# KGLM: Integrating Knowledge Graph Structure in Language Models for Link Prediction

**Jason Youn**[1,2,3] and **Ilias Tagkopoulos** [1,2,3]
[1] Department of Computer Science, University of California, Davis, CA 95616, USA.
[2] Genome Center, University of California, Davis, CA 95616, USA.
[3] USDA/NSF AI Institute for Next Generation Food Systems (AIFS),
University of California, Davis, CA 95616, USA.
{jyoun, itagkopoulos}@ucdavis.edu

## Abstract

The ability of knowledge graphs to represent complex relationships at scale has led to their adoption for various needs including knowledge representation, question-answering, and recommendation systems. Knowledge graphs are often incomplete in the information they represent, necessitating the need for knowledge graph completion tasks. Pre-trained and fine-tuned language models have shown promise in these tasks although these models ignore the intrinsic information encoded in the knowledge graph, namely the entity and relation types. In this work, we propose the Knowledge Graph Language Model (KGLM) architecture, where we introduce a new entity/relation embedding layer that learns to differentiate distinctive entity and relation types, therefore allowing the model to learn the structure of the knowledge graph. In this work, we show that further pre-training the language models with this additional embedding layer using the triples extracted from the knowledge graph, followed by the standard fine-tuning phase sets a new state-of-the-art performance for the link prediction task on the benchmark datasets.

## 1 Introduction

Knowledge graph (KG) is defined as a directed, multi-relational graph where entities (nodes) are connected with one or more relations (edges) (Wang et al., 2017). It is represented with a set of triples, where a triple consists of (*head entity*, *relation*, *tail entity*) or ($h$, $r$, $t$) for short, for example (*Bill Gates*, *founderOf*, *Microsoft*) as shown in Figure 1. Due to their effectiveness in identifying patterns among data and gaining insights into the mechanisms of action, associations, and testable hypotheses (Li and Chen, 2014; Silvescu et al., 2012), both manually curated KGs like DBpedia (Auer et al., 2007), WordNet (Miller, 1998), KIDS (Youn et al., 2022), and CARD (Alcock et al., 2020), and automatically curated ones like FreeBase (Bollacker et al., 2008), Knowledge Vault (Dong et al., 2014), and NELL (Carlson et al., 2010) exist. However, these KGs often suffer from incompleteness. For example, 71% of the people in FreeBase have no known place of birth (West et al., 2014). To address this issue, knowledge graph completion (KGC) methods aim at connecting the missing links in the KG.

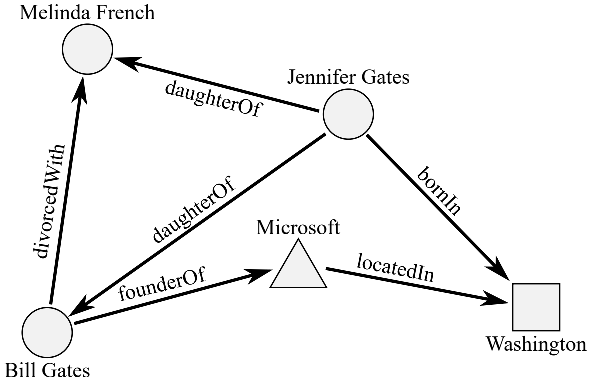

Figure 1: Sample knowledge graph with 6 triples. The graph contains three unique entity types (circle for person, triangle for company, and square for location) and 5 unique relation types or 10 if considering both the forward and inverse relations. The task of the knowledge graph completion is to complete the missing links in the graph, e.g., (*Bill Gates*, *bornIn?*, *Washington*) using the existing knowledge graph.

