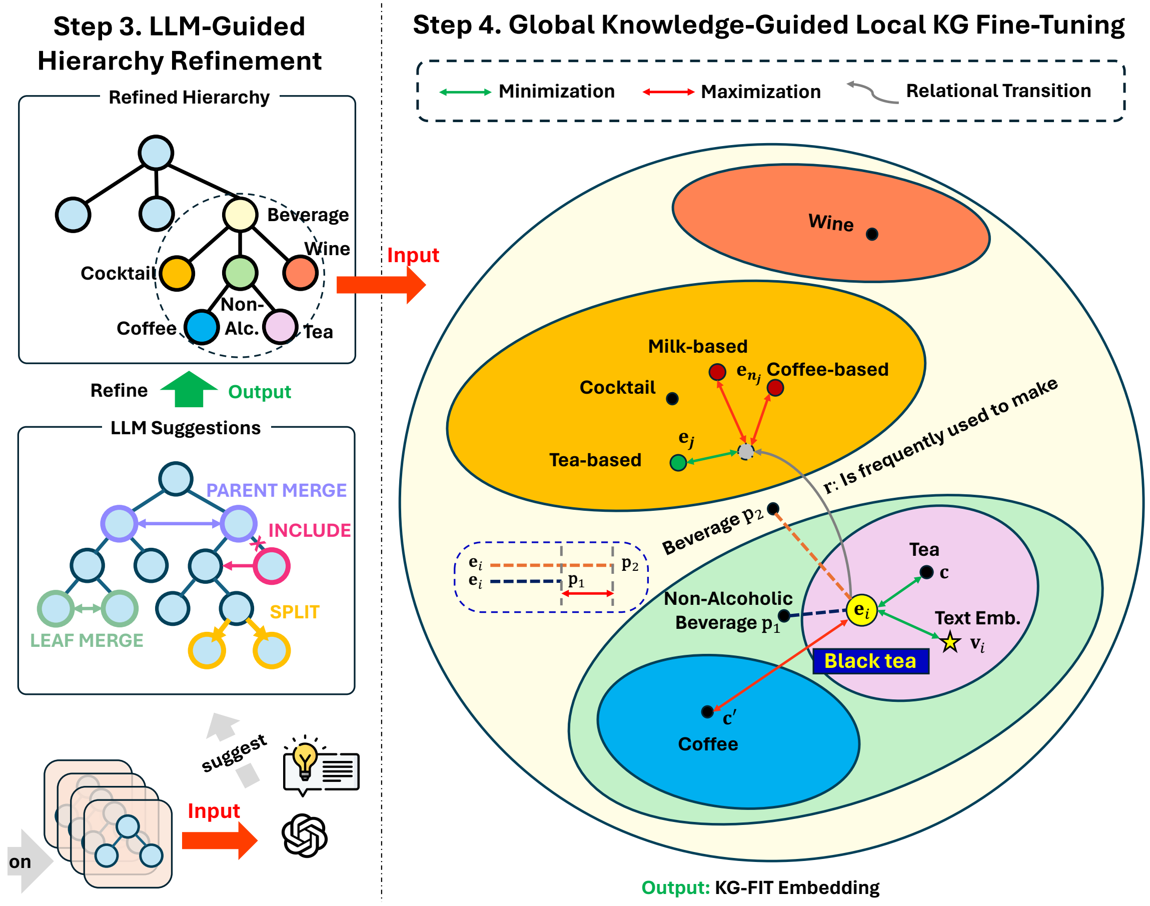
$$(P', L', R') = \text{LLM}(\mathcal{P}_{\text{REFINE}}(P, L, R))$$

- No Update: $(P', L', R') = (P, L, R)$.
- Parent Merge: $P' = P \cup L \cup R, L' = \emptyset, R' = \emptyset$.
- Leaf Merge: $P' = \{e_1, \ldots, e_p\}, L' = \emptyset, R' = \emptyset$, where $\{e_1, \ldots, e_p\} = L \cup R$.
- Include: $P' = P \cup R, L' = L, R' = \emptyset$ or $P' = P \cup L, L' = \emptyset, R' = R$.

