Text-Augmented Open Knowledge Graph Completion via Pre-Trained Language Models

Pengcheng Jiang , **Shivam Agarwal** , **Bowen Jin** , **Xuan Wang**[†] , **Jimeng Sun** and **Jiawei Han**

Department of Computer Science, University of Illinois at Urbana-Champaign

[†]Department of Computer Science, Virginia Tech

{pj20, shivama2, bowenj4, jimeng, hanj}@illinois.edu xuanw@vt.edu

Abstract

The mission of open knowledge graph (KG) completion is to draw new findings from known facts. Existing works that augment KG completion require either (1) factual triples to enlarge the graph reasoning space or (2) manually designed prompts to extract knowledge from a pre-trained language model (PLM), exhibiting limited performance and requiring expensive efforts from experts. To this end, we propose TAGREAL that automatically generates quality query prompts and retrieves support information from large text corpora to probe knowledge from PLM for KG completion. The results show that TAGREAL achieves state-of-the-art performance on two benchmark datasets. We find that TAGREAL has superb performance even with limited training data, outperforming existing embedding-based, graph-based, and PLM-based methods.

1 Introduction

A knowledge graph (KG) is a heterogeneous graph that encodes factual information in the form of entity-relation-entity triplets, where a *relation* connects a *head* entity and a *tail* entity (e.g., "*Miami-located_in-USA*") (Wang et al., 2017; Hogan et al., 2021). KG (Dai et al., 2020) plays a central role in many NLP applications, including question answering (Hao et al., 2017; Yasunaga et al., 2021), recommender systems (Zhou et al., 2020), and drug discovery (Zitnik et al., 2018). However, existing works (Wang et al., 2018; Hamilton et al., 2018) show that most large-scale KGs are incomplete and cannot fully cover the massive real-world knowledge. This challenge motivates KG completion, which aims to find one or more object entities given a subject entity and a relation (Lin et al., 2015). For example, in Figure 1, our goal is to predict the object entity with "*Detroit*" as the subject entity and "*contained_by*" as the relation.

However, existing KG completion approaches (Trouillon et al., 2016b; Das et al., 2018) have several limitations (Fu et al., 2019). First, their performance heavily depends on the density of the graph. They usually perform well on dense graphs with rich structural information but poorly on sparse graphs which are more common in real-world applications. Second, previous methods (e.g., Bordes et al. (2013)) assume a closed-world KG without considering vast open knowledge in the external resources. In fact, in many cases, a KG is usually associated with a rich text corpus (Bodenreider, 2004), which contains a vast amount of factual data not yet extracted. To overcome these challenges we investigate the task of open knowledge graph completion, where KG can be constructed using new facts from outside the KG. Recent text-enriched solutions (Fu et al., 2019) focus on using a predefined set of facts to enrich the knowledge graph. Nonetheless, the pre-defined set of facts is often noisy and constricted, that is, they do not provide sufficient information to efficiently update the KG.

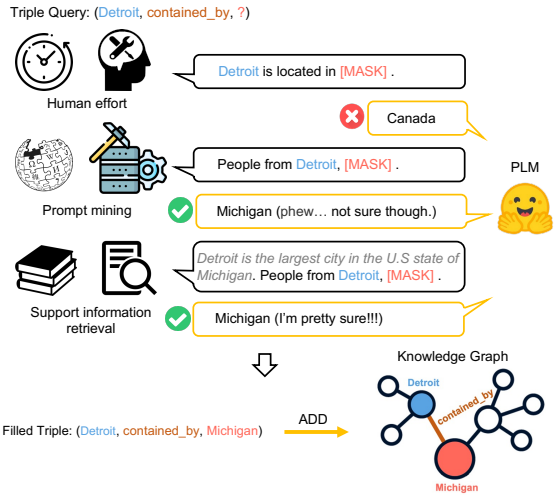

Figure 1: The quality of hand-crafted prompts can be limited, while prompt mining is a scalable alternative. Support information also helps PLM understand the purpose of prompts. In this example, Canada and Michigan are potentially valid options but given prompt mining and support information retrieval, the model becomes confident about Michigan as the answer here.

Pre-trained language models (PLMs) (Devlin et al., 2019; Liu et al., 2019a) have shown to be powerful in capturing factual knowledge implicitly from learning on massive unlabeled texts (Petroni et al., 2019b). Since PLMs are superb in text encoding, they can be utilized to facilitate knowledge graph completion with external text information. Recent knowledge graph completion methods (Shin et al., 2020; Lv et al., 2022) focus on using manually crafted prompts (e.g., "Detroit is located in [MASK]" in Figure 1) to query the PLMs for graph completion (e.g., "Michigan"). However, manually creating prompts can be expensive with limited quality (e.g., PLM gives a wrong answer "Canada" to the query with a handcrafted prompt, as shown in Figure 1).

Building on the above limitations of standard KG and the enormous power of PLMs (Devlin et al., 2019; Liu et al., 2019a), we aim to use PLMs for open knowledge graph completion. We propose an end-to-end framework that jointly exploits the implicit knowledge in PLMs and textual information in the corpus to perform knowledge graph completion (as shown in Figure 1). Unlike existing works (e.g., (Fu et al., 2019; Lv et al., 2022)), our method does not require a manually pre-defined set of facts and prompts, which is more general and easier to adapt to real-world applications.

Our contributions can be summarized as:

- We study the open KG completion problem that can be assisted by facts captured from PLMs. To this end, we propose a new framework **TagReal** that denotes **t**ext **a**u**g**mented open KG completion with **real**-world knowledge in PLMs.
- We develop prompt generation and information retrieval methods, which enable TagReal to automatically create high-quality prompts for PLM knowledge probing and search support information, making it more practical especially when PLMs lack some domain knowledge.
- Through extensive quantitative and qualitative experiments on real-world knowledge graphs such as Freebase[1] we show the applicability and advantages of our framework[2].

[1] https://github.com/thunlp/OpenNRE

[2] Our code is available at: https://github.com/pat-jj/TagReal

2 Related Work

2.1 KG Completion Methods

KG completion methods can be categorized into embedding-based and PLM-based methods. **Embedding-based methods** represent entities and relations as embedding vectors and maintain their semantic relations in the vector space. TransE (Bordes et al., 2013) vectorizes the head, the relation and the tail of triples into a Euclidean space. DistMult (Yang et al., 2014) converts all relation embeddings into diagonal matrices in bilinear models. RotatE (Sun et al., 2019) presents each relation embedding as a rotation in complex vector space from the head entity to the tail entity.

In recent years, researchers have realized that PLMs can serve as knowledge bases (Petroni et al., 2019a; Zhang et al., 2020; AlKhamissi et al., 2022). **PLM-based methods** for KG completion (Yao et al., 2019; Kim et al., 2020; Chang et al., 2021; Lv et al., 2022) start to gain attention. As a pioneer, KG-BERT (Yao et al., 2019) fine-tunes PLM with concatenated head, relation, and tail in each triple, outperforming the conventional embedding-based methods in link prediction tasks. Lv et al.(2022) present PKGC, which uses manually designed triple prompts and carefully selected support prompts as inputs to the PLM. Their result shows that PLMs could be used to substantially improve the KG completion performance, especially in the *open-world* (Shi and Weninger, 2018) setting. Compared to PKGC, our framework TagReal automatically generates prompts of higher quality without any domain expert knowledge. Furthermore, instead of pre-supposing the existence of support information, we search relevant textual information from the corpus with an information retrieval method to support the PLM knowledge probing.

2.2 Knowledge Probing using Prompts

LAMA (Petroni et al., 2019a) is the first framework for knowledge probing from PLMs. The prompts are manually created with a subject placeholder and an unfilled space for the object. For example, a triple query (*Miami*, *location*, ?) may have a prompt "Miami is located in `[MASK]`" where "`<subject>` is located in `[MASK]`" is the template for "location" relation. The training goal is to correctly fill `[MASK]` with PLM's prediction. Another work, BertNet (Hao et al., 2022), proposes an approach applying GPT-3 (Brown et al., 2020)

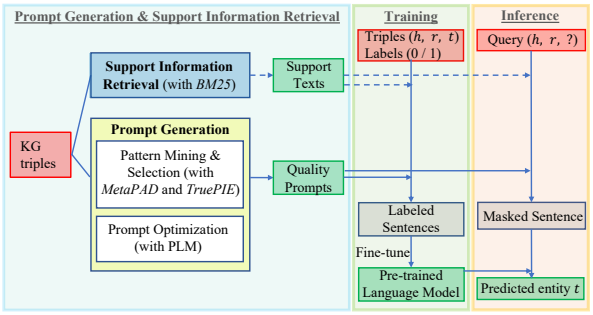

Figure 2: **TAGREAL Framework**. The input and output of each phase are highlighted by red and green, respectively. The dotted arrow indicates the optional process.

to automatically generate a weighted prompt ensemble with input entity pairs and a manual seed prompt. It then uses PLM again to search and select top-ranked entity pairs with the ensemble for KG completion.

