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

Ioana Marinescu
Princeton University
ioanam@

Levi Blinder
Princeton University
lblinder@

Tejas Gupta
Princeton University
tejasg@

Abstract

Most research in knowledge graph completion (KGC) focuses on artificially densely connected graphs, inconsistent with more sparsely connected graphs found in downstream use cases. (Lovelace et al., 2021) use a deep convolutional network with textual entity representation to generate potential candidate queries for KGC and then re-ranks those candidate queries using knowledge distilled from the original generation. This method demonstrates improvements to current KGC, particularly in sparse graphs. Our paper reproduces and makes two modifications to the (Lovelace et al., 2021) model. To improve textual entity generation, we modify the existing transformer to use Phrase-BERT (Wang et al., 2021) approach to produce more powerful phrase embeddings. In line with recent developments in using ranking models for information retrieval (Guo et al., 2020), we modify (Lovelace et al., 2021) re-ranking model to use additional pointwise, pairwise, and listwise loss functions. We find Phrase-BERT embeddings have lower accuracy and our alternative loss functions provide no improvement on existing loss objective. We discuss these results, offering possible theoretical explanations for these unexpected findings

1 Introduction

Knowledge graphs (KGs) are used to organize information about the world or a particular topic and are useful in many natural language processing tasks. Usually, a KG is a multi-relational directed graph $G = (E, R, T)$, where E are the set of entities (i.e. objects, events, situations or concepts), R are the sets of relations (i.e. the relationships between entities), and T are the sets of fact triplets. Each triplet fact consists of an entity e_1, a relation r, and a second entity e_2, such as $\{$Ottawa, capitalOf, Canada$\}$.

KGs, because of their rich structural information, play an important role in many downstream use cases such as question answering and search systems (such as Google Assistant or Siri), relation extraction (such as Google Knowledge Panels), and recommendation systems (such as Netflix or Spotify). However, the use of a knowledge graph is limited by its completion, that is how many of the entities have relations between them.

Knowledge graph completion (KGC) is the act of predicting the tail entity e_2 given the head entity e_1 and query relation r. To train models to achieve the task of KGC, many curated datasets are used to test performance, but these datasets overly simplify the task, and result in models that do not match with real production-ready knowledge graphs such as Freebase, YAGO and DBpedia.

Generally knowledge graph completion requires embedding the KG entities and relations into a continuous vector space before applying modelling and evaluation to embedded triplets (e_1, rel, e_2). More recently, (Malaviya et al., 2019) has introduced the tactic of using textual representations of entities to create semantically-relevant embeddings with BERT. (Lovelace et al., 2021) makes significant contributions to this overall method, using the BERT transformer to create entity embeddings at a large scale that are fine tuned on the specific knowledge graph, developing a deep convolutional architecture that utilizes those embeddings more effectively than previous models, and developing a re-ranking model that distills knowledge from the ranking model.

Our additional contributions can be summarized as follows:

1. Implementing entity embeddings using Phrase-BERT (Wang et al., 2021), which produces more semantically meaningful embeddings for KGs where the entities are

phrases.

2. Additionally, given the importance of candidate re-ranking to the improvements that Lovelace et al. demonstrate and that the authors acknowledge that the performance of the re-ranking model could be limited by the use of a pointwise loss function to generate the re-ranking of the top-k candidates, we implemented new loss functions for re-ranking. Specifically, we add (1) pointwise (2) pairwise and (3) listwise loss functions described in (Guo et al., 2020).

2 Related work

2.1 Knowledge Graph Completion

Knowledge graph completion (KGC) models generally learn embeddings for entities and relations based on known triplet facts. These embeddings are then used to score potential candidate triplets. Approaches to learning these embeddings can be separated into two categories: (1) neural (Socher et al., 2013),(Dong et al., 2014), (Dettmers et al., 2017), (Vashishth et al., 2020) and (2) non-neural (Nickel et al., 2016), (Trouillon et al., 2016), (Liu et al., 2017), (Sun et al., 2019). Most previous approaches are only effective on benchmark datasets which are denser than typical realistic KGs (Pujara et al., 2017).

One method of aiding KGC that doesn't rely on the KG density is the utilization of entity names or descriptions to distinguish between entities and give them richer embeddings. (Socher et al., 2013),(Xie et al., 2016),(Xiao et al., 2016). In particular, recent work generates entity embeddings based on the BERT embedding of the entity names/descriptions (Malaviya et al., 2019), (Yao et al., 2019).

(Lovelace et al., 2021)'s novel approach combines recent uses of BERT in this field, using a larger Ranking model that embeds entities based on fine-tuned bert embeddings as in (Malaviya et al., 2019) in conjunction with a smaller Reranking model that embeds triplets using directly fine-tuned BERT embeddings of their textual representations as in (Yao et al., 2019). In particular, the Reranking model setup is advantageous due to its directly fine-tuned BERT embeddings, and small implementation in (Lovelace et al., 2021) avoids the complexity issues faced by (Yao et al., 2019).