which results in $\mathcal{H}_{\text{LHR}}$

Methodology: KG-FIT

- **Illustration – Step 4**



(1) Initialization of Entity and Relation Embeddings

$$\mathbf{e}_i = \rho \mathbf{e}'_i + (1-\rho)\mathbf{v}'_i, \quad \mathbf{r}_j \sim N(0,\psi^2)$$

$\mathbf{e}'_i \in \mathbb{R}^n$ Random embedding

$\mathbf{v}'_i = [\mathbf{v}^e_i[:\frac{n}{2}]; \mathbf{v}^d_i[:\frac{n}{2}]] \in \mathbb{R}^n$ Sliced text embedding

(2) Hierarchical Clustering Constraints

$$\mathcal{L}_{\text{hier}} = \sum_{e_i \in E} \Bigg(\underbrace{\lambda_1 d(\mathbf{e}_i, \mathbf{c})}_{\textit{Cluster Cohesion}} \underbrace{- \lambda_2 \sum_{C' \in \mathcal{S}_m(C)} \frac{d(\mathbf{e}_i, \mathbf{c}')}{|\mathcal{S}_m(C)|}}_{\textit{Inter-level Cluster Separation}} \underbrace{- \lambda_3 \sum_{j=1}^{h-1} \frac{\beta_j (d(\mathbf{e}_i, \mathbf{p}_{j+1}) - d(\mathbf{e}_i, \mathbf{p}_j))}{h-1}}_{\textit{Hierarchical Distance Maintenance}} \Bigg)$$

(3) Semantic Anchoring Constraint

$$\mathcal{L}_{\text{anc}} = - \sum_{e_i \in \mathcal{E}} d(\mathbf{e}_i, \mathbf{v}'_i)$$

(crucial for large clusters where the diversity of entities may cause the fine-tuned embeddings to drift from original semantics)

(4) Score Function-Based Fine-Tuning

$$\mathcal{L}_{\text{link}} = - \sum_{(e_i, r, e_j) \in \mathcal{D}} \Bigg(\log \sigma(\gamma - f_r(\mathbf{e}_i, \mathbf{e}_j)) - \frac{1}{|\mathcal{N}_j|} \sum_{n_j \in \mathcal{N}_j} \log \sigma(\gamma - f_r(\mathbf{e}_i, \mathbf{e}_{n_j})) \Bigg)$$

(5) Training Objective: $\mathcal{L} = \zeta_1 \mathcal{L}_{\text{hier}} + \zeta_2 \mathcal{L}_{\text{anc}} + \zeta_3 \mathcal{L}_{\text{link}}$

Experiments & Findings

- **Datasets**

Table 1: **Datasets statistics.** #Ent./#Rel: number of entities/relations. #Train/#Valid/#Test: number of triples contained in the training/validation/testing set.

Dataset	#Ent.	#Rel.	#Train	#Valid	#Test
FB15k-237	14,541	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000
PrimeKG	10,344	11	100,000	3,000	3,000

- **Metrics**

Mean Rank (MR):
- Measures the average rank of true entities.

Mean Reciprocal Rank (MRR):
- Averages the reciprocal ranks of true entities.

Hits@N:
- Measures the proportion of true entities in the top N predictions.

FB15K-237:
- A subset of Freebase, a large collaborative knowledge base focusing on common knowledge.

YAGO3-10:
- A subset of YAGO, a large knowledge base derived from multiple sources including Wikipedia, WordNet, and GeoNames.

PrimeKG:
- A biomedical KG integrates 20 biomedical resources, detailing 17,080 diseases through 4,050,249 relationships. In this study, we extract a subset of PrimeKG, which contains 106,000 triples.

Experiments & Findings

- **Main Results on Link Prediction**

(1) KG-FIT consistently and significantly outperforms state-of-the-art PLM-based and structure-based methods across all datasets and metrics.