2.3 Prompt Mining Methods

When there are several relations to interpret, manual prompt design is costly due to the requirement of domain expert knowledge. In addition, the prompt quality could not be ensured. Hence, *quality prompt mining* catches the interest of researchers. Jiang et al. 2020 propose an approach MINE which searches middle words or dependency paths between the given inputs and outputs in a large text corpus (e.g., Wikipedia). They also propose a reasonable approach to optimize the ensemble of the mined prompts by weighting prompt individuals regarding their performance on the PLM.

Before the emergence and widespread use of PLMs, textual pattern mining performed a similar function to find reliable patterns for information extraction. For instance, MetaPAD (Jiang et al., 2017) generates quality meta patterns by context-aware segmentation with the pattern quality function, and TruePIE (Li et al., 2018) proposes the concept of pattern embedding and a self-training framework, that discovers positive patterns automatically.

3 Methodology

We propose TAGREAL, a PLM-based framework to handle KG completion tasks. In contrast to the previous work, our framework does not rely on handcrafted prompts or pre-defined relevant facts. As shown in Figure 2, we automatically create appropriate prompts and search relevant support information, which are further utilized as templates to explore implicit knowledge from PLMs.

3.1 Problem Formulation

Knowledge graph completion is to add new triples (facts) to the existing triple set of a KG. There are two tasks to achieve this goal. The first is **triple classification**, which is a binary classification task to predict whether a triple (h, r, t) belongs to the KG, where h, r, t denote head entity, relation and tail entity respectively. The second task is **link prediction**, which targets on predicting either the tail entity t with a query $(h, r, ?)$ or the head entity h with a query $(?, r, t)$.

3.2 Prompt Generation

Previous studies (e.g., Jiang et al. (2020)) demonstrate that the accuracy of relational knowledge extracted from PLMs heavily relies on the quality of prompts used for querying. To this end, we develop a comprehensive approach for automatic quality prompt generation given triples in KG as the only input, as shown in Figure 3. We use textual pattern mining methods to mine quality patterns from large corpora as the prompts used for PLM knowledge probing. As far as we know, we are pioneers in using **textual pattern mining** methods for **LM prompt mining**. We believe in the applicability of this approach for the following reasons.

- Similar data sources. We apply pattern mining on large corpora (e.g., Wikipedia) which are the data sources where most of PLMs are pre-trained.
- Similar objectives. Textual pattern mining is to mine patterns to extract new information from large corpora; prompt mining is to mine prompts to probe implicit knowledge from PLMs.
- Similar performance criteria. The reliability of a pattern or a prompt is indicated by how many accurate facts it can extract from corpora/PLMs.

Sub-corpora mining is the first step that creates the data source for the pattern mining. Specifically, given a KG with a relation set $R = (r_1, r_2, ..., r_k)$, we first extract tuples T_{r_i} paired by head entities and tail entities for each relation $r_i \in R$ from the KG. For example, for the relation r_1: `/business/company/founder`, we extract all tuples like `<microsoft, bill_gates>` in this relation from the KG. For each tuple t_j, we then search sentences s_{t_j} containing both head and tail from a large corpus (e.g., Wikipedia) and other reliable sources, which is added to compose the sub-corpus C_{r_i}. We limit the size of each set to θ

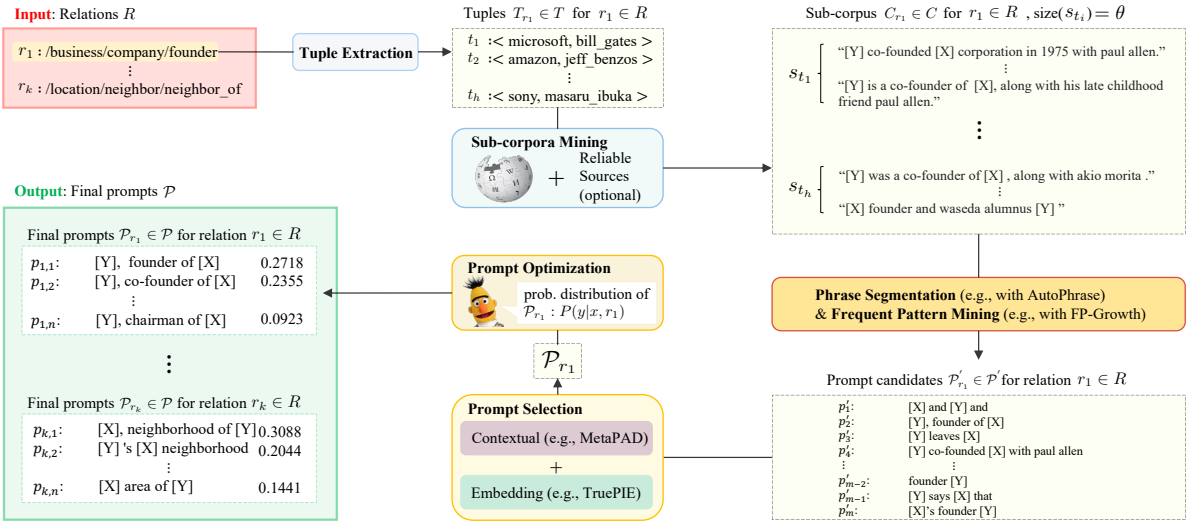

Figure 3: **Prompt generation process**. The solid lines connect the intermediate processes, and the arrows point to the intermediate/final results. Input and output are highlighted in **red** and **green** respectively. `[X]` and `[Y]` denote head and tail entities respectively.

for each tuple to mine more generic patterns for future applications.

Phrase segmentation and frequent pattern mining are applied to mine patterns from sub-corpora as prompt candidates. We use AutoPhrase (Shang et al., 2018) to segment corpora to more natural and unambiguous semantic phrases, and use FP-Growth algorithm (Han et al., 2000) to mine frequent appeared patterns to compose a candidate set $\mathcal{P}'_{r_i} = (p'_1, p'_2, ..., p'_m)$. The size of the set is large, as there are plenty of messy textual patterns.

Prompt selection. To select quality patterns from the candidate set, we apply two textual mining approaches: MetaPAD (Jiang et al., 2017) and TruePIE (Li et al., 2018). MetaPAD applies pattern quality function introducing several criteria of contextual features to estimate the reliability of a pattern. We explain why those features can also be adapted for LM prompt estimation: (1) *Frequency and concordance*: Since a PLM learns more contextual relations between frequent patterns and entities during the pre-training stage, a pattern occurs more frequently in the background corpus can probe more facts from the PLM. Similarly, if a pattern composed of highly associated sub-patterns appears frequently, it should be considered as a good one as the PLM would be familiar with the contextual relations among the sub-patterns. (2) *Informativeness*: A pattern with low informativeness (e.g., p'_1 in Figure 3) has the weak ability of PLM knowledge probing, as the relation between the subject or object entities cannot be well interpreted by it. (3) *Completeness*: The completeness of a pattern affects a lot to the PLM knowledge probing especially when any of the placeholders is missing (e.g., p'_{m-2} in Figure 3) so that PLM cannot even give an answer. (4) *Coverage*: A quality pattern should be able to probe accurate facts from PLM as many as possible. Therefore, patterns like p'_4 which only suit a few or only one case should have a low quality score. We then apply TruePIE on the prompts (patterns) selected by MetaPAD. TruePIE filters the prompts that have low cosine similarity with the positive samples (e.g., p'_3 and p'_{m-1} are filtered), which matters to the creation of prompt ensemble since we want the prompts in the ensemble to be semantically close to each other so that some poor-quality prompts would not significantly impact the prediction result by PLM. As a result, we create a more reliable prompt ensemble $\mathcal{P}_{r_i} = \{p_{i,1}, p_{i,2}, ..., p_{i,n}\}$ based on the averaged probabilities given by the prompts:

$$P(y|x, r_i) = \frac{1}{n} \sum_{j=1}^{n} P_{LM}(y|x, p_{i,j}), \quad (1)$$

where r_i is the i-th relation and $p_{i,j}$ is the j-th prompt in $\mathcal{P}_{r_i}$. Beyond prompt selection, a **prompt optimization** process is also employed. Pointed out by Jiang et al. 2020, some prompts in the ensemble are more reliable and ought to be weighted more. Thus, we change Equation 1 to:

$$P(y|x, r_i) = \sum_{j=1}^{n} w_{i,j} P_{LM}(y|x, p_{i,j}), \quad (2)$$

where $w_{i,j}$ is the weight of j-th prompt for i-th relation. In our setting, all weights $\{w_{1,1}, .., w_{k,n}\}$

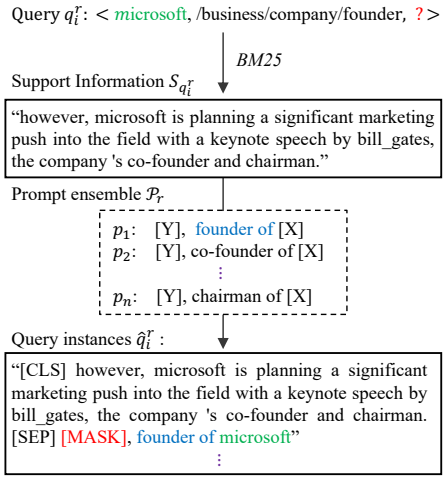

Figure 4: Support information retrieval.

are learned through PLM to optimize $P(y|x, r_i)$ for $r_i \in R$ ahead of the training process.

