Robust Knowledge Graph Completion with Stacked Convolutions and a Student Re-Ranking Network

Justin Lovelace[1] **Denis Newman-Griffis**[2] **Shikhar Vashishth**[3]*
Jill Fain Lehman[4] **Carolyn Penstein Rosé**[1]
[1]Language Technologies Institute, Carnegie Mellon University, USA
[2]Department of Biomedical Informatics, University of Pittsburgh, USA
[3]Microsoft Research,
[4]Human-Computer Interaction Institute, Carnegie Mellon University, USA
{jlovelac, jfl, cpa3}@cs.cmu.edu, dnewmangriffis@pitt.edu
t-svashishth@microsoft.com

Abstract

Knowledge Graph (KG) completion research usually focuses on densely connected benchmark datasets that are not representative of real KGs. We curate two KG datasets that include biomedical and encyclopedic knowledge and use an existing commonsense KG dataset to explore KG completion in the more realistic setting where dense connectivity is not guaranteed. We develop a deep convolutional network that utilizes textual entity representations and demonstrate that our model outperforms recent KG completion methods in this challenging setting. We find that our model's performance improvements stem primarily from its robustness to sparsity. We then distill the knowledge from the convolutional network into a student network that re-ranks promising candidate entities. This re-ranking stage leads to further improvements in performance and demonstrates the effectiveness of entity re-ranking for KG completion.[1]

1 Introduction

Knowledge graphs (KGs) have been shown to be useful for a wide range of NLP tasks, such as question answering (Bordes et al., 2014a,b), dialog systems (Ma et al., 2015), relation extraction (Mintz et al., 2009; Vashishth et al., 2018), and recommender systems (Zhang et al., 2016). However, because scaling the collection of facts to provide coverage for all the true relations that hold between entities is difficult, most existing KGs are incomplete (Dong et al., 2014), limiting their utility for downstream applications. Because of this problem, KG completion (KGC) has come to be a widely studied task (Yang et al., 2015; Trouillon et al., 2016; Shang et al., 2018; Dettmers et al., 2018; Sun et al., 2019; Balazevic et al., 2019; Malaviya et al., 2020; Vashishth et al., 2020a).

The increased interest in KGC has led to the curation of a number of benchmark datasets such as FB15K (Bordes et al., 2013), WN18 (Bordes et al., 2013), FB15k-237 (Toutanova and Chen, 2015), and YAGO3-10 (Rebele et al., 2016) that have been the focus of most of the work in this area. However, these benchmark datasets are often curated in such a way as to produce densely connected networks that simplify the task and are not representative of real KGs. For instance, FB15K includes only entities with at least 100 links in Freebase, while YAGO3-10 is limited to only include entities in YAGO3 (Rebele et al., 2016) that have at least 10 relations.

Real KGs are not as uniformly dense as these benchmark datasets and have many sparsely connected entities (Pujara et al., 2017). This can pose a challenge to typical KGC methods that learn entity representations solely from the knowledge that already exists in the graph.

Textual entity identifiers can be used to develop entity embeddings that are more robust to sparsity (Malaviya et al., 2020). It has also been shown that textual triplet representations can be used with BERT for triplet classification (Yao et al., 2019). Such an approach can be extended to the more common ranking paradigm through the exhaustive evaluation of candidate triples, but that does not scale to large KG datasets.

In our work, we found that existing neural KGC models lack the complexity to effectively fit the training data when used with the pre-trained textual embeddings that are necessary for representing sparsely connected entities. We develop an expressive deep convolutional model that utilizes textual entity representations more effectively and improves sparse KGC. We also develop a student re-ranking model that is trained using knowledge dis-

* Work performed while at Carnegie Mellon University.
[1]https://github.com/justinlovelace/robust-kg-completion

tilled from our original ranking model and demonstrate that the re-ranking procedure is particularly effective for sparsely connected entities. Through these innovations, we develop a KGC pipeline that is more robust to the realities of real KGs. Our contributions can be summarized as follows.

- We develop a deep convolutional architecture that utilizes textual embeddings more effectively than existing neural KGC models and significantly improves performance for sparse KGC.
- We develop a re-ranking procedure that distills knowledge from our ranking model into a student network that re-ranks promising candidate entities.
- We curate two sparse KG datasets containing biomedical and encyclopedic knowledge to study KGC in the setting where dense connectivity is not guaranteed. We release the encyclopedic dataset and the code to derive the biomedical dataset to encourage future work.

2 Related Work

Knowledge Graph Completion: KGC models typically learn entity and relation embeddings based on known facts (Nickel et al., 2011; Bordes et al., 2013; Yang et al., 2015) and use the learned embeddings to score potential candidate triples. Recent work includes both non-neural (Nickel et al., 2016; Trouillon et al., 2016; Liu et al., 2017; Sun et al., 2019) and neural (Socher et al., 2013; Dong et al., 2014; Dettmers et al., 2018; Vashishth et al., 2020b) approaches for embedding KGs. However, most of them only demonstrate their efficacy on artificially dense benchmark datasets. Pujara et al. (2017) show that the performance of such methods varies drastically with sparse, unreliable data. We compare our proposed method against the existing approaches in a realistic setting where the KG is not uniformly dense.

Prior work has effectively utilized entity names or descriptions to aid KGC (Socher et al., 2013; Xie et al., 2016; Xiao et al., 2016). In more recent work, Malaviya et al. (2020) explore the problem of KGC using commonsense KGs, which are much sparser than standard benchmark datasets. They adapt an existing KGC model to utilize BERT (Devlin et al., 2019) embeddings. In this paper, we develop a deep convoluational architecture that is more effective than adapting existing shallow models which we find to be underpowerered for large KG datasets.

Yao et al. (2019) developed a triplet classification model by directly fine-tuning BERT with textual entity representations and reported strong classification results. They also adapted their triplet classification model to the ranking paradigm by exhaustively evaluating all possible triples for a given query, $(e_1, r, ?)$. However, the ranking performance was not competitive[2], and such an approach is not scalable to large KG datasets like those explored in this work. Exhaustively applying BERT to compute all rankings for the test set for our largest dataset would take over two months. In our re-ranking setting, we reduce the number of triples that need to be evaluated by over $7700\times$, reducing the evaluation time to less than 15 minutes.