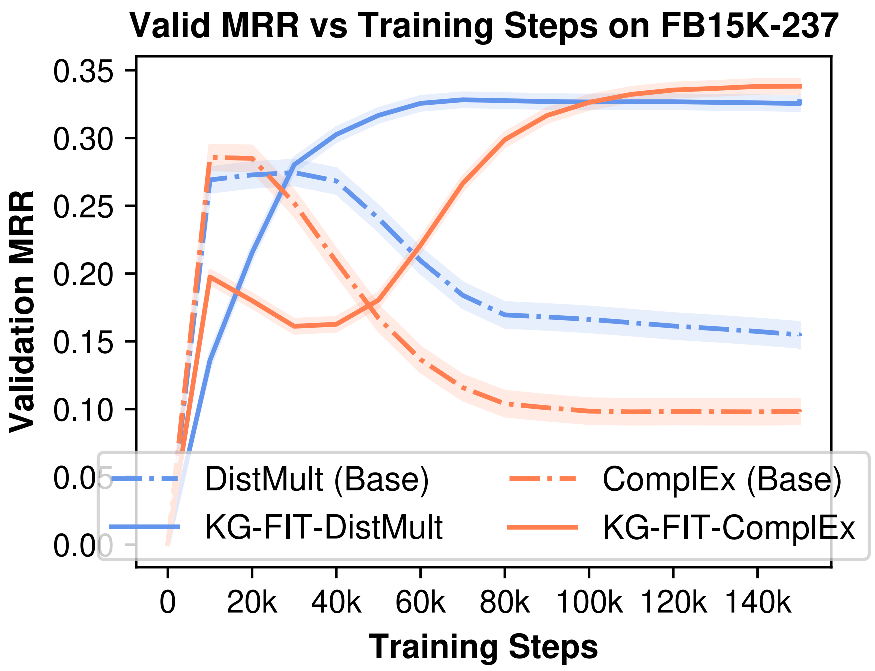
(2) With LLM-guided hierarchy refinement, KG-FIT achieves huge performance gains compared to the base models and KG-FIT with seed hierarchy.

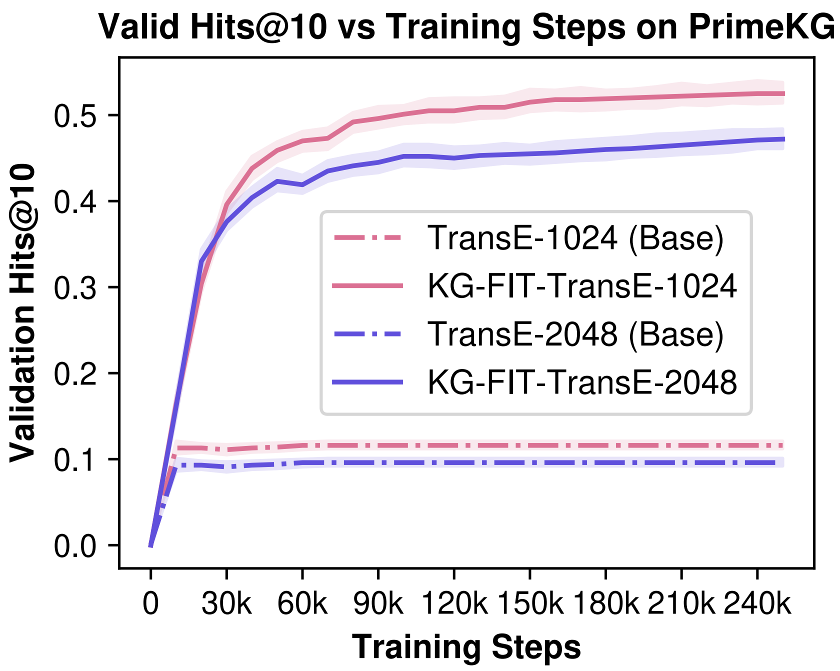
(3) KG-FIT is more effective for smaller KGs, e.g., more performance gains on PrimeKG (~ 0.1 million triples) than YAGO3-10 (~1 million triples).