3.3 Support Information Retrieval

In addition to the prompt mining, we also attach some query-wise and triple-wise support text information to the prompt to help the PLMs understand the knowledge we want to probe as well as to aid in training triple classification ability. As seen in Figure 4, for the i-th query q_i^r in relation r, we use BM25 (Robertson et al., 1995) to retrieve highly ranked support texts with score greater than δ and length shorter than ϕ from the reliable corpus and randomly select one of them as the support information. To compose the input cloze $\hat{q}_i^r$ to the PLM, we concatenate the support text to each prompt in the optimized ensemble we obtained through previous steps, with the subject filled and the object masked. `[CLS]` and `[SEP]` are the tokens for sequence classification and support information-prompt separation accordingly.

In the training stage, we search texts using triples rather than queries, and the `[MASK]` would be filled by the object entities. It is worth noting that support text is optional in TAGREAL, and we leave it blank if no matching data is found.

3.4 Training

To train our model, we create negative triples in addition to the given positive triples following the idea introduced by PKGC (Lv et al., 2022), to handle the triple classification task. We create negative triples by replacing the head and tail in each positive triple with the "incorrect" entity that achieves high probability by the KGE model. We also create random negative samples by randomly replacing the heads and tails to enlarge the set of negative training/validation triples. The labeled training triples are assembled as $\mathcal{T} = \mathcal{T}^+ \cup (\mathcal{T}^-_{KGE} \cup \mathcal{T}^-_{RAND})$ where $\mathcal{T}^+$ is the positive set, $\mathcal{T}^-_{KGE}$ and $\mathcal{T}^-_{RAND}$ are two negative sets we created by embedding model-based and random approaches respectively. Then, we transform all training triples of each relation r into sentences with the prompt ensemble $\mathcal{P}_r$ and the triple-wise support information retrieved by BM25 (if there is any). At the training stage, the `[MASK]` is replaced by the object entity in each positive/negative triple. The query instances $\hat{q}_i^r$ are then used to fine-tune the PLM by updating its parameters. Cross-entropy loss (Lv et al., 2022) is applied for optimization:

$$\mathcal{L} = -\sum_{\tau \in \mathcal{T}} (y_\tau \log(c_\tau^1) + (1 - y_\tau) \frac{\log(c_\tau^0)}{M}), \quad (3)$$

where $c_\tau^0, c_\tau^1 \in [0, 1]$ are the softmax classification scores of the token `[CLS]` for the triple τ, y_τ is the ground truth label $(1/0)$ of the triple, and $M = (|\mathcal{T}^+|/|\mathcal{T}^-|)$ is the ratio between the number of positive and negative triples. After the PLM is fine-tuned with positive/negative triples in training set, it should have a better performance on classifying the triples in the dataset compared to a raw PLM. This capability would enable it to perform KG completion as well.

3.5 Inference

Given a query $(h, r, ?)$, we apply the query-wise support information that is relevant to the head entity h and relation r, as we presume that we are unaware of the tail entity (our prediction goal). Then, we make the corresponding query instances containing `[MASK]`, with both support information and prompt ensemble, as shown in Figure 4. To leverage the triple classification capability of the PLM on link prediction, we replace `[MASK]` in a query instance with each entity in the known entity set and rank their classification scores in descending order to create a 1-d vector as the prediction result for each query. This indicates that the lower-indexed entities in the vector are more likely to compose a positive triple with the input query. For prompt ensemble, we sum up the scores by entity index before ranking them. The detailed illustration is placed in Appendix E.

	Model	20%			50%			100%		
		Hits@5	Hits@10	MRR	Hits@5	Hits@10	MRR	Hits@5	Hits@10	MRR
KGE-based	TransE (Bordes et al., 2013)	29.13	32.67	15.80	41.54	45.74	25.82	42.53	46.77	29.86
	DisMult (Yang et al., 2014)	3.44	4.31	2.64	15.98	18.85	13.14	37.94	41.62	30.56
	ComplEx (Trouillon et al., 2016a)	4.32	5.48	3.16	15.00	17.73	12.21	35.42	38.85	28.59
	ConvE (Dettmers et al., 2018)	29.49	33.30	24.31	40.10	44.03	32.97	50.18	54.06	40.39
	TuckER (Balažević et al., 2019)	29.50	32.48	24.44	41.73	45.58	33.84	51.09	54.80	40.47
	RotatE (Sun et al., 2019)	15.91	18.32	12.65	35.48	39.42	28.92	**51.73**	55.27	**42.64**
Text&KGE-based	RC-Net (Xu et al., 2014)	13.48	15.37	13.26	14.87	16.54	14.63	14.69	16.34	14.41
	TransE+Line (Fu et al., 2019)	12.17	15.16	4.88	21.70	25.75	8.81	26.76	31.65	10.97
	JointNRE (Han et al., 2018)	16.93	20.74	11.39	26.96	31.54	21.24	42.02	47.33	32.68
RL-based	MINERVA (Das et al., 2017)	11.64	14.16	8.93	25.16	31.54	22.24	43.80	44.70	34.62
	CPL (Fu et al., 2019)	15.19	18.00	10.87	26.81	31.70	23.80	43.25	49.50	33.52
PLM-based	PKGC (Lv et al., 2022)	35.77	43.82	28.62	41.93	46.70	31.81	41.98	52.56	32.11
	TagReal (our method)	**45.59**	**51.34**	**35.41**	**48.98**	**55.64**	**38.03**	50.85	**60.64**	38.86

Table 1: **Performance comparison of KG completion on FB60K-NYT10 dataset**. Results are averaged values of ten independent runs of head/tail entity predictions. The highest score is highlighted in **bold.**

4 Experiment

4.1 Datasets and Compared Methods

Datasets. We use the datasets FB60K-NYT10 and UMLS-PubMed provided by Fu et al., where FB60K and UMLS are knowledge graphs and NYT10 and PubMed are corpora. FB60K-NYT10 contains more general relations (e.g., "nationality of perso") whereas UMLS-PubMed focuses on biomedical domain-specific relations (e.g., "gene mapped to diseas"). We apply the pre-processed dataset [3] (with training/validation/testing data size 8:1:1) to align the evaluation of our method with the baselines. Due to the imbalanced distribution and noise present in FB60K-NYT10 and UMLS-PubMed, 16 and 8 relations are selected for the performance evaluation, respectively. We place more details of the datasets in Appendix A.

Compared Methods. We compare our model TAGREAL with four categories of methods. For (1) traditional KG embedding-based methods, we evaluate **TransE** (Bordes et al., 2013), **DisMult** (Yang et al., 2014), **ComplEx** (Trouillon et al., 2016a), **ConvE** (Dettmers et al., 2018), **TuckER** (Balažević et al., 2019) and **RotatE** (Sun et al., 2019) where TuckER is a newly added model. For (2) joint text and graph embedding methods, we evaluate **RC-Net** (Xu et al., 2014), **TransE+LINE** (Fu et al., 2019) and **JointNRE** (Han et al., 2018). For (3) reinforcement learning (RL) based path-finding methods, we evaluate **MINERVA** (Das et al., 2017) and **CPL** (Fu et al., 2019). For (4) PLM-based methods, we evaluate **PKGC** (Lv et al., 2022) and our method **TAGREAL**. We keep the reported data of (2) and (3) by Fu et al.2019 while re-evaluating all models in (1) in different settings for more rigorous comparison (see Appendix I for details). PKGC in our setting can be viewed as TAGREAL with manual prompts and without support information.

[3] https://github.com/INK-USC/CPL#datasets

4.2 Experimental Setup

For FB60K-NYT10, we use LUKE (Yamada et al., 2020), a PLM pre-trained on more Wikipedia data with RoBERTa (Liu et al., 2019b). For UMLS-PubMed, we use SapBert (Liu et al., 2021) that pre-trained on both UMLS and PubMed with BERT (Devlin et al., 2019). For sub-corpora mining, we use Wikipedia with 6,458,670 document examples as the general corpus and NYT10/PubMed as the reliable sources, and we mine 500 sentences at maximum ($\theta = 500$) for each tuple. For the prompt selection, we apply MetaPAD with its default setting, and apply TruePIE with the infrequent pattern penalty, and thresholds for positive patterns and negative patterns reset to {0.5, 0.7, 0.3} respectively. For support information retrieval, we use BM25 to search relevant texts with $\delta = 0.9$ and $\phi = 100$ in the corpora NYT10/PubMed. We follow the same fine-tuning process as PKGC. We use TuckER as the KGE model to create negative triples, and we set $M = 30$ as the ratio of positive/negative triples. To compare with baselines, we test our model on training sets in the ratios of [20%, 50%, 100%] for FB60K-NYT10 and [20%, 40%, 70%, 100%] for UMLS-PubMed. The evaluation metrics are described in Appendix F.