BERT as a Knowledge Base: Recent work (Petroni et al., 2019; Jiang et al., 2020; Rogers et al., 2020) has utilized the masked-language-modeling (MLM) objective to probe the knowledge contained within pre-trained models using fill-in-the-blank prompts (e.g. "Dante was born in `[MASK]`"). This body of work has found that pre-trained language models such as BERT capture some of the relational knowledge contained within their pre-training corpora. This motivates us to utilize these models to develop entity representations that are well-suited for KGC.

Re-Ranking: Wang et al. (2011) introduced cascade re-ranking for document retrieval. This approach applies inexpensive models to develop an initial ranking and utilizes expensive models to improve the ranking of the top-k candidates. Re-ranking has since been successfully applied across many retrieval tasks (Matsubara et al., 2020; Pei et al., 2019; Nogueira and Cho, 2019). Despite re-ranking's widespread success, recent KGC work utilizes a single ranking model. We develop an entity re-ranking procedure and demonstrate the effectiveness of the re-ranking paradigm for KGC.

Knowledge Distillation: Knowledge distillation is a popular technique that is often used for model compression where a large, high-capacity teacher is used to train a simpler student network (Hinton et al., 2015). However, knowledge distillation has since been shown to be useful for improving model performance beyond the original setting of model compression. Li et al. (2017) demonstrated that knowledge distillation improved image classification performance in a setting with noisy labels. The incompleteness of KGs leads to noisy

[2]Their reported Hits@10 for FB15K-237 was .420 which is lower than all of the models evaluated in this work.

Dataset	# Nodes	# Rels	# Train	# Valid	# Test
FB15k-237	14,451	237	272,115	17,535	20,466
SNOMED CT Core	77,316	140	502,224	71,778	143,486
CN-100K	78,088	34	100,000	1,200	1,200
FB15k-237-Sparse	14,451	237	18,506	17,535	20,466

Table 1: Dataset statistics

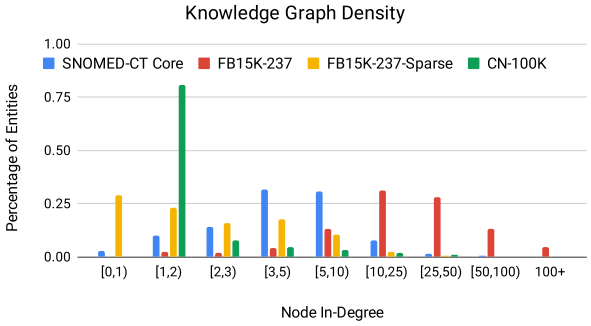

Figure 1: In-degrees of entities in the training KGs (including inverse relations)

training labels which motivates us to use knowledge distillation to train a student re-ranking model that is more robust to the label noise.

3 Datasets

We examine KGC in the realistic setting where KGs have many sparsely connected entities. We utilize a commonsense KG dataset that has been used in past work and curate two additional sparse KG datasets containing biomedical and encyclopedic knowledge. We release the encyclopedic dataset and the code to derive the biomedical dataset to encourage future work in this challenging setting. The summary statistics for all datasets are presented in Table 1 and we visualize the connectivity of the datasets in Figure 1.

3.1 SNOMED CT Core

For constructing SNOMED CT Core, we use the knowledge graph defined by SNOMED CT (Donnelly, 2006), which is contained within the Unified Medical Language System (UMLS) (Bodenreider, 2004). SNOMED CT is well-maintained and is one of the most comprehensive knowledge bases contained within the UMLS (Jiménez-Ruiz et al., 2011; Jiang and Chute, 2009). We first extract the UMLS[3] concepts found in the CORE Problem List Subset of the SNOMED CT knowledge base. This subset is intended to contain the concepts most useful for documenting clinical information. We then expand the graph to include all concepts that are directly linked to those in the CORE Problem List Subset according to the relations defined by the SNOMED CT KG. Our final KG consists of this set of concepts and the SNOMED CT relations connecting them. Importantly, we do not filter out rare entities from the KG, as is commonly done during the curation of benchmark datasets.

To avoid leaking data from inverse, or otherwise informative, relations, we divide the facts into training, validation, and testing sets based on unordered tuples of entities $\{e_1, e_2\}$ so that all relations between any two entities are confined to a single split. Unlike some other KG datasets that filter out inverse relations, we divide our dataset in such a way that this is not necessary; our dataset already includes inverse relations, and they do not need to be manually added for training and evaluation as is standard practice (Dettmers et al., 2018; Malaviya et al., 2020).

Because we represent entities using textual descriptions in this work, we also mine the entities' preferred concept names (e.g. *"Traumatic hematoma of left kidney"*) from the UMLS.

3.2 FB15k-237-Sparse

The FB15k-237 (Toutanova and Chen, 2015) dataset contains encyclopedic knowledge about the world, e.g. *(Barack Obama, placeOfBirth, Honolulu)*. Although the dataset is very densely connected, that density is artificial. FB15K (Bordes et al., 2013), the precursor to FB15k-237, was curated to only include entities with at least 100 links in Freebase (Bollacker et al., 2008).

The dense connectivity of FB15k-237 does allow us to to ablate the effect of this density. We utilize the FB15k-237 dataset and also develop a new dataset, denoted FB15k-237-Sparse, by randomly downsampling the facts in the training set of FB15k-237 to match the average in-degree of the ConceptNet-100K dataset. We use this to directly evaluate the effect of increased sparsity.

For the FB15k-237 dataset, we use the textual identifiers released by Xie et al. (2016). They released both entity names (e.g. "Jason Frederick Kidd") as well as brief textual descriptions (e.g. "Jason Frederick Kidd is a retired American professional basketball player…") for most entities. We utilize the textual descriptions when available.

3.3 ConceptNet-100K

ConceptNet (Speer and Havasi, 2013) is a KG that contains commonsense knowledge about the world

[3]We work with the 2020AA release of the UMLS.