			FB15K-237					YAGO3-10					PrimeKG				
PLM-based Embedding Methods																	
Model	PLM		MR	MRR	H@1	H@5	H@10	MR	MRR	H@1	H@5	H@10	MR	MRR	H@1	H@5	H@10
KG-BERT [22]*	BERT		153	.245	.158	–	.420	–	–	–	–	–	–	–	–	–	–
StAR [23]*	RoBERTa		**117**	.296	.205	–	.482	–	–	–	–	–	–	–	–	–	–
PKGC [28]	RoBERTa		184	.342	.236	.441	.525	1225	.501	.426	.596	.660	219	.485	.391	.565	.625
C-LMKE [26]*	BERT		141	.306	.218	–	.484	–	–	–	–	–	–	–	–	–	–
KGT5 [25]*	T5		–	.276	.210	–	.414	–	.426	.368	–	.528	–	–	–	–	–
KG-S2S [24]*	T5		–	.336	.257	–	.498	–	–	–	–	–	–	–	–	–	–
SimKGC [27]	BERT		–	.336	.249	–	.511	–	–	–	–	–	168	.527	.524	.679	.742
CSProm-KG [32]	BERT		–	.358	.269	–	.538	1145	.488	.451	.624	.675	157	.540	.492	.652	.745
LLM Emb. (zero-shot)	TE-3-S		2044	.023	.002	.035	.068	22741	.009	.000	.016	.024	5581	.000	.000	.000	.000
	TE-3-L		1818	.030	.004	.048	.085	18780	.015	.000	.019	.032	4297	.001	.000	.000	.000
Structure-based Embedding Methods																	
Model	Frame	$\mathcal{H}$	MR	MRR	H@1	H@5	H@10	MR	MRR	H@1	H@5	H@10	MR	MRR	H@1	H@5	H@10
TransE	Base [14]	—	233	.287	.192	.389	.478	1250	.500	.398	.626	.685	182	.048	.000	.043	.124
	KG-FIT	Seed	142	.345	.242	.457	.547	952	.520	.429	.638	.700	80	.298	.000	.315	.516
		LHR	122	**.362**	.264	.478	.568	**529**	.544	.463	.650	.705	69	.334	.000	.342	.536
DisMult	Base [15]	—	283	.260	.163	.349	.437	5501	.451	.365	.553	.615	174	.577	.475	.699	.782
	KG-FIT	Seed	184	.316	.198	.415	.512	963	.486	.413	.591	.673	107	.589	.495	.715	.799
		LHR	154	.331	.226	.433	.529	861	.527	.441	.636	.682	78	.617	.526	.747	.813
ComplEx	Base [16]	—	347	.252	.161	.344	.439	6681	.463	.384	.560	.612	202	.614	.522	.728	.789
	KG-FIT	Seed	201	.325	.223	.436	.523	997	.491	.422	.603	.669	94	.638	.548	.767	.823
		LHR	151	.344	.247	.458	.551	842	.544	.460	.646	.697	82	**.651**	**.566**	**.772**	**.835**
ConvE	Base [17]	—	341	.312	.224	.401	.508	1105	.529	.451	.619	.673	144	.516	.456	.645	.760
	KG-FIT	Seed	181	.318	.237	.411	.521	912	.535	.455	.628	.685	93	.627	.534	.757	.812
		LHR	177	.318	.241	.415	.525	885	.541	.461	.647	.695	72	.648	.547	.767	.824
TuckER	Base [18]	—	363	.320	.230	.417	.505	1110	.529	.454	.633	.690	171	.543	.442	.663	.737
	KG-FIT	Seed	175	.330	.241	.433	.521	874	.538	.458	.651	.703	77	.640	.542	.770	.805
		LHR	144	.349	.255	.448	.543	838	.545	**.466**	**.654**	.708	62	.648	.550	**.779**	.820
pRotatE	Base[19]	—	188	.310	.205	.399	.502	974	.477	.385	.573	.655	118	.491	.399	.593	.681
	KG-FIT	Seed	160	.355	.257	.461	.558	910	.525	.436	.622	.693	75	.635	.538	.745	.809
		LHR	**119**	**.371**	**.277**	**.483**	**.572**	829	**.550**	.464	.648	**.710**	69	**.649**	**.574**	**.779**	**.833**
RotatE	Base [19]	—	190	.333	.241	.428	.528	1620	.495	.402	.550	.670	57	.539	.447	.646	.727
	KG-FIT	Seed	141	.354	.261	.464	.555	790	.529	.440	.643	.708	**46**	.622	.517	.740	.805
		LHR	**120**	**.369**	**.274**	**.488**	**.570**	**744**	**.563**	**.475**	**.658**	**.722**	**34**	.645	.532	.758	.817
HAKE	Base [20]	—	184	.344	.247	.435	.538	1220	530	.431	.634	.681	95	.595	.515	.708	.760
	KG-FIT	Seed	162	.358	.268	.470	.563	854	.541	.455	.647	.703	82	.638	.540	.747	.808
		LHR	137	**.362**	**.275**	**.485**	**.572**	**810**	**.568**	**.474**	**.662**	**.718**	**42**	**.682**	**.605**	**.785**	**.835**