5 Results

5.1 Performance Comparison

We show the performance comparison with the state-of-the-art methods in Tables 1 and 2. As one can observe, TAGREAL outperforms the existing

	Model	20%		40%		70%		100%	
		Hits@5	Hits@10	Hits@5	Hits@10	Hits@5	Hits@10	Hits@5	Hits@10
KGE-based	TransE (Bordes et al., 2013)	19.70	30.47	27.72	41.99	34.62	49.29	40.83	53.62
	DisMult (Yang et al., 2014)	19.02	28.35	28.28	40.48	32.66	47.01	39.53	53.82
	ComplEx (Trouillon et al., 2016a)	11.28	17.17	24.64	35.15	25.89	38.19	34.54	49.30
	ConvE (Dettmers et al., 2018)	20.45	30.72	27.90	42.49	30.67	45.91	29.85	45.68
	TuckER (Balažević et al., 2019)	19.94	30.82	25.79	41.00	26.48	42.48	30.22	45.33
	RotatE (Sun et al., 2019)	17.95	27.55	27.35	40.68	34.81	48.81	40.15	53.82
Text&KGE-based	RC-Net (Xu et al., 2014)	7.94	10.77	7.56	11.43	8.31	11.81	9.26	12.00
	TransE+Line (Fu et al., 2019)	23.63	31.85	24.86	38.58	25.43	34.88	22.31	33.65
	JointNRE (Han et al., 2018)	21.05	31.37	27.96	40.10	30.87	44.47	-	-
RL-based	MINERVA (Das et al., 2017)	11.55	19.87	24.65	35.71	35.80	46.26	57.63	63.83
	CPL (Fu et al., 2019)	15.32	24.22	26.96	38.03	37.23	47.60	58.10	**65.16**
PLM-based	PKGC (Lv et al., 2022)	31.08	43.49	41.34	52.44	47.39	55.52	55.05	59.43
	TagReal (our method)	**35.83**	**46.45**	**46.26**	**55.99**	**53.46**	**60.40**	**60.68**	62.88

Table 2: **Performance comparison of KG completion on UMLS-PubMed dataset**. Results are averaged values of ten independent runs of head/tail entity predictions. The highest score is highlighted in **bold.**

Condition	FB60K-NYT10			UMLS-PubMed			
	20%	50%	100%	20%	40%	70%	100%
man	(35.77, 43.82)	(41.93, 46.70)	(41.98, 52.56)	(31.08, 43.49)	(41.34, 52.44)	(47.39, 56.52)	(55.05, 59.43)
man+supp	(43.23, 47.74)	(47.10, 52.02)	(48.66, 57.46)	(32.95, 44.42)	(44.37, 54.96)	(51.98, 59.09)	(59.99, 61.23)
mine+supp	(44.54, 49.53)	(47.43, 53.87)	(49.03, 58.82)	(35.56, 45.33)	(45.35, 55.44)	(53.12, 59.65)	(60.27, 61.70)
optim+supp	(45.59, 51.34)	(48.98, 55.64)	(50.85, 60.64)	(35.83, 46.45)	(46.26, 55.99)	(53.46, 60.40)	(60.68, 62.88)

Table 3: **Ablation study on prompt and support information**. Data in brackets denotes Hits@5 (left) and Hits@10 (right). "man", "mine" and "optim" denote TAGREAL with manual prompts, mined prompt ensemble without optimization and optimized prompt ensemble, respectively. "supp" denotes application of support information.

works in most cases. Given dense training data, KGE-based methods (e.g., RotatE) and RL-based methods (e.g., CPL) can still achieve relatively high performance. However, when the training data is limited, these approaches suffer, whereas PLM-based methods (PKGC and TAGREAL) are not greatly impacted. Our approach performs noticeably better in such cases than the current non-PLM-based ones. This is because the KGE models cannot be trained effectively with inadequate data, and the RL-based path-finding models cannot recognize the underlying patterns given insufficient evidential and general paths in KG. On the other hand, PLMs already possess implicit information that can be used directly, and the negative effects of insufficient data in fine-tuning would be less harsh than in training from scratch. TAGREAL outperforms PKGC due to its ability to automatically mine quality prompts and retrieve support information in contrast to manual annotations which are often limited. Next, we analyze the impacts of support information and prompt generation on the performance of TAGREAL.

5.2 Model Analysis

We conduct an ablation study to verify the effectiveness of both automatically generated prompts and retrieved support information. The results are presented in Table 3, Figure 5 and 6.

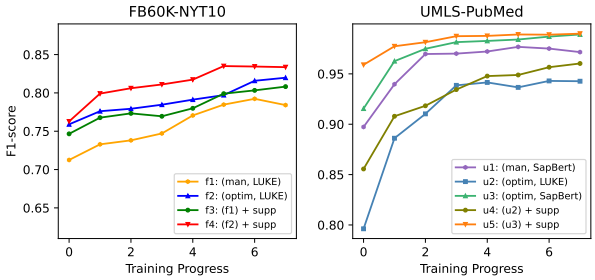

Figure 5: **Performance (F1-Score) variation of triple classification w.r.t training time.** "man" or "optim" means TAGREAL with manual prompts or optimized prompt ensemble. "supp" denotes support information.

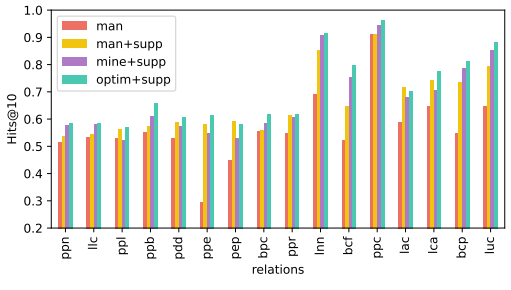

Figure 6: **Relation-wise KG completion performance (Hits@10) comparison on FB60K-NYT10**. Labels on the x-axis are the abbreviations of relations.

Support Information. As shown in Table 3, for FB60K-NYT10, support information helps improve Hits@5 and Hits@10 in ranges of [5.2%,

Query: (?, /location/location/contains, alba)

Manual Prompt	Optimized Prompt Ensemble	weights
[Y] is located in [X].	[Y], [X] .	0.10490836
	home in [Y], [X] .	0.23949857
	[Y] is in [X] .	0.24573646
	school in [Y], [X] .	0.32810964
	people from [Y], [X] .	0.34946583

Support Information (retrieved by BM25)

" **in alba** , **italy** 's truffle capital , **in the northwestern province of piedmont** , demand for the fungi has spawned a cottage industry of package tours , food festivals and a strip mall of truffle-themed shops . "

Predictions (Top10 in descending order of classification scores)

Man :	united_states_of_america, pennsylvania, france, lombardy, abruzzo, jamaica, **piedmont**, ivrea, massachusetts, iraq
Optim :	cuneo, **piedmont**, italy, sicily, lazio, texas, campania, northern_italy, scotland, calabria
Man + Supp :	sicily, italy, massachusetts, lazio, **piedmont**, united_states_of_america, abruzzo, tuscany, iraq, milan
Optim + Supp :	**piedmont**, cuneo, italy, northern_italy, canale, tuscany, campania, sicily, lazio, calabria

Figure 7: **Example of the link prediction with TAGREAL on FB60K-NYT10**. **Man** denotes manual prompt. **Optim** denotes optimized prompt ensemble. **Supp** denotes support information. The **ground truth tail entity**, **helpful information** and **optimized prompts** (darker for higher weights) are highlighted.

7.5%] and [3.8%, 5.3%], respectively. For UMLS-PubMed, it helps improve Hits@5 and Hits10 in ranges of [1.9%, 4.94%] and [0.9%, 3.6%], respectively. Although the overlap between UMLS and PubMed is higher than that between FB60K and NYT10 (Fu et al., 2019), the textual information in PubMed could not help as much as NYT10 since that: (1) SapBert already possesses adequate implicit knowledge on both UMLS and PubMed so that a large portion of additional support texts might be useless. The lines "u2", "u3", "u4" and "u5" in Figure 5 show that support information helps more when using LUKE as the PLM as it contains less domain-specific knowledge. It also infers that the support information could be generalized to any application, especially when fine-tuning a PLM is difficult in low-resource scenarios (Arase and Tsujii, 2019; mahabadi et al., 2021). (2) UMLS contains more queries with multiple correct answers than FB60K (see Appendix A), which means some queries are likely "misled" to another answer and thus not counted into the Hits@N metric.