Graph feature models like path ranking algorithm (PRA) (Lao and Cohen, 2010; Lao et al., 2011) attempt to solve the KGC tasks by extracting the features from the observed edges over the KG to predict the existence of a new edge (Nickel et al., 2015). For example, the existence of the path *Jennifer Gates* $\xrightarrow{daughterOf}$ *Melinda French* $\xleftarrow{divorcedWith}$ *Bill Gates* in Figure 1 can be used as a clue to infer the triple (*Jennifer Gates*, *daughterOf*, *Bill Gates*). Other popular types of models are latent feature models such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), and RotatE (Sun et al., 2019) where entities and relations are converted into a latent space using embeddings.

*Proceedings of the The 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023)*, pages 217–224
July 13-14, 2023 ©2023 Association for Computational Linguistics

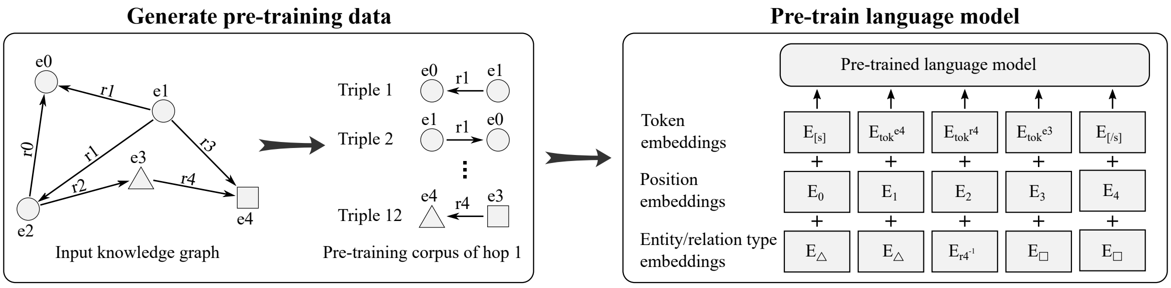

Figure 2: Proposed pre-training approach of the KGLM. First, both the forward and inverse triples are extracted from the knowledge graph to serve as the pre-training corpus. We then continue pre-training the language model, RoBERTa in our case, using the masked language model training objective, with an additional entity/relation-type embedding layer. The entity/relation-type embedding scheme shown here corresponds to the KGLM$_{\text{GER}}$, the most fine-grained version where both the entity and relation types are considered unique. Note that the inverse relation denoted by $^{-1}$ is different from its forward counterpart. For demonstration purposes, we assume all entities and relations to have a single token.

TransE, a representative latent feature model, models the relationship between the entities by interpreting them as a translational operation. That is, the model optimizes the embeddings by enforcing the vector operation of head entity embedding $\boldsymbol{h}$ plus the relation embedding $\boldsymbol{r}$ to be close to the tail entity embedding $\boldsymbol{t}$ for a given fact in the KG, or simply $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$.

Recently, pre-trained language models like BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019) have shown state-of-the-art performance in all of the natural language processing (NLP) tasks. As a natural extension, models like KG-BERT (Yao et al., 2019) and BERTRL (Zha et al., 2021) that utilize these pre-trained language models by treating a triple in the KG as a textual sequence, e.g., (*Bill Gates*, *founderOf*, *Microsoft*) as '*Bill Gates founder of Microsoft*', have also shown state-of-the-art results on the downstream KGC tasks. Although such *textual encoding* (Wang et al., 2021) models are generalizable to unseen entities or relations (Zha et al., 2021), they still fail to learn the intrinsic structure of the KG as the models are only trained on the textual sequence. To solve this issue, a hybrid approach like StAR (Wang et al., 2021) has recently been proposed to take advantage of both latent feature models and textual encoding models by enforcing a translation-based graph embedding approach to train the textual encoders. Yet, current textual encoding models still suffer from entity ambiguation problems (Cucerzan, 2007) where an entity *Apple*, for example, can refer to either the company Apple Inc. or the fruit. Moreover, there are no ways to distinguish forward relation (*Jennifer Gates*, *daughterOf*, *Melinda French*) from inverse relation (*Melinda French*, *daughterOf*$^{-1}$, *Jennifer Gates*).