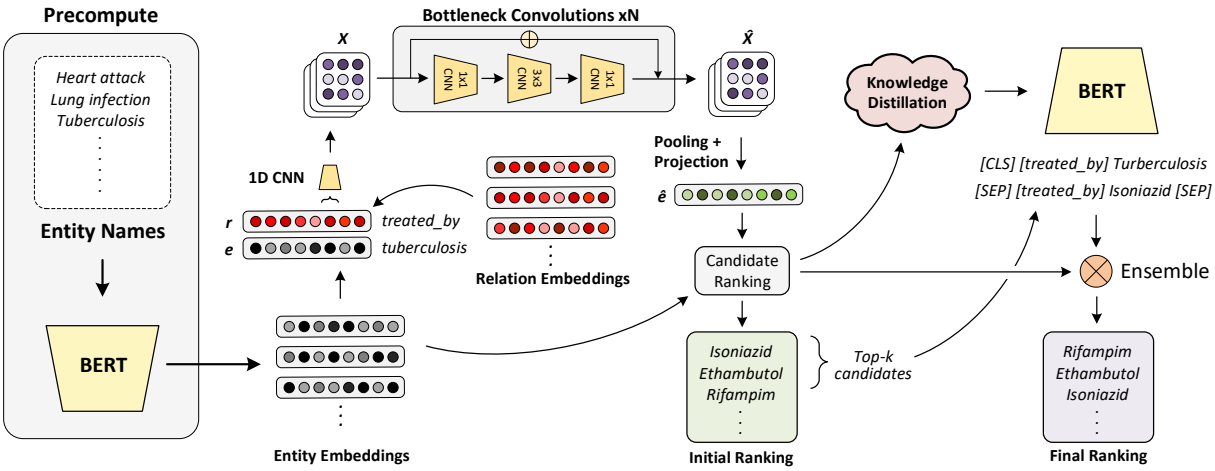

Figure 2: We utilize BERT to precompute entity embeddings. We then stack the precomputed entity embedding with a learned relation embedding and project them to a two-dimensional spatial feature map, upon which we apply a sequence of two-dimensional convolutions. The final feature map is then average pooled and projected to a query vector, which is used to rank candidate entities. We extract promising candidates and train a re-ranking model utilizing knowledge distilled from the original ranking model. The final candidate ranking is generated by ensembling the ranking and re-ranking models.

such as the fact *(go to dentist, motivatedBy, prevent tooth decay)*. We utilize ConceptNet-100k (CN-100K) (Li et al., 2016) which consists of the Open Mind Common Sense entries in the ConceptNet dataset. This KG is much sparser than benchmark datasets like FB15k-237, which makes it well-suited for our purpose. We use the training, validation, and testing splits of Malaviya et al. (2020) to allow for direct comparison. We also use the textual descriptions released by Malaviya et al. (2020) to represent the KG entities.

4 Methods

We provide an overview of our model architecture in Figure 2. We first extract feature representations from BERT (Devlin et al., 2019) to develop textual entity embeddings. Motivated by our observation that existing neural KG architectures are underpowered in our setting, we develop a deep convolutional network utilizing architectural innovations from deep convolutional vision models. Our model's design improves its ability to fit complex relationships in the training data which leads to downstream performance improvements.

Finally, we distill our ranking model's knowledge into a student re-ranking network that adjusts the rankings of promising candidates. In doing so, we demonstrate the effectiveness of the re-ranking paradigm for KGC and develop a KGC pipeline with greater robustness to the sparsity of real KGs.

4.1 Entity Ranking

We follow the standard formulation for KGC. We represent a KG as a set of entity-relation-entity facts (e_1, r, e_2). Given an incomplete fact, $(e_1, r, ?)$, our model computes a score for all candidate entities e_i that exist in the graph. An effective KGC model should assign greater scores to correct entities than incorrect ones. We follow recent work (Dettmers et al., 2018; Malaviya et al., 2020) and consider both forward and inverse relations (e.g. treats and treated_by) in this work. For the datasets that do not already include inverse relations, we introduce an inverse fact, (e_2, r^{-1}, e_1), for every fact, (e_1, r, e_2), in the dataset.

4.1.1 Textual Entity Representations

We utilize BERT (Devlin et al., 2019) to develop entity embeddings that are invariant to the connectivity of the KG. We follow the work of Malaviya et al. (2020) and adapt BERT to each KG's naming style by fine-tuning BERT using the MLM objective with the set of entity identifiers in the KG.

For CN-100K and FB15k-237, we utilize the BERT-base uncased model. For SNOMED CT Core KG, we utilize PubMedBERT (Gu et al., 2020) which is better suited for the biomedical terminology in the UMLS.

We apply BERT to the textual entity identifiers and mean-pool across the token representations from all BERT layers to obtain a summary feature vector for the concept name. We fix these embed-

dings during training because we must compute scores for a large number of potential candidate entities for each training example. This makes fine-tuning BERT prohibitively expensive.

4.1.2 Deep Convolutional Architecture

Inspired by the success of deep convolutional models in computer vision (Krizhevsky et al., 2012; Simonyan and Zisserman, 2015; He et al., 2016; Huang et al., 2019, 2017), we develop a knowledge base completion model based on the seminal ResNet architecture (He et al., 2016) that is sufficiently expressive to model complex interactions between the BERT feature space and the relation embeddings.

Given an incomplete triple $(e_i, r_j, ?)$, we begin by stacking the precomputed entity embedding $\mathbf{e} \in \mathbb{R}^{1\times d}$ with the learned relation embedding of the same dimension $\mathbf{r} \in \mathbb{R}^{1\times d}$ to produce a feature vector of length d with two channels $\mathbf{q} \in \mathbb{R}^{2\times d}$. We then apply a one-dimensional convolution with a kernel of width 1 along the length of the feature vector to project each position i to a two-dimensional spatial feature map $\mathbf{x_i} \in \mathbb{R}^{f\times f}$ where the convolution has $f \times f$ filters. Thus the convolution produces a two-dimensional spatial feature map $\mathbf{X} \in \mathbb{R}^{f\times f\times d}$ with d channels, representing the incomplete query triple $(e_i, r_j, ?)$.