Experiments & Findings

- **KG-FIT can Overcome Overfitting and Underfitting Issues**





Experiments & Findings

- **Ablation Studies**

Table 3: **Ablation study for the proposed constraints.** *SA*, *HDM*, *ICS*, *CC* denote Semantic Anchoring, Hierarchical Distance Maintenance, Inter-level Cluster Separation, and Cluster Cohesion, respectively. We use TransE and HAKE as the base models for `KG-FIT` on FB15K-237 and YAGO3-10, respectively.

HDM	SA	ICS	CC	**FB15K-237** ($\texttt{KG-FIT}_{\text{TransE}}$) MRR	H@1	H@5	H@10	**YAGO3-10** ($\texttt{KG-FIT}_{\text{HAKE}}$) MRR	H@1	H@5	H@10
✓	✓	✓	✓	.362	.264	.478	.568	.568	.474	.662	.718
✗	✓	✓	✓	.345 (↓.017)	.248 (↓.016)	.454 (↓.024)	.542 (↓.026)	.558 (↓.010)	.467 (↓.007)	.654 (↓.008)	.709 (↓.009)
✗	✗	✓	✓	.335 (↓.027)	.241 (↓.023)	.444 (↓.034)	.533 (↓.035)	.545 (↓.023)	.452 (↓.022)	.640 (↓.022)	.695 (↓.023)
✗	✓	✗	✓	.343 (↓.019)	.244 (↓.020)	.449 (↓.029)	.538 (↓.030)	.544 (↓.024)	.453 (↓.021)	.643 (↓.019)	.691 (↓.027)
✗	✓	✓	✗	.332 (↓.030)	.239 (↓.025)	.437 (↓.041)	.529 (↓.039)	.558 (↓.010)	.465 (↓.009)	.656 (↓.006)	.711 (↓.007)
✗	✗	✗	✗	.287 (↓.075)	.192 (↓.072)	.389 (↓.089)	.478 (↓.090)	.530 (↓.038)	.431 (↓.043)	.634 (↓.028)	.681 (↓.037)

Findings:

- **Hierarchical Distance Maintenance** is crucial for both datasets. Its removal significantly degrades performance across all metrics, highlighting the necessity of preserving the hierarchical structure in the embedding space.
- **Semantic Anchoring** proves more critical for the denser YAGO3-10 graph, where each cluster contains more entities, making it harder to distinguish between them based solely on cluster cohesion. The sparser FB15K-237 dataset is less impacted by the absence of this constraint.
- Similar to the semantics anchoring, the removal of **Inter-level Cluster Separation** significantly affects the denser YAGO3-10 more than FB15K-237. Without this constraint, entities in YAGO3-10 may not be well-separated from other clusters, whereas FB15K-237 is less influenced.
- Removing **Cluster Cohesion** has a larger impact on the sparser FB15K-237 than on YAGO3-10. This difference suggests that sparse graphs rely more on the prior information provided by clusters, while denser graphs can learn this information more effectively from their abundant data.