In this paper, we propose the Knowledge Graph Language Model (KGLM) (Figure 2), a simple yet effective language model pre-training approach that learns from both the textual and structural information of the knowledge graph. We continue pre-training the language model that has already been pre-trained on other large natural language corpora using the corpus generated by converting the triples in the knowledge graphs as textual sequences, while enforcing the model to better understand the underlying graph structure and by adding an additional entity/relation-type embedding layer. Testing our model on the WN18RR dataset for the link prediction task shows that our model improved the mean rank by 21.2% compared to the previous state-of-the-art method (51 vs. 40.18, respectively). All code and instructions on how to reproduce the results are available online.[1]

## 2 Background

**Link Prediction.** The link prediction (LP) task, one of the commonly researched knowledge graph completion tasks, attempts to predict the missing head entity ($h$) or tail entity ($t$) of a triple ($h$, $r$, $t$) given a KG $G = (E, R)$, where $\{h, t\} \in E$ is the set of all entities and $r \in R$ is the set of all relations. Specifically, given a single test positive triple ($h$, $r$, $t$), its corresponding link prediction test dataset can be constructed by corrupting either the head or the tail entity in the filtered setting (Bordes et al., 2013) as

[1] https://github.com/ibpa/KGLM

$$\mathcal{D}_{LP}^{(h,r,t)} = \{(h,r,t') \mid t' \in (E-\{h,t\}) \wedge (h,r,t') \notin \mathcal{D}\} \cup\{(h',r,t) \mid h' \in (E-\{h,t\}) \wedge (h',r,t) \notin \mathcal{D}\} \cup\{(h,r,t)\}, \tag{1}$$

where $\mathcal{D} = \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{test}$ is the complete dataset. Evaluation of the link prediction task is measured with mean rank (MR), mean reciprocal rank (MRR), and hits@N (Rossi et al., 2021). MR is defined as

$$MR = \frac{\sum\limits_{(h,r,t)\in\mathcal{D}_{test}} rank((h,r,t) \mid \mathcal{D}_{LP}^{(h,r,t)})}{|\mathcal{D}_{test}|}, \tag{2}$$

where $rank(\cdot|\cdot)$ is the rank of the positive triple among its corrupted versions and $|\mathcal{D}_{test}|$ is the number of positive test triples. MRR is the same as MR except that the reciprocal rank $1/rank(\cdot|\cdot)$ is used. Hits@N is defined as

$$hits@N = \frac{\sum\limits_{(h,r,t)\in\mathcal{D}_{test}} \begin{cases} 1, \text{ if } rank((h,r,t) \mid \mathcal{D}_{LP}^{(h,r,t)}) < N \\ 0, \; otherwise \end{cases}}{|\mathcal{D}_{test}|}, \tag{3}$$

where $N \in \{1, 3, 10\}$ is commonly reported. Higher MRR and hits@N values are better, whereas, for MR, lower values denote higher performance.

## 3 Proposed Approach

In this work, we propose to continue pre-training, instead of pre-training from scratch, the language model RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019) that has already been trained on English-language corpora of varying sizes and domains, using both the forward and inverse knowledge graph textual sequences (Figure 2). Following the convention used in the KG-BERT and StAR (see Appendix A), we use a textual representation of a given triple, e.g., (*Bill Gates*, *founderOf*, *Microsoft*) as '*Bill Gates founder of Microsoft*', to generate the pre-training corpus. However, instead of extracting only the forward triple as done in the previous work, we extract both the forward and inverse versions of the triple, e.g., (*Jennifer Gates*, *daughterOf*, *Bill Gates*) and (*Bill Gates*, *daughterOf*$^{-1}$, *Jennifer Gates*), where the $^{-1}$ notation denotes the inverse direction of the corresponding relation.