The spatial feature map, $\mathbf{X} \in \mathbb{R}^{f\times f\times d}$, is analogous to a square image with a side length of f and d channels, allowing for the straightforward application of deep convolutional models such as ResNet. We apply a sequence of $3N$ bottleneck blocks to the spatial feature map where N is a hyperparameter that controls the depth of the network. A bottleneck block consists of three consecutive convolutions: a 1×1 convolution, a 3×3 convolution, and then another 1×1 convolution. The first 1×1 convolution reduces the feature map dimensionality by a factor of 4 and then the second 1×1 convolution restores the feature map dimensionality. This design reduces the dimensionality of the expensive 3×3 convolutions and allows us to increase the depth of our model without dramatically increasing its parameterization. We double the feature dimensionality of the bottleneck blocks after N and $2N$ blocks so the dimensionality of the final feature map produced by the sequence of convolutions is $4d$.

We add residual connections to each bottleneck block which improves training for deep networks (He et al., 2016). If we let $\mathcal{F}(X)$ represent the application of the bottleneck convolutions, then the output of the bottleneck block is $Y = \mathcal{F}(X) + X$. We apply batch normalization followed by a ReLU nonlinearity (Nair and Hinton, 2010) before each convolutional layer (He et al., 2016) .

We utilize circular padding (Wang et al., 2018; Vashishth et al., 2020a) with the 3×3 convolutions to maintain the spatial size of the feature map and use a stride of 1 for all convolutions. For the bottleneck blocks that double the dimensionality of the feature map, we utilize a projection shortcut for the residual connection (He et al., 2016).

4.1.3 Entity Scoring

Given an incomplete fact $(e_i, r_j, ?)$, our convolutional architecture produces a feature map $\hat{\mathbf{X}} \in \mathbb{R}^{f\times f\times 4d}$. We average pool this feature representation over the spatial dimension which produces a summary feature vector $\hat{\mathbf{x}} \in \mathbb{R}^{4d}$. We then apply a fully connected layer followed by a PReLU nonlinearity (He et al., 2015) to project the feature vector back to the original embedding dimensionality d. We denote this final vector $\hat{\mathbf{e}}$ and compute scores for candidate entities using the dot product with candidate entity embeddings. The scores can be efficiently computed for all entities simultaneously using a matrix-vector product with the embedding matrix $\mathbf{y} = \hat{\mathbf{e}}\mathbf{E}^T$ where $\mathbf{E} \in \mathbb{R}^{m\times d}$ stores the embeddings for all m entities in the KG.

4.1.4 Training

Adopting the terminology used by Ruffinelli et al. (2020), we utilize a 1vsAll training strategy with the binary cross-entropy loss function. We treat every fact in our dataset, (e_i, r_j, e_k), as a training sample where $(e_i, r_j, ?)$ is the input to the model. We compute scores for all entities as described previously and apply a sigmoid operator to induce a probability for each entity. We treat all entities other than e_k as negative candidates and then compute the binary cross-entropy loss.

We train our model using the Adam optimizer (Kingma and Ba, 2015) with decoupled weight decay regularization (Loshchilov and Hutter, 2019) and label smoothing. We train our models for a maximum of 200 epochs and terminate training early if the validation Mean Reciprocal Rank (MRR) has not improved for 20 epochs. We trained all of the models used in this work using a single NVIDIA GeForce GTX 1080 Ti.

4.2 Entity Re-Ranking

4.2.1 Re-Ranking Network

We use our convolutional network to extract the top-k entities for every unique training query and then train a re-ranking network to rank these entities. We design our student re-ranking network as a triplet classification model that utilizes the full candidate fact, (e_i, r_j, e_k), instead of an incomplete fact, $(e_i, r_j, ?)$. This allows the network to model interactions between all elements of the triple. The re-ranking setting also enables us to directly fine-tune BERT which often improves performance (Peters et al., 2019).

We introduce relation tokens[4] for each relation in the knowledge graph and construct the textual input by prepending the head and tail entities with the relation token and then concatenating the two sequences. Thus the triple ("head name", r_i, "tail name") would be represented as "`[CLS] [REL_i] head name [SEP] [REL_i] tail name [SEP]`". We use a learned linear combination of the `[CLS]` embedding from each layer as the final feature representation for the prediction.

4.2.2 Knowledge Distillation

A sufficiently performant ranking model can provide an informative prior that can be used to smooth the noisy training labels and improve our re-ranking model. For each training query i, we normalize the logits produced by our teacher ranking model, $f_T(\mathbf{x}_i)$, for the k candidate triples, $f_T(\mathbf{x}_i)_{0:k}$, as

$$\mathbf{s}_{ik:(i+1)k} = \text{softmax}(f_T(\mathbf{x}_i)_{0:k}/T)$$

where T is the temperature (Hinton et al., 2015).

Our training objective for our student model, $f_S(x_i)$, is a weighted average of the binary cross entropy loss, $\mathcal{L}_{bce}$, using the teacher's normalized logits, $\mathbf{s}$, and the noisy training labels, $\mathbf{y}$.

$$\begin{aligned}\mathcal{L}_{KD}(y_i, x_i) &= \lambda \mathcal{L}_{bce}(s_i, f_S(x_i)) \\ &+ (\lambda - 1)\mathcal{L}_{bce}(y_i, f_S(x_i)) \\ &= \mathcal{L}_{bce}((\lambda - 1)y_i + \lambda s_i, f_S(x_i))\end{aligned}$$

We select $\lambda \in \{.25, .5, .75, 1\}$ to optimize the balance between the two objectives using validation performance.

4.2.3 Training

For our experiments, we extract the top $k = 10$ candidates produced by our ranking model for every query in the training set. We train our student network using the Adam optimizer (Kingma and Ba, 2015) with decoupled weight decay regularization (Loshchilov and Hutter, 2019). We fine-tune BERT for a maximum of 10 epochs and terminate training early if the Mean Reciprocal Rank (MRR) on validation data has not improved for 3 epochs.