Experiments & Findings

- **Ablation Studies**

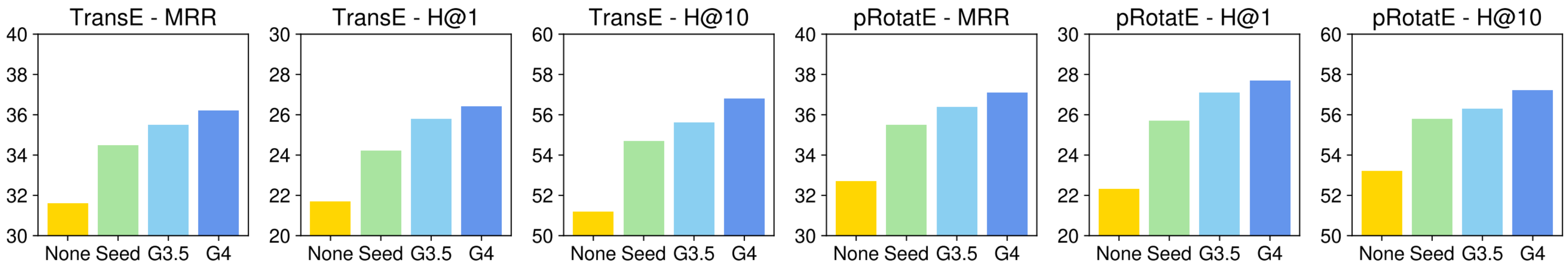


Figure 3: `KG-FIT` on FB15K-237 with different hierarchy types. *None* indicates no hierarchical information input. *Seed* denotes the seed hierarchy. *G3.5/G4* denotes the LHR hierarchy constructed by GPT-3.5/4o. LHR hierarchies outperform the seed hierarchy, with more advanced LLMs constructing higher-quality hierarchies.

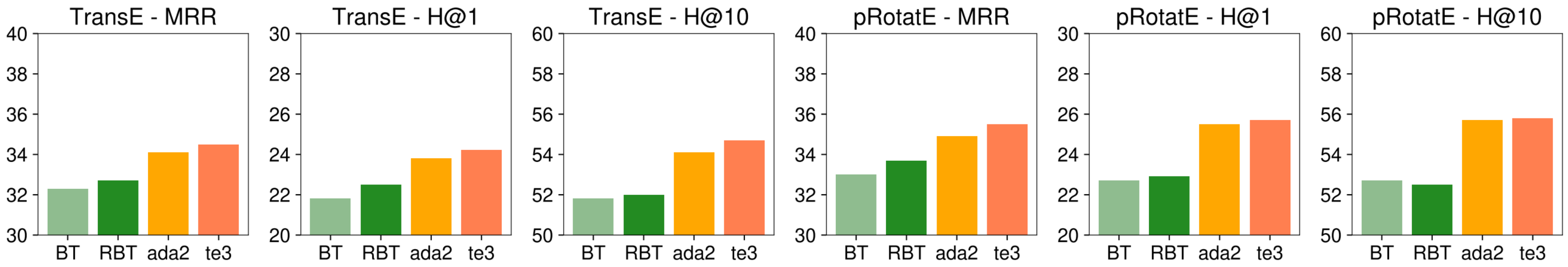


Figure 4: `KG-FIT` on FB15K-237 with different text embedding. *BT*, *RBT*, *ada2*, and *te3* are BERT, RoBERTa, text-embedding-ada-002, and text-embedding-3-large, respectively. Seed hierarchy is used for all settings. It is observed that pre-trained text embeddings from LLMs are substantially better than those from small PLMs.