Table 1: Statistics of the benchmark knowledge graphs used for link prediction.

| Dataset | # ent | # rel | # train | # val | # test |
|---|---|---|---|---|---|
| WN18RR | 40,943 | 11 | 86,835 | 3,034 | 3,134 |
| FB15k-237 | 14,951 | 237 | 272,115 | 17,535 | 20,466 |
| UMLS | 135 | 46 | 5,216 | 652 | 661 |

To enforce the model to learn the knowledge graph structure, we introduce a new embedding layer *entity/relation-type embedding* (ER-type embedding) in addition to the pre-existing token and position embeddings of RoBERTa as shown in Figure 2. This additional layer aims to embed the tokens in the input sequence with its corresponding entity/relation-type, where the set of entities $E$ in the knowledge graph can have $t_E$ different entity types depending on the schema of the knowledge graph, (e.g., $t_E = 3$ for person, company, and location in Figure 1). Note that many knowledge graphs do not specify the entity types, in which case $t_E = 1$. For the set of relations $R$, there exist $t_R = 2n_R$, where $n_R$ is the number of unique relations in the knowledge graph and the multiplier of 2 comes from forward and inverse directions (e.g., $t_R = 10$ for the sample knowledge graph in Figure 1).

In this work, we propose three different variations of ER-type embeddings. KGLM$_{\text{Base}}$ is the simplified version where all entities are assigned a single entity type and relations are assigned either forward or inverse relation type regardless of their unique relation types, resulting in a total of 3 ER-type embeddings. The KGLM$_{\text{GR}}$ is a version with granular relation types with $t_R + 1$ ER-type embeddings. The KGLM$_{\text{GER}}$ is the most granular version where we utilize all $t_E + t_R$ ER-type embeddings. In other words, all entity types as well as all relation types including both directions are considered.

To be specific, we convert a triple ($h$, $r$, $t$) to a sequence of tokens $w^{(h,r,t)} = \langle \texttt{[s]} w_a^h w_b^r w_c^t \texttt{[/s]} : a \in \{1..|h|\} \,\&\, b \in \{1..|r|\} \,\&\, c \in \{1..|t|\}\rangle \in \mathbb{R}^{(|h|+|r|+|t|+2)}$, where `[s]` and `[/s]` are special tokens denoting beginning and end of the sequence, respectively. The input to the RoBERTa model is then constructed by adding the ER-type embedding $\mathbf{t}^{(h,r,t)}$ and the $\mathbf{p}^{(h,r,t)}$ position embeddings to the

Table 2: Link prediction results on the benchmark datasets WN18RR, FB15k-237, and UMLS. Bold numbers denote the best performance for a given metric and class of models. Underlined numbers denote the best performance for a given metric regardless of the model type. Note that we do not report KGLM$_{\text{GER}}$ performance since the tested datasets do not specify entity types in their schema.

| | WN18RR | | | | | FB15k-237 | | | | | UMLS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | Hits @1 | Hits @3 | Hits @10 | MR | MRR | Hits @1 | Hits @3 | Hits @10 | MR | MRR | Hits@10 | MR |
| *Model type: Not based on language models* | | | | | | | | | | | | |
| TransE | .043 | .441 | .532 | 2300 | .243 | .198 | .376 | .441 | 323 | .279 | .989 | 1.84 |
| TransH | .053 | .463 | .540 | 2126 | .279 | .306 | .450 | .613 | 219 | .320 | - | - |
| DistMult | .412 | .470 | .504 | 7000 | .444 | .199 | .301 | .446 | 512 | .281 | .846 | 5.52 |
| ComplEx | .409 | .469 | .530 | 7882 | .449 | .194 | .297 | .450 | 546 | .278 | .967 | 2.59 |
| ConvE | .390 | .430 | .480 | 5277 | .46 | .239 | .350 | .491 | 246 | .316 | **.990** | **1.51** |
| RotatE | .428 | .492 | .571 | 3340 | .476 | .241 | .375 | .533 | 177 | .338 | - | - |
| GAAT | .424 | **.525** | **.604** | **1270** | .467 | **.512** | **.572** | **.650** | 187 | **.547** | - | - |
| LineaRE | **.453** | .509 | .578 | 1644 | **.495** | .264 | .391 | .545 | 155 | .357 | - | - |
| QuatDE | .438 | .509 | .586 | 1977 | .489 | .268 | .400 | .563 | **90** | .365 | - | - |
| *Model type: Based on language models* | | | | | | | | | | | | |
| KG-BERT | .041 | .302 | .524 | 97 | .216 | - | - | .420 | 153 | - | .990 | 1.47 |
| StAR | .243 | .491 | .709 | 51 | .401 | **.205** | **.322** | **.482** | **117** | **.296** | .991 | 1.49 |
| **KGLM$_{\text{Base}}$** | .305 | .518 | .730 | 47.97 | .445 | - | - | - | - | - | - | - |
| **KGLM$_{\text{GR}}$** | **.330** | **.538** | **.741** | **40.18** | **.467** | .200 | .314 | .468 | 125.9 | .289 | **.995** | **1.19** |