Prompt Generation. Almost all of the relations, as shown in Figure 6, could be converted into better prompts by our prompt mining and optimization, albeit some of them might be marginally worse than manually created prompts due to the following fact. A few of the mined prompts, which are of lower quality than the manually created prompts, may significantly negatively affect the prediction score for the ensemble with equal weighting. Weighting based on PLM reduces such negative effects of the poor prompts for the optimized ensembles and enables them to outperform most handcrafted prompts. In addition, Table 3 shows the overall improvement for these three types of prompts, demonstrating that for both datasets, optimized ensembles outperform equally weighted ensembles, which in turn outperform manually created prompts. Moreover, by comparing line "f1" with line "f2", or line "u1" with line "u3" in Figure 5, we find a performance gap between PLM with manual prompts and with the optimized ensemble for triple classification, highlighting the effectiveness of our method.

5.3 Case Study

Figure 7 shows an example of using TAGREAL for link prediction with a query (?, */location/location/contains, alba*) where "*piedmont*" is the ground truth. By comparing the prediction results in different pairs, we find that both prompt generation and support information could enhance the KG completion performance. With the handcrafted prompt, the PLM simply lists out the terms that have some connections to the subject entity "*alba*" without being aware that we are trying to find the place it is located in. Differently, with the optimized prompt ensemble, the PLM lists entities that are highly relevant to our target, where "*cuneo*", "*italy*", "*northern_italy*" are correct real-world answers, indicating that our intention is well conveyed to the PLM. With the support information, the PLM increases the score of entities that are related to the keywords ("*italy*", "*piedmont*") in the text. Moreover, the optimized ensemble removes "*texas*" and "*scotland*" from the list and leaves only Italy-related locations. More examples are placed in Appendix H.

6 Conclusion and Future Works

In this study, we proposed a novel framework to exploit the implicit knowledge in PLM for open KG completion. Experimental results show that our method outperforms existing methods especially when the training data is limited. We showed that the optimized prompts with our approach outperform the handcrafted ones in PLM knowledge probing. The effectiveness of the support information retrieval to aid the prompting is also demonstrated. In the future, we may leverage QA model's power to retrieve more reliable support information. Another potential extension is to make our model more explainable by exploring path-finding tasks.

7 Limitations

Due to the nature of deep learning, our method is less explainable than path-finding-based KG completion methods (e.g., CPL), which provide a concrete reasoning path to the target entity. Composing the path with multiple queries might be an applicable strategy that is worthwhile to investigate in order to extend our work on the KG reasoning task.

For the link prediction task, we adapt the "recall and re-ranking" strategy from PKGC (Lv et al., 2022), which brings a trade-off between prediction efficiency and accuracy. We alleviate the issue by applying different hyper-parameters given different sizes of training data, which is discussed in detail in Appendix C.

As a common issue of existing KG completion models, the performance of our model also degrades when the input KG contains noisy data. The advantage of our approach in addressing this issue is that it can use both corpus-based textual information and implicit PLM knowledge to reduce noise.

8 Ethical Statements

In this study, we use two datasets FB60K-NYT10 and UMLS-PubMed, which include the knowledge graphs FB60K and UMLS as well as the text corpora NYT10 and PubMed. The data is all publicly available. Our task is knowledge graph completion, which is performed by finding missing facts given existing knowledge. This work is only relevant to NLP research and will not be put to improper use by ordinary people.

9 Acknowledgements

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004 and INCAS Program No. HR001121C0165, National Science Foundation IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897, and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under Award No. 2118329, and NSF Award SCH-2205289, SCH-2014438, IIS-2034479.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases. *arXiv preprint arXiv:2204.06031*.

Yuki Arase and Jun'ichi Tsujii. 2019. Transfer fine-tuning: A BERT case study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

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

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Ting-Yun Chang, Yang Liu, Karthik Gopalakrishnan, Behnam Hedayatnia, Pei Zhou, and Dilek Hakkani-Tur. 2021. Incorporating commonsense knowledge graph in pretrained models for social commonsense tasks. *arXiv preprint arXiv:2105.05457*.

Yuanfei Dai, Shiping Wang, Neal Naixue Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9:750.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *International Conference on Learning Representations*.

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. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Cong Fu, Tong Chen, Meng Qu, Woojeong Jin, and Xiang Ren. 2019. Collaborative policy learning for open knowledge graph reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2672–2681, Hong Kong, China. Association for Computational Linguistics.

Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. *Advances in neural information processing systems*, 31.

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Shibo Hao, Bowen Tan, Kaiwen Tang, Hengzhe Zhang, Eric P Xing, and Zhiting Hu. 2022. Bertnet: Harvesting knowledge graphs from pretrained language models. *arXiv preprint arXiv:2206.14268*.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, Vancouver, Canada. Association for Computational Linguistics.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge graphs. *ACM Comput. Surv.*, 54(4).

Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M Kaplan, Timothy P Hanratty, and Jiawei Han. 2017. Metapad: Meta pattern discovery from massive text corpora. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 877–886.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. 2020. Multi-task learning for knowledge graph completion with pre-trained language models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1737–1743.

Qi Li, Meng Jiang, Xikun Zhang, Meng Qu, Timothy P Hanratty, Jing Gao, and Jiawei Han. 2018. Truepie: Discovering reliable patterns in pattern-based information extraction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1675–1684.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2021. Self-alignment pretraining for biomedical entity representations. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4228–4238.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019a. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. *arXiv preprint arXiv:1811.04588*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3570–3581.

Rabeeh Karimi mahabadi, Yonatan Belinkov, and James Henderson. 2021. Variational information bottleneck for effective low-resource fine-tuning. In *International Conference on Learning Representations*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019a. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019b. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.

Baoxu Shi and Tim Weninger. 2018. Open-world knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016b. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA. PMLR.

Meng Wang, Ruijie Wang, Jun Liu, Yihe Chen, Lei Zhang, and Guilin Qi. 2018. Towards empty answers in sparql: approximating querying with rdf embedding. In *International semantic web conference*, pages 513–529. Springer.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29:2724–2743.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1219–1228.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*.

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*.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 535–546, Online. Association for Computational Linguistics.

Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower entity set expansion via language model probing. *arXiv preprint arXiv:2004.13897*.

Sijing Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive recommender system via knowledge graph-enhanced reinforcement learning. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466.

A Dataset Overview

We use the datasets **FB60K-NYT10** and **UMLS-PubMed** provided by (Fu et al., 2019) [4]. They take the following steps to split the data: (1) split the data of each KG (FB60K or UMLS) in the ratio of 8:1:1 for training/validation/testing data. (2) For training data, they keep all triples in any relations. (3) For validation/testing data, they only keep the triples in 16/8 relations they concern (see relations in Table 5). The processed data has {train: 268280, valid: 8765, test: 8918} for FB60K and {train: 2030841, valid: 8756, test: 8689} for UMLS. As for the corpora, there are 742536 and 5645558 documents in NYT10 and PubMed respectively.

	FB60K-NYT10	UMLS-PubMed
#query_tail	57279	12956
#query_head	23319	12956
#triples/#queries	2.22	6.81

Table 4: The number of queries and the ratio of triples/queries for FB60K-NYT10 and UMLS-PubMed

Sub-training-set splitting. To split the training data in the ratio of 20%/50% for FB60K-NYT10 or 20%/40%/70% for UMLS-PubMed, we use the same random seeds (55, 83, 5583) as Fu et al. used, and report the results in average.

Query-triple ratio. Within the relations that we focus on, we calculate the ratio of the triples by the queries (including both $(h, r, ?)$ and $(?, r, t)$) to indicate the number of correct answers a query may have in average. The result is given in Table 4. For UMLS-PubMed, as the relations are symmetric in pairs, the number of queries for head and tail predictions are the same. Table 5 presents the counting in a more detailed setting. Both tables show that there are more multi-answer queries in UMLS-PubMed than in FB60K-NYT10, which explains why the support information may not be as helpful in the former as it is in the latter, as revealed by Table 3 and discussed in Section 5.2.

B Textual Pattern Mining

The purpose of pattern mining is to find rules that describe particular patterns in the data. Information extraction is a common goal for pattern mining and prompt mining, where the former focuses on extracting facts from massive text corpora and the latter on extracting facts from PLMs. In this section, we use another example (Figure 8) to explain in detail how the textual pattern mining approaches like MetaPAD (Jiang et al., 2017) and TruePIE (Li et al., 2018) are implemented to mine quality prompts. In the example, given the relation `location/neighborhood/neighborhood_of` as the input, we first extract tuples (e.g., `<east new york, brooklyn>`) in the relation from the KG (i.e., FB60K). Then, we construct a sub-corpus by searching the sentences in a large corpus (e.g., Wikipedia) and the KG-related corpus (i.e. NYT10 for FB60). After the creation of sub-corpus, we apply phrase segmentation and frequent pattern mining to mine raw prompt candidates. Since the candidate set is noisy as some prompts with low completeness (e.g., `in lower [Y]`), low informativeness (e.g., `the [Y], [X]`) and low coverage (e.g., `[X], manhattan, [Y]`) are present, we use MetaPAD to handle the prompt filtering with its quality function introducing those contextual features. After the prompts have been processed by MetaPAD, we choose one of them to serve as a seed prompt (for example, `[X] neighborhood of [Y]`) so that other prompts can be compared to it by computing their cosine similarity. As the positive seed prompt is selected manually, we can tell that there is still room for future improvement.