Experiments & Findings

- **Efficiency Analysis**

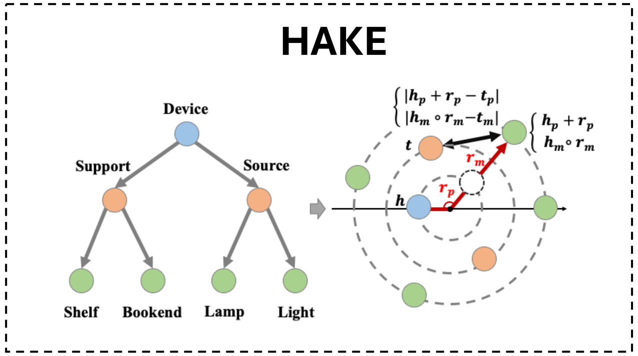
Model Efficiency on PrimeKG. T/Ep and Inf denote training time per epoch and inference time.

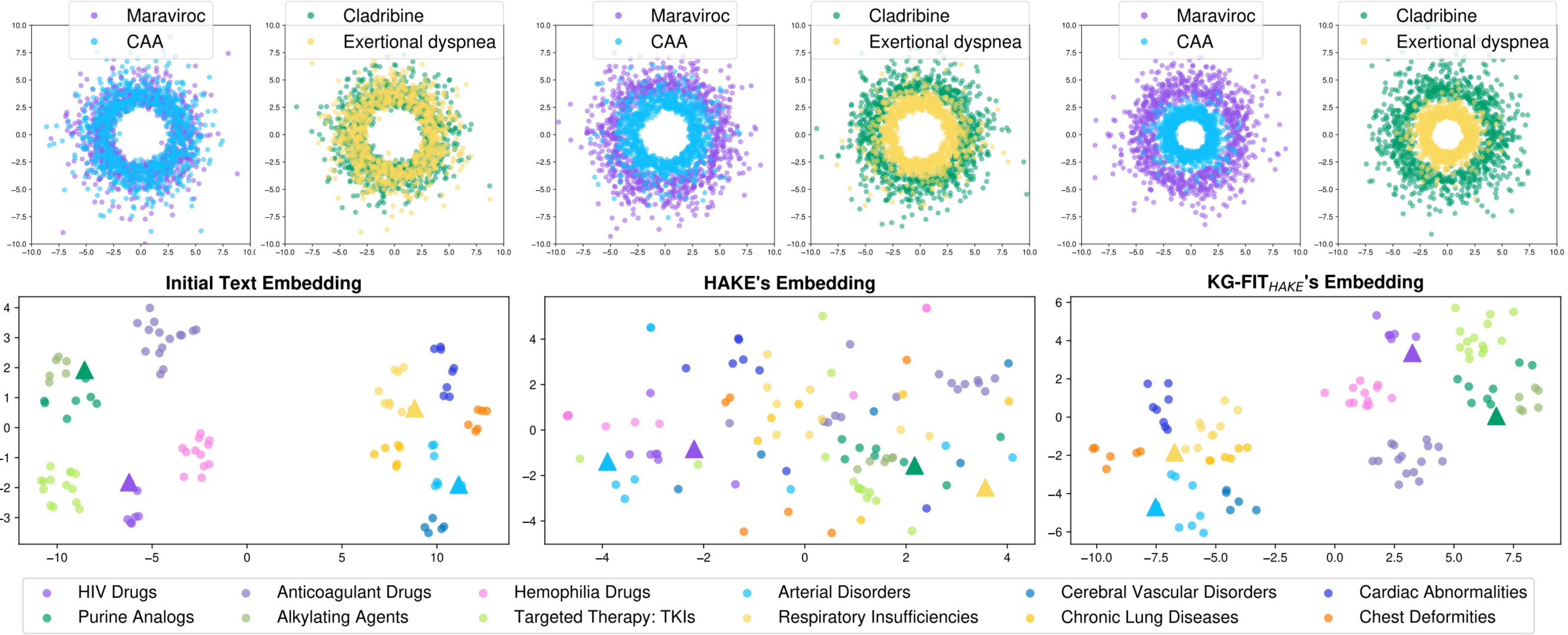
Method	LM	T/Ep	Inf
KG-BERT	RoBERTa	170m	2900m
PKGC	RoBERTa	190m	50m
TagReal	LUKE	190m	50m
StAR	RoBERTa	125m	30m
KG-S2S	T5	30m	110m
SimKGC	BERT	20m	0.5m
CSProm-KG	BERT	15m	0.2m
KG-FIT (ours)	Any LLM	1.2m	0.1m
Structure-based	—	0.2m	0.1m