$\mathbf{w}^{(h,r,t)}$ token embeddings, as

$$\mathbf{X}^{(h,r,t)} = \mathbf{w}^{(h,r,t)} + \mathbf{p}^{(h,r,t)} + \mathbf{t}^{(h,r,t)}. \quad (4)$$

Unlike the segment embeddings in the KG-BERT and StAR that were used to mark the input tokens with either the entity ($\mathbf{s}_e$) or relation ($\mathbf{s}_r$), the ER-type embedding now replaces its functionality. Finally, we pre-train the model using the masked language model (MLM) training objective (Liu et al., 2019).

For fine-tuning, we extend the idea of how the KG-BERT scores a triple (see Equation 6 in Appendix A) to take advantage of the ER-type embeddings learned in our pre-training stage. For a given target triple, we calculate the weighted average score of both directions as

$$\begin{aligned} score_{KGLM}(h, r, t) = \alpha \text{SeqCls}(\mathbf{X}^{(h,r,t)}) + \\ (1 - \alpha)\text{SeqCls}(\mathbf{X}^{(t,r^{-1},h)}), \end{aligned} \quad (5)$$

where SeqCls(·) is a RoBERTa model transformer with a sequence classification head on top of the pooled output (last layer hidden-state of the [CLS] token followed by dense layer and $\tanh$ activation function), $(t, r^{-1}, h)$ denotes the inverse version of $(h, r, t)$, and $0 \leq \alpha \leq 1$ denotes the weight used for balancing the scores from forward and inverse scores. For example, $\alpha = 1.0$ considers only the forward direction score.

## 4 Experiments and Results

### 4.1 Datasets

We tested our proposed method on three benchmark datasets WN18RR, FB15k-237, and UMLS as shown in Table 1. WN18RR (Dettmers et al., 2018) is derived from WordNet (Miller, 1998), a large English lexical database of semantic relationships between words, FB15k-237 (Toutanova and Chen, 2015) is extracted from Freebase (Bollacker et al., 2008), a large community-drive KG of general facts about the world, and UMLS contains biomedical relationships. WN18RR and FB15k-237 are subsets of WN18 (Bordes et al., 2013) and FB15k (Bordes et al., 2013), respectively, where the *inverse relation test leakage* problem, i.e. the problem of inverted test triples appearing in the training set, has been corrected.

### 4.2 Settings

We used RoBERTa$_{\text{LARGE}}$ (Liu et al., 2019), a BERT$_{\text{LARGE}}$-based architecture with 24 layers, 1024 hidden size, 16 self-attention heads, and 355M parameters, for the pre-trained language model as it has been shown in a previous study to perform better than BERT (hits@1 0.243 vs. 0.222 and MR 51 vs. 99, link prediction on WN18RR) (Wang et al., 2021). For pre-training, we used learning rate = 5e-05, batch size = 32, epoch = 20 (WN18RR), 10 (FB15k-237), and 1,000 (UMLS),