C Re-ranking Recalls from KGE Model

Re-ranking framework. According to the inference process we present in Figure 9, we fill the placeholder (`[MASK]`) with each entity $(e_1, e_2, ..., e_n)$ in the entity set E. However, as mentioned by Lv et al.2022, the inference speed of PLM-based models is much slower than that of KGE models, which is a disadvantage of using PLM for KG completion. To address this issue, they use the recalls from KGE models, that is, using KGE models to run KG completion and select X top-ranked entities for each query as the entity set E. Then, they shuffle the set and re-rank those entities using the PLM-based model. In our work, we adapt this re-ranking framework to accelerate the inference and evaluation as our time complexity is Z times as large as PKGC (Lv et al., 2018) for each case where Z is the size of prompt ensemble. We use the recalls from TuckER (Balažević et al., 2019) for both datasets.

[4] https://github.com/INK-USC/CPL#datasets

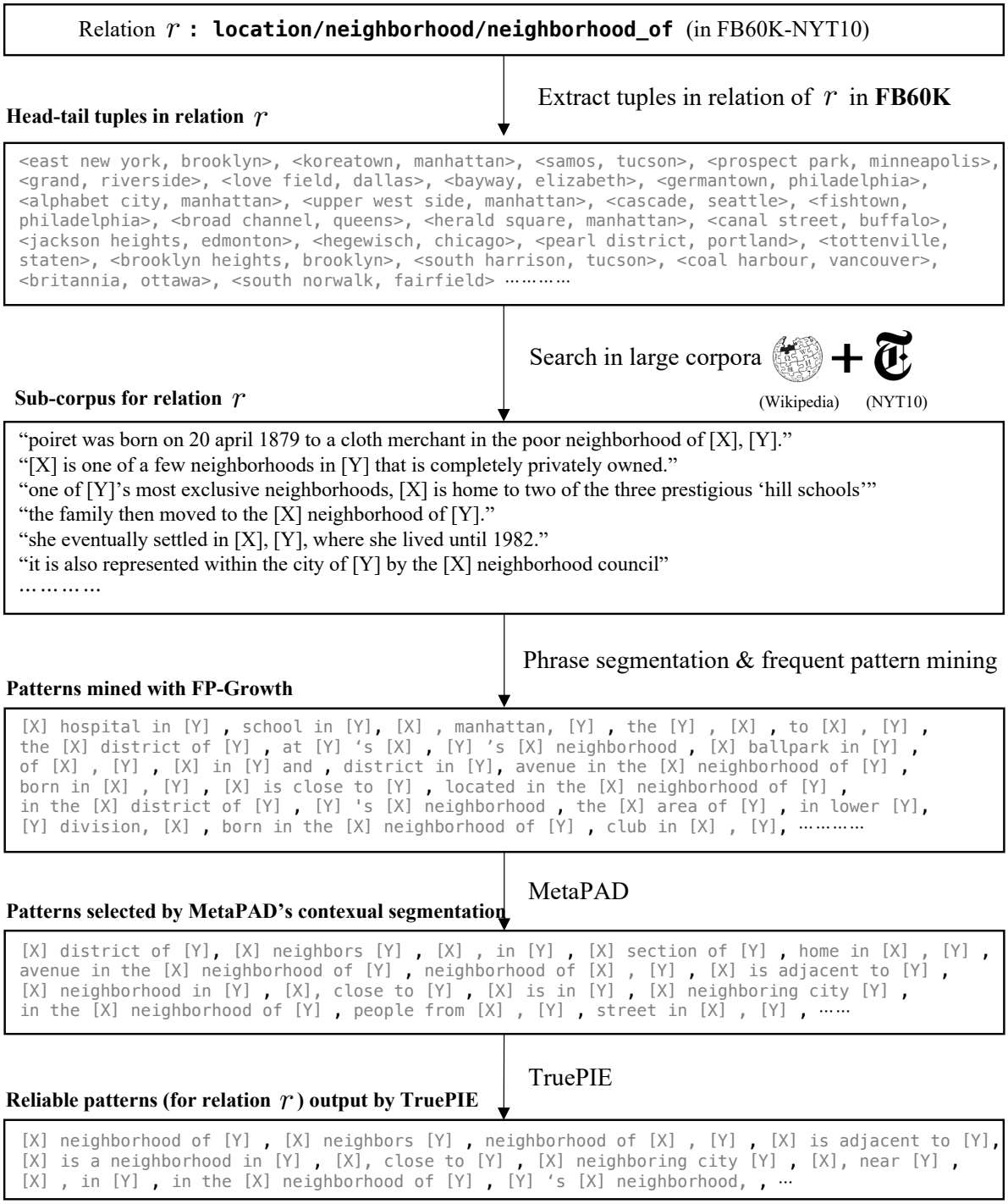

Figure 8: Example of textual pattern mining

relations	#triples(all)	#queries(all)	ratio(all)	#triples(test)	#queries(test)	ratio(test)
FB60K-NYT10						
/people/person/nationality	44186	20215	2.19	4438	2282	1.94
/location/location/contains	42306	11971	3.53	4244	2373	1.79
/people/person/place_lived	29160	12760	2.29	3094	2066	1.50
/people/person/place_of_birth	28108	16341	1.72	2882	2063	1.40
/people/deceased_person/place_of_death	6882	4349	1.58	678	518	1.31
/people/person/ethnicity	5956	2944	2.02	574	305	1.88
/people/ethnicity/people	5956	2944	2.02	592	318	1.86
/business/person/company	4334	2370	1.83	450	379	1.19
/people/person/religion	3580	1688	2.12	300	175	1.71
/location/neighborhood/neighborhood_of	1275	547	2.33	130	91	1.43
/business/company/founders	904	709	1.28	94	87	1.08
/people/person/children	821	711	1.15	56	56	1.00
/location/administrative_division/country	829	498	1.66	88	72	1.22
/location/country/administrative_divisions	829	498	1.66	102	79	1.29
/business/company/place_founded	754	548	1.38	80	73	1.10
/location/us_county/county_seat	264	262	1.01	32	32	1.00
UMLS-PubMed						
may_be_treated_by	71424	7703	9.27	7020	3118	2.25
may_treat	71424	7703	9.27	6956	3091	2.25
may_be_prevented_by	10052	3232	3.11	1014	584	1.74
may_prevent	10052	3232	3.11	1034	586	1.76
gene_mapped_to_disease	6164	1732	3.56	596	331	1.80
disease_mapped_to_gene	6164	1732	3.56	652	357	1.82
gene_associated_with_disease	536	289	1.85	58	49	1.18
disease_has_associated_gene	536	289	1.85	48	41	1.17

Table 5: Number of triples (#triples) and queries (#queries) in relations for FB60K-NYT10 and UMLS-PubMed. Triples/queries for both head prediction and tail prediction are counted. "all" and "test" denote the whole dataset and testing data respectively.

Limitations. Nonetheless, implementing the re-ranking framework has a trade-off between efficiency and Hits@N performance. When the training data is large (e.g., 100%), the KGE model could be well trained so that the ground truth entity e_{gt} is more likely to be contained in the top X ranked ones. However, when the training data is limited (e.g., 20%), the trained KGE model could not perform well on link prediction, as shown in Table 1 and 2. In such a case, there is a probability that e_{gt} is not among the top X entities if we keep using the same X regardless of the size of the training data. To alleviate this side effect, we test and select different values of the hyper-parameter X for different sizes of training data, as presented in Table 6.

Daatset	20%	40%	50%	70%	100%
FB60K-NYT10	70	-	40	-	20
UMLS-PubMed	50	50	-	30	30

Table 6: Best X for different training sizes

To check how much space there is for improvement, we manually add the ground truth entity into the recalls (we should not do this for the evaluation of TAGREAL as we suppose the object entity is unknown) and test the performance of TAGREAL on UMLS-PubMed. The result is shown in Table 7. By comparing this data with Table 3 for UMLS-PubMed, we find that changing the values of X could not perfectly address the issue. We leave the improvement as one of our major future works.