4.2.4 Student-Teacher Ensemble

For every query, we apply our re-ranking network to the top $k = 10$ triples and compute the final ranking using an ensemble of the teacher and student networks. The final ranking are computed with

$$\begin{aligned}\hat{\mathbf{s}}_{ik:(i+1)k} &= \alpha(\text{softmax}(f_S(\mathbf{x}_{ik:(i+1)k}))) \\ &+ (1 - \alpha)(\text{softmax}(f_T(\mathbf{x}_i)_{0:k})))\end{aligned}$$

where $0 \leq \alpha \leq 1$ controls the impact of the student re-ranker. The cost of computing $\hat{\mathbf{s}}_{ik:(i+1)k}$ is negligible, so we sweep over $[0, 1]$ in increments of .01 and select the α that achieves the best validation MRR.

5 Experiments

5.1 Baselines

We utilize the same representative selection of KG models from Malaviya et al. (2020) as baselines: DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016) ConvE (Dettmers et al., 2018), and ConvTransE (Shang et al., 2018). This is not an exhaustive selection of all recent KG methods, but a recent replication study by Ruffinelli et al. (2020) found that the baselines that we use are competitive with the state-of-the-art and often outperform more recent models when trained appropriately.

We develop additional baselines by adapting the shallow convolutional KGC models to use BERT embeddings to evaluate the benefits of utilizing our proposed convolutional architecture instead of simply repurposing existing KGC models. We refer to these models as BERT-ConvE and BERT-ConvTransE. Malaviya et al. (2020) used BERT embeddings in conjunction with ConvTransE for commonsense KGC, but their model was prohibitively large to reproduce. We refer to

[4] We use relation tokens instead of free-text relation representations because the relation identifiers for our datasets are not all well-formed using natural language, and the different styles would introduce a confounding factor that would complicate our evaluation. Utilizing appropriate free-text relation identifiers may improve performance, but we leave that to future work.

	SNOMED CT Core					CN-100K				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [♣]	5146	.293	.226	.318	.426	–	.090	.045	.098	.174
ComplEx [♣]	3903	.302	.224	.332	.456	–	.114	.074	.125	.190
ConvE [♣]	3739	.271	.191	.303	.429	–	.209	.140	.229	.340
ConvTransE [♣]	3585	.290	.213	.321	.442	–	.187	.079	.239	.390
BERT-ConvE	414	.383	.277	.430	.591	260	.453	.332	.521	.691
BERT-ConvTransE	514	.373	.273	.417	.568	276	.458	.340	.520	.675
BERT-Large-ConvTransE [♣]	–	–	–	–	–	–	.523	.410	.585	.735
BERT-DeepConv	265	.479	.374	.532	.685	**161**	.540	.418	.610	**.772**
BERT-ResNet	265	.492*	.389	.544	**.691**	169	.550*	.426	.628	.769
+ Re-ranking	265	.562†	.482	.608	**.691**	170	.377	.216	.437	.769
+ Knowledge Distillation (KD)	265	.566†	.487	.614	**.691**	169	.528	.402	.603	.769
+ Ranking Ensemble (RE)	**264**	.576†	**.503**	.619	**.691**	169	.555	.438	.623	.769
+ KD and RE	**264**	**.577**†	.501	**.623**	**.691**	169	**.569**†	**.452**	**.647**	.769

	FB15k-237					FB15k-237-Sparse				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
DistMult [♠]	–	.343	–	–	.531	3061	.136	.092	.146	.223
ComplEx [♠]	–	.348	–	–	**.536**	3333	.132	.091	.143	.216
ConvE [♠]	–	.339	–	–	.521	2263	.156	.106	.165	.258
ConvTransE [♢]	–	.33	.24	.37	.51	2285	.153	.103	.161	.255
BERT-ConvE	193	.305	.224	.330	.465	408	.190	.128	.200	.315
BERT-ConvTransE	211	.296	.218	.321	.449	**390**	.188	.127	.199	.310
BERT-DeepConv	190	.327	.246	.354	.488	422	.188	.127	.197	.314
BERT-ResNet	186	.346*	.262	.379	.514	413	.191*	.128	.201	**.317**
+ Re-ranking	187	.304	.212	.329	.514	413	.190	.128	.200	**.317**
+ Knowledge Distillation (KD)	187	.310	.220	.334	.514	413	.197†	.135	.209	**.317**
+ Ranking Ensemble (RE)	**186**	**.354**†	**.270**	**.387**	.514	413	**.199**†	**.137**	.210	**.317**
+ KD and RE	**186**	.353†	.269	.386	.514	413	.198†	.136	**.211**	**.317**

Table 2: Comparison of KGC results across all datasets. We indicate statistical significance for: (1) Improvements of deep convolutional BERT models over both shallow convolutional BERT models with an underline ($p < 0.005$); (2) Improvements of BERT-ResNet over BERT-DeepConv with a $*$ ($p < 0.05$); (3) Improvements of the re-ranking configurations over the original rankings with a † ($p < 0.005$). [♣] indicates that CN-100K results are from Malaviya et al. (2020). [♠] indicates that FB15k-237 results are from Ruffinelli et al. (2020). [♢] indicates that FB15k-237 results are from Shang et al. (2018). Dashes indicate that the metric was not reported by the prior work.

their model as BERT-Large-ConvTransE and compare directly against their reported results.

We also develop a deep convolutional baseline, termed BERT-DeepConv, to evaluate the effect of the architectural innovations used in our model. BERT-DeepConv transforms the input embeddings to a spatial feature map like our proposed model, but it then applies a stack of 3×3 convolutions instead of a sequence of bottleneck blocks with residual connections. We select hyperparameters (detailed in the Appendix) for all of our BERT baselines so that they have a comparable number of trainable parameters to our proposed model. We discuss the size of these models in detail in in Section 6.4.