Table 3: Breakdown of the original hypothesis and their results on WN18RR. For claim 1, we continued to pre-train RoBERTa$_{\text{LARGE}}$ using the knowledge graph without the ER-type embeddings. Note that we did not also use the ER-type embeddings layer in the fine-tuning stage. For claim 2, we learned the ER-type embeddings in the fine-tuning stage only without any further pre-training.

| | | ER-type embeddings | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | Continue pre-training | Pre-train | Fine-tune | Hits @1 | Hits @3 | Hits @10 | MR | MRR |
| Claim 1 | o | x | x | **0.331** | 0.529 | 0.728 | 53.5 | 0.462 |
| Claim 2 | x | - | o | 0.322 | 0.489 | 0.672 | 66.4 | 0.439 |
| KGLM$_{\text{GR}}$ | o | o | o | 0.330 | **0.538** | **0.741** | **40.18** | **0.467** |

and AdamW optimizer (Loshchilov and Hutter, 2017). For fine-tuning training data, we sampled 10 negative triples for a positive triple by corrupting both the head and tail entity 5 times each. We used the validation set to find the optimal learning rates = $\{1e-06, 5e-07\}$, batch size = $\{16, 32\}$, epochs = $\{1, 2, 3, 4, 5\}$ for WN18RR and FB15k-237 and 25, 50, 75, 100 for UMLS, and $\alpha$ from 0.0 to 1.0 with an increment of 0.1. For all experiments, we set $\alpha = 0.5$ based on the WN18RR validation set performance. Both pre-training and fine-tuning were performed on 3 × Nvidia Quadro RTX 6000 GPUs in a distributed manner using the 16-bit mixed precision and DeepSpeed (Rasley et al., 2020; Rajbhandari et al., 2020) library in the stage-2 setting. We used the Transformers library (Wolf et al., 2019).

### 4.3 Link Prediction Results

The hypothesis behind the KGLM was that learning the ER-type embedding layers in the pre-training stage using the corpus generated by the knowledge graph, followed by fine-tuning has the best performance. To test our hypothesis, we broke down the hypothesis into two separate claims. For the first claim, we only continued pre-training RoBERTa$_{\text{LARGE}}$ followed by fine-tuning without the ER-type embeddings. This test removes the contribution from the ER-type embeddings and solely tests the performance gained by further pre-training the model with the knowledge graph as input. Table 3 shows that claim 1 falls behind the KGLM$_{\text{GR}}$ in all metrics except for hits @1 (0.331 vs. 0.330, respectively). For the second claim, we did not continue pre-training and instead used the RoBERTa$_{\text{LARGE}}$ pre-trained weights as-is. We then learned the ER-type embeddings in the fine-tuning stage. This test shows if the ER-type embeddings can be learned only during the fine-tuning stage. Table 3 shows that KGLM$_{\text{GR}}$ outperforms all of the metrics obtained using the second claim. This result shows that the combination of these two claims works in a non-linear fashion to maximize performance.

The results of performing link prediction on the benchmark datasets are shown in Table 2. Compared to StAR, which had the best performance on MR and hits@10 on WN18RR, KGLM$_{\text{GR}}$ outperformed all the metrics with 21.2% improved MR (40.18 vs. 51, respectively) and 4.5% increased hits@10 (0.709 vs. 0.741, respectively). Although still inferior compared to the graph embedding approaches, KGLM$_{\text{GR}}$ has 35.8% improved hits@1 compared to the best language model-based approach StAR (0.243 vs. 0.330, respectively). Across all model types, KGLM$_{\text{GR}}$ has the best performance on all metrics for WN18RR except for hits@1. Although we did not observe any improvement compared to StAR for the FB15k-237 dataset, we had the best performance on all metrics for UMLS with 21.2% improved MR than ComplEx (1.19 vs. 1.51, respectively). KGLM$_{\text{GR}}$ outperformed KGLM$_{\text{Base}}$ in all metrics.