Condition	20%	40%	70%	100%
man	(44.83, 60.99)	(50.81, 67.69)	(52.98, 69.21)	(60.19, 72.58)
mine	(44.98, 61.56)	(52.81, 68.66)	(56.30, 70.20)	(61.29, 74.76)
optim	(45.71, 63.61)	(54.22, 69.03)	(58.18, 71.05)	(63.67, 75.55)

Table 7: **Link prediction of TAGREAL on UMLS-PubMed with ground truth added to the KGE recalls**. Data in brackets are Hits@5 (left) and Hits@10 (right).

D Computing Infrastructure & Budget

We trained and evaluated TAGREAL on 7 NVIDIA RTX A6000 running in parallel as we support multi-GPU computing. Training TAGREAL to a good performance took about 22 and 14 hours on the entire FB60K-NYT10 dataset (with LUKE (Yamada et al., 2020)) and the entire UMLS-PubMed dataset (with SapBert (Liu et al., 2021)) respectively. The training time is proportional to the size (ratio) of the training data. The evaluation took about 12 minutes for FB60K-NYT10 with LUKE when hyper-parameter $X = 20$, and 16 minutes for UMLS-PubMed with SapBert when $X = 30$. The evaluation time is proportional to X, which

explains why we applied the re-ranking framework (Appendix C) to improve the prediction efficiency.

E Link Prediction with Ensemble

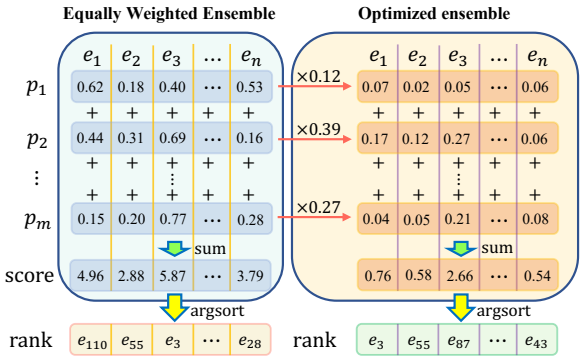

Figure 9: Prediction with ensemble. $e_1, e_2, ..., e_n$ denote the indices of entities. $p_1, p_2, ..., p_m$ denote the prompts in the ensemble.

For the link prediction with equally-weighted or optimized ensembles, we apply the method shown in Figure 9. Specifically, for each sentence with `[MASK]` filled with an entity e_i, we calculate its classification score with the fine-tuned PLM. For each query, we get an $m \times n$ matrix where m is the number of prompts in the ensemble, n is the number of entities in the entity set (which is X if the re-ranking framework is applied). For an ensemble that is equally weighted, we simply sum the scores of each entity obtained from the different prompts, whereas for an optimized ensemble, we multiply the weighting of the prompts by the scores before the addition. After sorting the vector in size of $1 \times n$ in descending order, we can get the ranking of entities as the result of the link prediction.

F Evaluation Metrics

Following previous KG completion works (Fu et al., 2019; Lv et al., 2022), we use Hits@N and Mean Reciprocal Rank (MRR) as our evaluation metrics. As mentioned in Section 3.5, the prediction of each query $(h, r, ?)$ is a 1-d vector of indices of entities in descending order regarding their scores. Specifically, for a query q_i, We record the rank of the object entity t as R_i, then we have:

$$Hits@N = \sum_{i=1}^{Q} \frac{\mathrm{R}_{i,in}}{Q} \text{ and } \mathrm{R}_{i,in} = \begin{cases} 0, \mathrm{R}_i > N \\ 1, \mathrm{R}_i \leq N, \end{cases} \tag{4}$$

$$MRR = \sum_{i=1}^{Q} \frac{1}{Q\mathrm{R}_i}, \tag{5}$$

where Q is the number of queries in evaluation.

G Code Interpretation

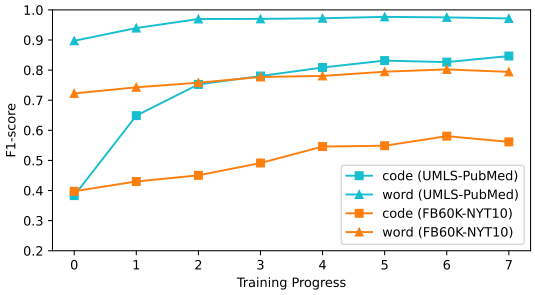

Figure 10: **Performance variation of triple classification w.r.t training time.** "code" and "word" denote the representation of KG entities.

To exploit the power of PLM, we need to map the code (entity_id) in KG/corpus into the words (Figure 10 shows the performance difference of PLM between using word and using code). For FB60K-NYT10, we use the mapping provided by JointNRE (Han et al., 2018) [5], which covers the translation for all entities. For UMLS-PubMed, we jointly use three mappings [6,7,8] which cover 97.22% of all entities.

H Case Study

In addition to Figure 7, we show more examples applying TAGREAL on link prediction in Figure 11. We can see that the predictions with optimized prompt ensemble outperform those with manual prompts in all the cases, and even outperforms predictions with manual prompts and support information in some cases. In all these examples, the support information aids the PLM knowledge probing in different ways. For the first example, we believe that the PLM captures the words "*brother james_murray*" and "*his wife jenny*", and realize that we are talking about the Scottish lexicographer "*james_murray*" but not the American comedian with the same name, based on our survey. For the second example, the PLM probably captures "*glycemic control*" which is highly relevant to the disease "*hyperglycemia*". For the third example, the term "*antiemetic*" (the drug against vomiting) is likely captured so that the answer "*vomiting*" could be correctly predicted. Hence, it is not necessary for the support information to include the object

[5] https://github.com/thunlp/JointNRE
[6] https://evs.nci.nih.gov/ftp1/NCI_Thesaurus/
[7] https://www.ncbi.nlm.nih.gov/books/NBK9685/
[8] https://bioportal.bioontology.org/ontologies/VANDF

Example 1: **Query:** (james_murray, /people/person/nationality, ?) **Dataset: FB60K-NYT10**

Manual Prompt	Optimized Prompt Ensemble	weights
The nationality of [X] is [Y] .	[X]'s nationality is [Y] .	0.17283396
	[X] was born in [Y] .	0.28863361
	[X] is from [Y] .	0.31687216
	[X] is from [Y] (country) .	0.33789972
	[X] born in [Y] .	0.35120992

Support Information (retrieved by BM25)

"survived by **brother james_murray** and **his wife jenny** of sidney , australia , fourteen nieces and nephews and thirteen great nieces and nephews in usa , scotland , england and australia ."

Predictions (Top10 in descending order of classification scores)

Man :	united_states_of_america, united_kingdom, republic_of_ireland, england, south_africa, sweden, wales, **scotland**, pakistan, canada
Optim :	united_states_of_america, **scotland**, england, republic_of_ireland, united_kingdom, pakistan, wales, germany, switzerland, sweden
Man + Supp :	australia, england, **scotland**, germany, united_states_of_america, south_africa, canada, sweden, wales, belgium
Optim + Supp :	**scotland**, australia, england, united_states_of_america, great_britain, wales, germany, belgium, republic_of_ireland, south_africa

Example 2: **Query:** (insulin_degludec, may_be_treated_by, ?) **Dataset: UMLS-PubMed**

Manual Prompt	Optimized Prompt Ensemble	weights
[Y] may be treated by [X].	[X] is a therapy for [Y] .	0.10776438
	a cure for [Y] is [X] .	0.11534966
	[X], treatment to [Y] .	0.13597418
	[Y] treated by [X] .	0.28779642
	[X] treats [Y] .	0.49708182

Support Information (retrieved by BM25)

"this reportedly allows for less pharmacodynamic variability and within-subject variability than currently available insulin analogs , and a duration of action that is over 24 hours . the lack of proof of carcinogenicity with insulin_degludec is yet another factor that would be taken into consideration when choosing the optimal basal insulin for a diabetic individual . a formulation of insulin insulin_degludec with insulin aspart , **insulin_degludec** 70% / aspart 30% , may permit improved flexibly of dosing without compromising **glycemic control** or safety ."

Predictions (Top10 in descending order of classification scores)

Man :	type_2_diabetes_mellitus, diabetic_ketoacidosis, type_1_diabetes_mellitus, **hyperglycemia**, hyperkalemia, abnorm_drug_ind, inj_myocardial_reperfusion, obesity, defic_dis, hiv_infect
Optim :	type_2_diabetes_mellitus, **hyperglycemia**, type_1_diabetes_mellitus, hyperkalemia, abnorm_drug_ind, defic_dis, obesity, diabetic_ketoacidosis, pain_postop, atrial_fibrillation
Man + Supp :	type_2_diabetes_mellitus, type_1_diabetes_mellitus, **hyperglycemia**, abnorm_drug_ind, aids, diabetic_ketoacidosis, hyperkalemia, defic_dis, obesity, blood_pois
Optim + Supp :	**hyperglycemia**, aids, hyperkalemia, abnorm_drug_ind, type_2_diabetes_mellitus, type_1_diabetes_mellitus, blood_pois, diabetic_ketoacidosis, delirium, leg_dermatoses

Example 3: **Query:** (aprepitant, may_prevent, ?) **Dataset: UMLS-PubMed**

Manual Prompt	Optimized Prompt Ensemble	weights
[Y] may be prevented by [X] .	[X] that prevents [Y]	0.09786690
	[X] in prevention of [Y] .	0.23915859
	[Y] prevented by [X] .	0.32334973
	[X] prevents [Y] .	0.38601556

Support Information (retrieved by BM25)

"the prophylactic and therapeutic efficacy of **antiemetic** used for rinv may be enhanced by adding **aprepitant** before starting radiotherapy in high risk cases as in ours ."