Our KG-FIT achieves 12 times the training speed of the most efficient PLM-based baseline!

Experiments & Findings

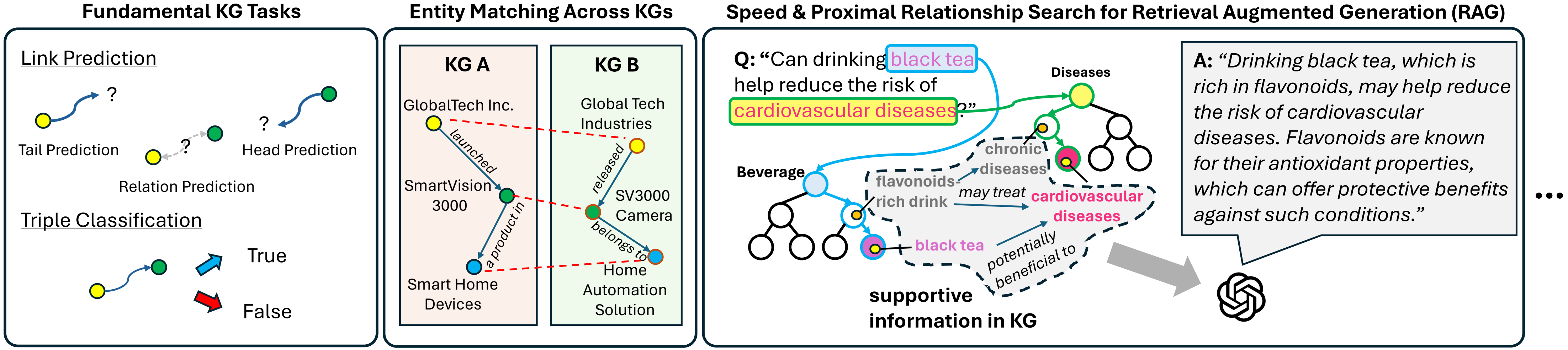
- **Visualization**





KG-FIT preserves both local and global semantics!

(Potential Downstream Applications)



(1) Traditional KG Tasks: link prediction, triple classification, relation extraction, entity resolution, KG-based question answering (KGQA), ...

(2) Entity Matching across KGs: leverage both local & global semantics to match the same entities with different labels in various KGs.

(3) Speed & Proximal Relationship Search for RAG: leverage the hierarchical structure of KG-FIT to efficiently and effectively search highly relevant triples related to the context.

Conclusions

We introduced KG-FIT, a novel framework that enhances knowledge graph (KG) embeddings by integrating open-world entity knowledge from Large Language Models (LLMs).

- KG-FIT effectively combines the knowledge from LLM and KG to preserve both global and local semantics, achieving state-of-the-art link prediction performance on benchmark datasets.
- It shows significant improvements in accuracy compared to the base models. Notably, KG-FIT can seamlessly integrate knowledge from any LLM, enabling it to evolve with ongoing advancements in language models.
- Future work will explore incorporating LLM-generated summaries of KG triples in training set as entity descriptions, further enhancing the embedding quality.

Code: https://github.com/pat-jj/KG-FIT

Thank you!

Contact Patrick (pj20@illinois.edu)
if you have further questions.