## 5 Conclusion

In this work, we presented KGLM, which introduces a new entity/relation (ER)-type embedding layer for learning the structure of the knowledge graph. Compared to the previous language model-based methods that only fine-tune for a given task, we found that learning the ER-type embeddings in the pre-training stage followed by fine-tuning resulted in better performance. In future work, we plan to further test the version of KGLM that takes into account entity types, KGLM$_{\text{GER}}$, on domain-specific knowledge graphs like KIDS (Youn et al., 2022) with entity types in their schema.

## Limitations

Although KGLM outperforms state-of-the-art models when the training set includes full sentences (e.g., UMLS and WN18RR), the model performed similarly to the state-of-the-art in cases where the training dataset had only ontological relationships, such as the /music/artist/origin relation present in the FB15k-237 dataset. One major limitation of the proposed method is the long training and inference time, which we plan to alleviate by adopting Siamese-style textual encoders (Wang et al., 2021; Li et al., 2022) in future work.

## Ethics Statement

The authors declare no competing interests.

## Acknowledgements

We would like to thank the members of the Tagkopoulos lab for their suggestions. This work was supported by the USDA-NIFA AI Institute for Next Generation Food Systems (AIFS), USDA-NIFA award number 2020-67021-32855 and the NIEHS grant P42ES004699 to I.T. J.Y. conceived the project and performed all experiments. Both J.Y. and I.T. wrote the manuscript. I.T. supervised all aspects of the project.

## References

Brian P Alcock, Amogelang R Raphenya, Tammy TY Lau, Kara K Tsang, Mégane Bouchard, Arman Edalatmand, William Huynh, Anna-Lisa V Nguyen, Annie A Cheng, Sihan Liu, et al. 2020. Card 2020: antibiotic resistome surveillance with the comprehensive antibiotic resistance database. *Nucleic acids research*, 48(D1):D517–D525.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

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

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

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 708–716.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-second AAAI conference on artificial intelligence*.

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

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 529–539.

Da Li, Ming Yi, and Yukai He. 2022. Lp-bert: Multi-task pre-training knowledge graph bert for link prediction. *arXiv preprint arXiv:2201.04843*.

Yixue Li and Luonan Chen. 2014. Big biological data: challenges and opportunities. *Genomics, proteomics & bioinformatics*, 12(5):187.

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

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

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Andrea Rossi, Denilson Barbosa, Donatella Firmani, Antonio Matinata, and Paolo Merialdo. 2021. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(2):1–49.

Adrian Silvescu, Doina Caragea, and Anna Atramentov. 2012. Graph databases. *Artificial Intelligence Research Laboratory Department of Computer Science, Iowa State University*.

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

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

Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2019. A capsule network-based embedding model for knowledge graph completion and search personalization. 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 2180–2189.

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

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(12):2724–2743.

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

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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

Jason Youn, Navneet Rai, and Ilias Tagkopoulos. 2022. Knowledge integration and decision support for accelerated discovery of antibiotic resistance genes. *Nature Communications*, 13(1):1–11.

Hanwen Zha, Zhiyu Chen, and Xifeng Yan. 2021. Inductive relation prediction by bert. *arXiv preprint arXiv:2103.07102*.