Predictions (Top10 in descending order of classification scores)

Man :	perennial_allergic_rhinitis, hiccups, motion_sickness, **vomiting**, nasal_polyp, nausea, lv_dysfunction, status_epilepticus, pain, postop_compl
Optim :	nasal_polyp, **vomiting**, status_epilepticus, motion_sickness, perennial_allergic_rhinitis, nausea, hiccups, asthma, lv_dysfunction, pain
Man + Supp :	**vomiting**, nausea, postop_compl, motion_sickness, withdrawal_syndrome, status_epilepticus, pain, anxiety_disorder, hiccups, reye_s_syndrome
Optim + Supp :	**vomiting**, motion_sickness, nausea, status_epilepticus, withdrawal_syndrome, reye_s_syndrome, pain, postop_compl, hiccups, psychotic_disorder

Figure 11: **Examples of the link prediction with TAGREAL**. **Man** denotes manual prompt. **Optim** denotes optimized prompt ensemble. **Supp** denotes support information. The **ground truth tail entity** , **helpful information** and **optimized prompts** (darker for higher weights) are highlighted.

entity itself, and including only some text relevant to it could also be helpful.

I Re-evaluation of Knowledge Graph Embedding Models

We find that the performance of some KGE models was underestimated by Fu et al.2019 due to the low embedding dimension set for entity and relation. According to our re-evaluation (Table 8), many of these models could perform much better with higher dimension, and we report their best performance in Table 1 and 2 based on our experiments. For the previously evaluated models, we use the same code [9,10,11] as Fu et al. used to ensure the fairness of the comparison. For TuckER (Balažević et al., 2019), we use the code provided by the author [12]. Same as Fu et al., to make the comparison more rigorous, we do not apply the filtered setting (Bordes et al., 2013; Sun et al., 2019) of the Hits@N evaluation to all the models including TAGREAL.

J Code for Paper

The source code of TAGREAL is attached as supplemental material to the submission of this paper.

[9] https://github.com/thunlp/OpenKE
[10] https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding
[11] https://github.com/TimDettmers/ConvE
[12] https://github.com/ibalazevic/TuckER

FB60K-NYT10	Fu et al.'s setting (edim, rdim, filter)	Fu et al.'s setting Ratio: (Hits@5, Hits@10, MRR)	Our setting (edim, rdim, filter)	Our setting Ratio: (Hits@5, Hits@10, MRR)
TransE (Bordes et al., 2013)	(100, 100, n/a)	20%: (15.12, 18.83, 12.57) 50%: (19.38, 23.20, 13.36) 100%: (38.53, 43.38, 29.90)	**(600, 600, n/a)**	20%: (29.13, 32.67, 15.80) 50%: (41.54, 45.74. 25.82) 100%: (42.53, 46.77, 29.86)
DisMult (Yang et al., 2014)	(100, 100, n/a)	20%: (1.42, 2.55, 1.05) 50%: (15.23, 19.05, 12.36) 100%: (32.11, 35.88, 24.95)	**(600, 600, n/a)**	20%: (3.44, 4.31, 2.64) 50%: (15.98, 18.85, 13.14) 100%: (37.94, 41.62, 30.56)
ComplEx (Trouillon et al., 2016a)	(100, 100, n/a)	20%: (4.22, 5.97, 3.44) 50%: (19.10, 23.08, 12.99) 100%: (32.91, 34.62, 24.67)	**(600, 600, n/a)**	20%: (4.32, 5.48, 3.16) 50%: (15.00, 17.73, 12.21) 100%: (35.42, 38.85, 28.59)
ConvE (Dettmers et al., 2018)	(200, 200, n/a)	20%: (20.60, 26.90, 11.96) 50%: (24.39, 30.59, 18.51) 100%: (33.02, 39.78, 24.45)	(100, 100, n/a)	20%: (22.91, 26.29, 19.48) 50%: (26.52, 29.84, 22.67) 100%: (31.71, 35.66, 25.58)
			(600, 600, n/a)	20%: (29.49, 33.30, 24.31) 50%: (40.10, 44.03, 32.97) 100%: (50.18, 54.06, 40.39)
TuckER (Balažević et al., 2019)	-	-	(100, 100, n/a)	20%: (20.04, 23.02, 16.27) 50%: (24.04, 27.88, 20.21) 100%: (34.54, 38.77, 28.19)
			(600, 600, n/a)	20%: (29.50, 32.48, 24.44) 50%: (41.73, 45.58, 33.84) 100%: (51.09, 54.80, 40.47)
RotatE (Sun et al., 2019)	(200, 100, ?)	20%: (9.25, 11.83, 8.04) 50%: (25.96, 31.63, 23.34) 100%: (58.32, 60.66, 51.85)	(100, 50, n/a)	20%: (1.34, 2.13, 1.08) 50%: (2.54, 4.03, 1.91) 100%: (5.42, 7.87, 2.09)
			(200, 100, n/a)	20%: (7.47, 9.14, 5.81) 50%: (21.68, 25.45, 17.35) 100%: (47.96, 52.02, 39.17)
			(600, 300, n/a)	20%: (15.91, 18.32, 12.65) 50%: (35.48, 39.42, 28.92) 100%: (51.73, 55.27, 42.64)

UMLS-PubMed	Fu et al.'s setting (edim, rdim, filter)	Fu et al.'s setting Ratio: (Hits@5, Hits@10)	Our setting (edim, rdim, filter)	Our setting Ratio: (Hits@5, Hits@10)
TransE (Bordes et al., 2013)	(100, 100, n/a)	20%: (7.12, 11.17) 40%: (26.86, 38.08) 70%: (31.32, 43.58) 100%: (32.28, 45.52)	**(600, 600, n/a)**	20%: (19.70, 30.47) 40%: (27.72, 41.99) 70%: (34.62, 49.29) 100%: (40.83, 53.62)
DisMult (Yang et al., 2014)	(100, 100, n/a)	20%: (14.66, 21.16) 40%: (26.90, 38.35) 70%: (31.65, 44.98) 100%: (32.80, 47.50)	**(600, 600, n/a)**	20%: (19.02, 28.35) 40%: (28.28, 40.48) 70%: (32.66, 47.01) 100%: (39.53, 53.82)
ComplEx (Trouillon et al., 2016a)	(100, 100, n/a)	20%: (18.18, 19.58) 40%: (23.77, 34.15) 70%: (30.04, 43.60) 100%: (31.84, 46.57)	**(600, 600, n/a)**	20%: (11.28, 17.17) 40%: (24.64, 35.15) 70%: (25.89, 38.19) 100%: (34.54, 49.30)
ConvE (Dettmers et al., 2018)	(200, 200, n/a)	20%: (20.51, 30.11) 40%: (28.01, 42.04) 70%: (31.01, 45.81) 100%: (30.35, 45.35)	**(200, 200, n/a)**	20%: (20.45, 30.72) 40%: (27.90, 42.49) 70%: (30.67, 45.91) 100%: (29.85, 45.68)
			(600, 600, n/a)	20%: (20.26, 30.29) 40%: (26.85, 41.57) 70%: (26.97, 42.44) 100%: (25.43, 41.58)
TuckER (Balažević et al., 2019)	-	-	(100, 100, n/a)	20%: (5.13, 8.06) 40%: (20.48, 31.20) 70%: (29.66, 42.89) 100%: (31.56, 44.72)
			(256, 256, n/a)	20%: (19.94, 30.82) 40%: (25.79, 41.00) 70%: (26.48, 42.48) 100%: (30.22, 45.33)
			(600, 600, n/a)	20%: (18.84, 27.94) 40%: (24.57, 37.79) 70%: (25.50, 41.32) 100%: (24.41, 40.56)
RotatE (Sun et al., 2019)	(200, 100, n/a)	20%: (4.03, 6.50) 40%: (8.65, 13.21) 70%: (14.90, 21.67) 100%: (20.75, 27.82)	**(600, 300, n/a)**	20%: (17.95, 27.55) 40%: (27.35, 40.68) 70%: (34.81, 48.81) 100%: (40.15, 53.82)

Table 8: **Performance of knowledge graph embedding models on FB60K-NYT10 and UMLS-PubMed**. "edim" and "rdim" denotes the embedding size of entity and relation respectively. "filter" denotes the application of the filtered setting. Best setting for each model is highlighted in **bold**.