To evaluate the impact of our re-ranking stage, we ablate the use of knowledge distillation and ensembling. Thus we conduct experiments where our re-ranker uses only knowledge distillation, uses only ensembling, and uses neither. This means that in the most naive setting, we train the re-ranker using the hard training labels and re-rank the candidates using only the re-ranker.

5.2 Evaluation

We report standard ranking metrics: Mean Rank (MR), Mean Reciprocal Rank (MRR), Hits at 1 (H@1), Hits at 3 (H@3), and Hits at 10 (H@10). We follow past work and use the filtered setting (Bordes et al., 2013), removing all positive entities other than the target entity before calculating the target entity's rank.

We utilize paired bootstrap significance testing (Berg-Kirkpatrick et al., 2012) with the MRR to validate the statistical significance of improvements. To account for the large number of comparisons

being performed, we apply the Holm–Bonferroni method (Holm, 1979) to correct for multiple hypothesis testing. We define families for the three primary hypotheses that we tested with our experiments. They are as follows: (1) The deep convolutional BERT models outperform the shallow convolutional BERT models. (2) BERT-ResNet improves upon our BERT-DeepConv baseline. (3) The re-ranking procedure improves the original rankings.

This selection has the benefit of allowing for a more granular analysis of each conclusion while significantly reducing the number of hypotheses. The first family includes all pairwise comparisons between the two deep convolutional models and the two shallow convolutional models. The second family involves all comparisons between BERT-ResNet and BERT-DeepConv. The third family includes comparisons between all re-ranking configurations and the original rankings. We note that the p-value for each family bounds the strict condition that we report any spurious finding within the family.

6 Results and Discussion

6.1 Ranking Performance

We report results across all of our datasets in Table 2. Our ranking model, BERT-ResNet, outperforms the previously published models and our baselines across all of the sparse datasets. We find that for all sparse datasets, the models that use free text entity representations outperform the models that learn the entity embeddings during training. Among the models utilizing textual information, the deep convolutional methods generally outperform the adaptations of existing neural KG models. BERT-ResNet outperforms BERT-DeepConv across all datasets, demonstrating that the architectural innovations do improve downstream performance.

On the full FB15k-237 dataset, our proposed model is able to achieve competitive results compared to strong baselines. However, the focus of this work is not to achieve state-of-the-art performance on densely connected benchmark datasets such as FB15k-237. These results do, however, allow us to observe the outsized impact of sparsity on models that do not utilize textual information.

6.2 Re-Ranking Performance

Re-ranking entities without knowledge distillation or ensembling leads to poor results, degrading the MRR across most datasets. We note that the performance of our re-ranking model could be limited by our use of a pointwise loss function. Further exploration of pairwise or listwise learning learning-to-rank methods is a promising direction for future exploration that could lead to further improvements Guo et al. (2020).

The inclusion of either knowledge distillation or ensembling improves performance. Ensembling is particularly important, achieving a statistically significant improvement over the initial rankings across most datasets. Our final setting using both knowledge distillation and ensembling is the only setting to achieve a statistically significant improvement across all four datasets, although using both does not consistently improve performance over ensembling alone.

A plausible explanation for this is that knowledge distillation improves performance by reducing the divergence between the re-ranker and the teacher, but ensembling can already achieve a similar effect by simply increasing the weight of the teacher in the final prediction. We observe that the weight of the teacher is reduced across all four datasets when knowledge distillation is used which would be consistent with this explanation. Knowledge distillation has also been shown to be useful in situations with noisy labels (Li et al., 2017) which may explain why it was particularly effective for our sparsest dataset, CN-100K, where training with the hard labels led to particularly poor performance.

6.3 Effect of Re-Ranking

We bin test examples by the in-degree of the tail nodes and compute the MRR within these bins for our model before and after re-ranking. We report this breakdown for the SNOMED CT Core dataset in Figure 3. Our re-ranking stage improves performance uniformly across all levels of sparsity, but it is particularly useful for entities that are rarely seen during training. This is also consistent with the comparatively smaller topline improvement for the densely connected FB15k-237 dataset.

6.4 Model Capacity

We report the number of trainable parameters for the models that use textual representations along with the train and test set MRR for SNOMED CT Core in Table 3. We observe a monotonic relationship between training and testing performance and note that the shallow models fail to achieve

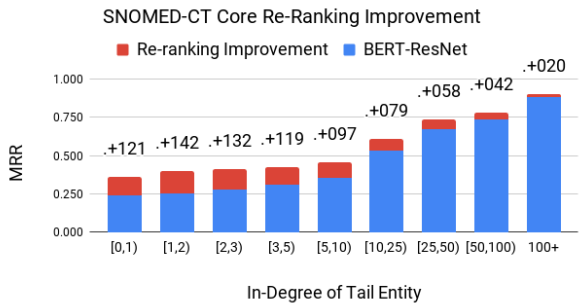

Figure 3: Effect of re-ranking on performance for SNOMED CT Core across varying levels of sparsity.

Model	Trainable Params	SNOMED CT Core Train/Test MRR
BERT-ConvE	34M	.460 / .383
BERT-ConvTransE	37M	.449 /.373
BERT-DeepConv	38M	.696 /.479
BERT-ResNet	33M	.715 / .492

Table 3: Comparison of trainable parameters for KGC models that utilize textual entity representations.

our model's test performance on the training set. This demonstrates that the shallow models lack the complexity to adequately fit the training data. A similar trend held for all datasets except for FB15k-237-Sparse whose smaller size reduces the risk of underfitting. This explains the smaller performance improvement for that dataset.

Malaviya et al. (2020) scaled up BERT-Large-ConvTransE to use over 524M trainable parameters, and their model did outperform our smaller BERT-ConvTransE baseline. However, their model still fails to match the performance of either of our deep convolutional models despite using over $15\times$ the number of trainable parameters.

7 Conclusion

KGs often include many sparsely connected entities where the use of textual entity embeddings is necessary for strong performance. We develop a deep convolutional network that is better-suited for this setting than existing neural models developed on artificially dense benchmark KGs. We also introduce a re-ranking procedure to distill the knowledge from our convolutional model into a student re-ranking network and demonstrate that our procedure is particularly effective at improving the ranking of sparse candidates. We utilize these innovations to develop a KGC pipeline with greater robustness to the realities of KGs and demonstrate the generalizability of our improvements across biomedical, commonsense, and encyclopedic KGs.