# A Previous Work

## A.1 KG-BERT

KG-BERT (Yao et al., 2019) is a fine-tuning method that utilizes the base version of the pre-trained language model BERT (BERT$_{\text{BASE}}$) (Devlin et al., 2018) as an encoder for entities and relations of the knowledge graph. Specifically, KG-BERT first converts a triple ($h$, $r$, $t$) to a sequence of tokens $w^{(h,r,t)} = \langle \texttt{[CLS]} w_a^h \texttt{[SEP]} w_b^r \texttt{[SEP]} w_c^t \texttt{[SEP]} : a \in \{1..|h|\} \,\&\, b \in \{1..|r|\} \,\&\, c \in \{1..|t|\} \rangle$, where $w_n$ denotes the n$^{\text{th}}$ token of either entity or relation, `[CLS]` and `[SEP]` are the special tokens, while $|h|$, $|r|$, and $|t|$ denote the number of tokens in the head entity, relation, and tail entity, respectively. This textual token sequence is then converted to a sequence of token embeddings $\mathbf{w}^{(h,r,t)} \in \mathbb{R}^{d \times (|h|+|r|+|t|+4)}$, where $d$ is the dimension of the embeddings and 4 is from the special tokens. Then the segment embeddings $\mathbf{s}^{(h,r,t)} = \langle (\mathbf{s}_e)_{\times(|h|+2)} (\mathbf{s}_r)_{\times(|r|+1)} (\mathbf{s}_e)_{\times(|t|+1)} \rangle$, where $\mathbf{s}_e$ and $\mathbf{s}_r$ are used to differentiate entities from relations, respectively, as well as the position embeddings $\mathbf{p}^{(h,r,t)} = \langle \mathbf{p}_i : i \in \{1..(|h|+|r|+|t|+4)\} \rangle$ are added to the token embeddings $\mathbf{w}^{(h,r,t)}$ to form a final input representation $\mathbf{X}^{(h,r,t)} \in \mathbb{R}^{d \times (|h|+|r|+|t|+4)}$ that is fed to BERT as input. Then, the score of how likely a given triple ($h$, $r$, $t$) is to be true is computed by

$$score_{\text{KG-BERT}}(h, r, t) = \text{SeqCls}(\mathbf{X}^{(h,r,t)}). \quad (6)$$

KG-BERT significantly improved the MR of the link prediction task compared to the previous state-of-the-art approach CapsE (Vu et al., 2019) (97 compared to 719, an 86.5% decrease), but suffered from poor hits@1 of 0.041 due to the entity ambiguation problem and lack of structural learning (Wang et al., 2021; Cucerzan, 2007).

### A.2 StAR

StAR (Wang et al., 2021) is a hybrid model that learns both the contextual and structural information of the knowledge graph by augmenting the structured knowledge in the encoder. It divides a triple into two parts, (*h*, *r*) and (*t*), and applies a Siamese-style transformer with a sequence classification head to generate $\boldsymbol{u} = \text{Pool}(\mathbf{X}^{(h,r)}) \in \mathbb{R}^{d\times(|h|+|r|+3)}$ and $\boldsymbol{v} = \text{Pool}(\mathbf{X}^{(t)}) \in \mathbb{R}^{d\times(|t|+2)}$, respectively, where Pool(·) is the output of the RoBERTa's pooling layer. The first scoring module focuses on classifying the triple by applying a

$$score^{c}_{\text{StAR}}(h, r, t) = \text{Cls}([\boldsymbol{u}; \boldsymbol{u} \times \boldsymbol{v}; \boldsymbol{u} - \boldsymbol{u}; \boldsymbol{v}]), \tag{7}$$

where Cls(·) is a neural binary classifier with a dense layer followed by a softmax activation function. The second scoring module then adopts the idea of how translation-based graph embedding methods like TransE learns the graph structure by minimizing the distance between $\boldsymbol{u}$ and $\boldsymbol{v}$ as

$$score^{d}_{\text{StAR}}(h, r, t) = -||\boldsymbol{u} - \boldsymbol{v}||, \tag{8}$$

where $||\cdot||$ is the $L2$-normalization. During the training, StAR uses a weighted average of the binary cross entropy loss computed using $score^{c}_{\text{StAR}}(h, r, t)$ and the margin-based hinge loss computed using $score^{d}_{\text{StAR}}(h, r, t)$, whereas only the $score^{c}_{\text{StAR}}(h, r, t)$ is used for inference. This approach shows a new state-of-the performance over the metrics MR (51) and hits@10 (0.709), as well as significantly improving the hits@1 compared to the KG-BERT (0.041 to 0.243, a 492.7% increase).