Acknowledgments

This work was supported by the National Science Foundation grant IIS 1917955 and the National Library Medicine of the National Institutes of Health under award number T15 LM007059.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. 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 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

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

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*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*, pages 165–180, Berlin, Heidelberg. Springer Berlin Heidelberg.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d

knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

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.

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*, KDD '14, pages 601–610, New York, NY, USA. ACM.

Kevin Donnelly. 2006. SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in health technology and informatics*, 121:279.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *ArXiv*, abs/2007.15779.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6):102067.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. *2016 European Conference on Computer Vision (ECCV)*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. 2019. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Guoqian Jiang and Christopher G. Chute. 2009. Auditing the Semantic Completeness of SNOMED CT Using Formal Concept Analysis. *Journal of the American Medical Informatics Association*, 16(1):89–102.

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.

Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Ian Horrocks, and Rafael Berlanga. 2011. Logic-based assessment of the compatibility of UMLS ontology sources. *Journal of Biomedical Semantics*, 2(1):S2.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, and L. Li. 2017. Learning from noisy labels with distillation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1928–1936.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2168–2178, International Convention Centre, Sydney, Australia. PMLR.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Y. Ma, P. A. Crook, R. Sarikaya, and E. Fosler-Lussier. 2015. Knowledge graph inference for spoken dialog systems. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5346–5350.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Yoshitomo Matsubara, Thuy Vu, and Alessandro Moschitti. 2020. *Reranking for Efficient Transformer-Based Answer Selection*, page 1577–1580. Association for Computing Machinery, New York, NY, USA.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961. AAAI Press.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA. Omnipress.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*.

Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 3–11, New York, NY, USA. Association for Computing Machinery.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. 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.

Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *International semantic web conference*, pages 177–185. Springer.

Anna Rogers, O. Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *ArXiv*, abs/2002.12327.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2018. End-to-end structure-aware convolutional networks for knowledge base completion. *CoRR*, abs/1811.04441.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, pages 926–934, USA. Curran Associates Inc.

Robyn Speer and Catherine Havasi. 2013. *ConceptNet 5: A Large Semantic Network for Relational Knowledge*, pages 161–176. Springer Berlin Heidelberg, Berlin, Heidelberg.

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*.

J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. 2015. Efficient object localization using convolutional networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656.

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, Beijing, China. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080. JMLR.org.

Shikhar Vashishth, Rishabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1257–1266. Association for Computational Linguistics.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020a. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 3009–3016. AAAI Press.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020b. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 105–114, New York, NY, USA. Association for Computing Machinery.

Tsun-Hsuan Wang, Hung-Jui Huang, Juan-Ting Lin, Chan-Wei Hu, Kuo-Hao Zeng, and Min Sun. 2018. Omnidirectional CNN for visual place recognition and navigation. *arXiv preprint arXiv:1803.04228*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. Ssp: Semantic space projection for knowledge graph embedding with text descriptions.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *The 30th AAAI Conference on Artificial Intelligence*.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 353–362, New York, NY, USA. ACM.

A Implementation Details

A.1 BERT MLM Pre-training

We utilize the HuggingFace Transformers library (Wolf et al., 2020) to work with pre-trained language models. We fine-tune the pre-trained language model with the masked-language-modeling objective upon the set of textual entity identifiers for the knowledge graph. We train the model for 3 epochs with a batch size of 32 using a learning rate of 3e-5. We use a warmup proportion of 0.1 of the total training steps for each dataset. We use a max sequence length of 64 during this pre-training except when using the textual descriptions associated with FB15k-237 where we use a max sequence length of 256. We utilize these dataset-specific language models for both generating the entity embeddings and for initializing the re-ranking model.

A.2 Ranking

A.2.1 Training Procedure

We train all of the ranking models implemented in this work for a maximum of 200 epochs and terminate training early if the validation MRR has not improved for 20 epochs. For evaluation, we reload the model weights from the epoch that achieved the best validation MRR and evaluate it on the test set.

A.2.2 BERT-ResNet Implementations

For our BERT-ResNet model, we set $f = 5$ where f is the hyperparameter that controls the size of

the spatial feature map produced by the initial 1D convolution. Thus our initial 1D convolution has $f \times f = 25$ filters. We set $N = 2$ where N is the hyperparameter that controls the depth of the convolutional network. This means that our BERT-ResNet model consists of $3N = 6$ sequential bottleneck blocks.

We trained the models using a batch size of 64 with a 1vsAll strategy (Ruffinelli et al., 2020) with the binary cross entropy loss function. We use the Adam optimizer (Kingma and Ba, 2015) with decoupled weight decay regularization (Loshchilov and Hutter, 2019) and train the model with a learning rate of 1e-3. We use label smoothing with a value of 0.1, clip gradients to a max value of 1, and regularize the model using weight decay with a weight of 1e-4. We apply dropout with drop probability 0.2 after the embedding layer and apply 2D dropout (Tompson et al., 2015) with the same drop probability before the 2D convolutions. We apply dropout with probability 0.3 after the pooling and fully connected layer. We manually tuned the hyperparameters for this model based on validation performance.

A.2.3 Baseline Implementations

For our baseline implementations of DistMult, ComplEx, ConvE, and ConvTransE, we adapt the implementations released by Dettmers et al. (2018) and Malaviya et al. (2020). We utilize the hyperparameters reported in the original papers and conduct a grid search to tune the embedding dimension from [100, 200, 300] and the initial learning rate from [5e-3, 1e-3, 5e-4, 1e-4] for each dataset. We train the models with a batch size of 128 using the 1vsAll strategy with the cross entropy loss function because the replication study by Ruffinelli et al. (2020) found that this training strategy generally led to better performance than other training strategies. For the grid search, we train each model for a maximum of 50 epochs and then select the hyperparameters with the best validation performance and retrain the model with our aforementioned training procedure.

For our implementation of BERT-ConvE and BERT-ConvTransE, we adapt the baseline ConvE and ConvTransE to use BERT embeddings in the same manner as our model. The convolution for BERT-ConvE has 32 channels and the convolution for BERT-ConvTransE has 64 channels. These values were selected to produce models with a comparable number of trainable parameters to our model. We then project the final feature vector down to the embedding dimensionality and rank candidates identically to our model.

We trained both models with a batch size of 64 using 1vsAll strategy (Ruffinelli et al., 2020) with the binary cross entropy loss function using the Adam optimizer (Kingma and Ba, 2015) with decoupled weight decay regularization (Loshchilov and Hutter, 2019). We train the models with a learning rate of 1e-4, use label smoothing with value 0.1, clip gradients to a max value of 1, and regularize the model using weight decay with a weight of 0.0001. We apply dropout with drop probability 0.2 after the embedding layer and after the convolution. We apply dropout with probability 0.3 after the fully connected layer.

For our baseline BERT-DeepConv model, we use the same hyperparamters as BERT-ResNet for the initial 1-D convolution and then apply a sequence of three 3×3 convolutions with circular padding. The second convolution doubles the number of channels so the dimensionality of the final feature map produced by the sequence of convolutions is $2d$. We then mean pool and project the feature map to the embedding dimensionality identically to our proposed model. We selected these hyperparameters so that this baseline has a similar number of trainable parameters to our proposed model. All other implementation details are identical to our BERT-Resnet model (e.g. use of pre-activations, application of dropout, training hyperparameters, etc.).

A.3 Re-Ranking

We fine-tune BERT with a learning rate of $3e{-}5$ using the Adam optimizer (Kingma and Ba, 2015) with decoupled weight decay regularization (Loshchilov and Hutter, 2019). We truncate the textual triple representation to a max length of 32 tokens and fine-tune BERT with a batch size of 128 for a maximum of 10 epochs. Training is terminated early if the validation MRR does not improve for 3 epochs. We set the weight decay parameter to 0.01 and clip gradients to a max value of 1 during training. We apply dropout with probability 0.3 to the final feature representation before the prediction and otherwise use the default parameters provided by the HuggingFace Transformers library (Wolf et al., 2020). We set $\lambda = 0.5$ for SNOMED CT Core, $\lambda = 1.0$ for CN-100K, and $\lambda = 0.75$ for FB15k-237 and FB15k-237-Sparse. We set the

temparature as $T = 1$ for all models.

B Evaluation Metrics

We provide a mathematical formulation for our evaluation metrics. If we denote the set of all facts in the test set as $\mathcal{T}$, then the Mean Rank (MR) is simply computed as

$$\text{MR} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} \text{rank}(x_i)$$

The Mean Reciprocal Rank (MRR) is computed as

$$\text{MRR} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} \frac{1}{\text{rank}(x_i)}$$

The Hits at k (H@k) is calculated as

$$\text{H@k} = \frac{1}{|\mathcal{T}|} \sum_{x_i \in \mathcal{T}} I[\text{rank}(x_i) \leq k]$$

where $I[P]$ is 1 if the condition P is true and is 0 otherwise. When computing $\text{rank}(x_i)$, we first filter out all positive samples other than the target entity x_i. This is commonly referred to as the filtered setting.

C Supplementary Tables

	SNOMED CT Core				
	MR	MRR	H@1	H@3	H@10
DistMult	5039	.294	.226	.319	.427
ComplEx	3850	.303	.225	.335	.457
ConvE	3618	.271	.191	.303	.429
ConvTransE	3484	.293	.216	.323	.446
BERT-ConvE	386	.384	.278	.431	.593
BERT-ConvTransE	487	.374	.274	.417	.569
BERT-DeepConv	250	.481	.376	.534	.687
BERT-ResNet	249	.493	.389	.546	.694

Table 4: Validation ranking results for SNOMED CT Core.

	CN-100K				
	MR	MRR	H@1	H@3	H@10
BERT-ConvE	283	.370	.253	.423	.606
BERT-ConvTransE	323	.381	.267	.430	608
BERT-DeepConv	261	.463	.342	.526	.705
BERT-ResNet	269	.463	.341	.53	.700

Table 5: Validation ranking results for CN-100K.

	FB15k-237				
	MR	MRR	H@1	H@3	H@10
BERT-ConvE	189	.308	.228	.334	.467
BERT-ConvTransE	208	.301	.224	.326	.449
BERT-DeepConv	186	.332	.251	.360	.490
BERT-ResNet	185	.351	.269	.384	.514

Table 6: Validation ranking results for FB15k-237.

	FB15k-237-Sparse				
	MR	MRR	H@1	H@3	H@10
DistMult	3034	.136	.093	.146	.227
ComplEx	3311	.134	.092	.144	.220
ConvE	2247	.158	.107	.166	.261
ConvTransE	2275	.154	.103	.163	.257
BERT-ConvE	412	.192	.128	.202	.321
BERT-ConvTransE	390	.192	.129	.204	.318
BERT-DeepConv	419	.193	.131	.203	.320
BERT-ResNet	412	.194	.131	.204	.321

Table 7: Validation ranking results for FB15k-237-Sparse.

	SNOMED CT Core			
	MR	MRR	H@1	H@3
BERT-ResNet	2	.698	.561	.787
+ Re-ranking + KD + TE	2	.822	.724	.901
	CN-100K			
	MR	MRR	H@1	H@3
BERT-ResNet	3	.648	.488	.758
+ Re-ranking + KD + TE	2	.668	.511	.780
	FB15k-237			
	MR	MRR	H@1	H@3
BERT-ResNet	3	.664	.523	.748
+ Re-ranking + KD + TE	3	.678	.539	.761
	FB15k-237-Sparse			
	MR	MRR	H@1	H@3
BERT-ResNet	3	.567	.407	.634
+ Re-ranking + KD + TE	3	.589	.427	.667

Table 8: Validation re-ranking results. We report metrics for the subset of queries where the retrieved entity is already in the top 10 entities because the re-ranking procedure leaves other rankings unchanged.