2.2 Entity embeddings

2.2.1 BERT as a Knowledge Base

(Petroni et al., 2019), (Jiang et al., 2020), (Rogers et al., 2020) found that pre-trained language models such as BERT can be useful for embedding KG entities since relational information between entities in pre-training corpora may be similar to relations in the specific KG due to the shared semantic meaning of the entity name/description. Consequently, the textual embeddings of entities generated by pre-trained LMs already contain relational information relevant to KG completion task. These models also use masked-language-model (MLM) as a fine-tuning objective with fill-in-the-blank prompts (e.g. "Dante was born in [MASK]") to specialize embeddings for the given KG. This work motivates (Lovelace et al., 2021) to utilize these models to develop entity representations that are well-suited for KGC and thus encode the phrases that denote the entities.

2.2.2 Phrase embeddings

The task of learning dense phrase representation has been approached by finding a composition function that combines component word embeddings together into a single phrase embedding. Past work implemented the function as a rule-based composition over word vectors (Yu and Dredze, 2015) or using recurrent models (Zhou et al., 2017) that use a pair-wise GRU model using datasets such as PPDB (Pavlick et al., 2015). Some other work involved learning task-specific phrase embedding for tasks such as semantic parsing (Socher et al., 2011), machine translation (Bing et al., 2015) and question answering (Lee et al., 2021). Many of these phrase embedding models use pre-trained language models to generate general word embeddings that are then modified and/or combined in some way to generate specific phrase embeddings (Zhou et al., 2017), (Lee et al., 2021), (Yu and Dredze, 2015). Recently, the pre-trained language models used for word embedding have been large scale such as the BERT model (Lee et al., 2021). A recent development, Phrase-BERT (Wang et al., 2021), fine tunes the BERT model to produce general-purpose phrase embeddings useful for any task, which makes it well suited for our KCG task and motivates our variation.

2.3 Re-Ranking

The re-ranking approach has been introduced by (Wang et al., 2011) and involves using computationally cheap model to initially develop a ranking and then using an expensive model on the top-k candidates to re-rank them. Though initially not used to improve accuracy but rather improve efficiency by limiting the number of ranking candidates (Matsubara et al., 2020), (Lovelace et al., 2021) demonstrates the increase in accuracy from re-ranking given the large task of KGC on a sparse graph when coupled with knowledge distillation. Our variation of the re-ranking model's loss function was motivated the importance of re-ranking in KGC for sparse but large graphs in the original paper.

2.4 Knowledge Distillation

The concept of knowledge distillation was introduced relatively recently in (Hinton et al., 2015) as a way to transfer knowledge from a trained larger Teacher model to a new smaller Student model, thereby functioning to compress models.

The concept of distillation is generalized at a theoretical level by (Lopez-Paz et al., 2015), including broadening the scope of use cases to include "Learning from Multiple Tasks", which well described the use of distillation in the current study.

Additionally, distillation in practice has been implemented in non-compression KGC scenarios by (Lopez-Paz et al., 2015) and (Li et al., 2017) which show the effectiveness of knowledge distillation for a Teacher model that has access to cleaned labels or additional relevant features to improve the performance of a Student model without access to this "privileged information".

Importantly, these results show improvement *only* in the Student model that is trained with knowledge from the Teacher model rather than showing an improvement in the overall performance of the entire system, which is demonstrated in (Lovelace et al., 2021).

2.5 Loss objectives

(Guo et al., 2020) discusses possible types of loss objectives and their advantages and disadvantages for ranking tasks. These types are pointwise, pairwise, and listwise. The pointwise loss reduces the task to a classification or regression problem; the pairwise loss considers the relative preference between the candidates to rank; the listwise loss considers permutations of all the candidates at the same time, i.e. preferences for the entire ranking. (Lovelace et al., 2021) uses a pointwise loss function in its re-ranking model and notes that function could be a limitation, which motivates our variation of alternative loss functions that take into account more information for re-ranking as well as relational information between candidates.

3 Datasets

(Lovelace et al., 2021) examine KGC in the realistic setting where KGs have many sparsely connected entities. They utilize a commonsense KG dataset: CN100K (Li et al., 2016) that has been used in past work and curate two additional sparse KG datasets containing biomedical: SNOMED CT Core based on SNOMED CT (Donnelly, 2006) and encyclopedic: FB15k-237-Sparse based on FB15k-237 (Toutanova and Chen, 2015) knowledge. In our reproduction and variations, we work only with CN100K and FB15k-237-Sparse datasets to minimize training time.

3.1 FB15k-237-Sparse

The FB15k-237 dataset (Toutanova and Chen, 2015) contains encyclopedic knowledge about the world, e.g. {Barack Obama, placeOfBirth, Honolulu}. Although the dataset is very densely connected, that density is artificial. (Lovelace et al., 2021) 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.

3.2 CN100K

ConceptNet (Speer and Havasi, 2013) is a KG that contains commonsense knowledge about the world 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 of KGC on sparse graphs.

4 Methods

4.1 Entity Ranking

(Lovelace et al., 2021) represent a KG as a set of entity-relation-entity facts (e_1, r, e_2). Given an

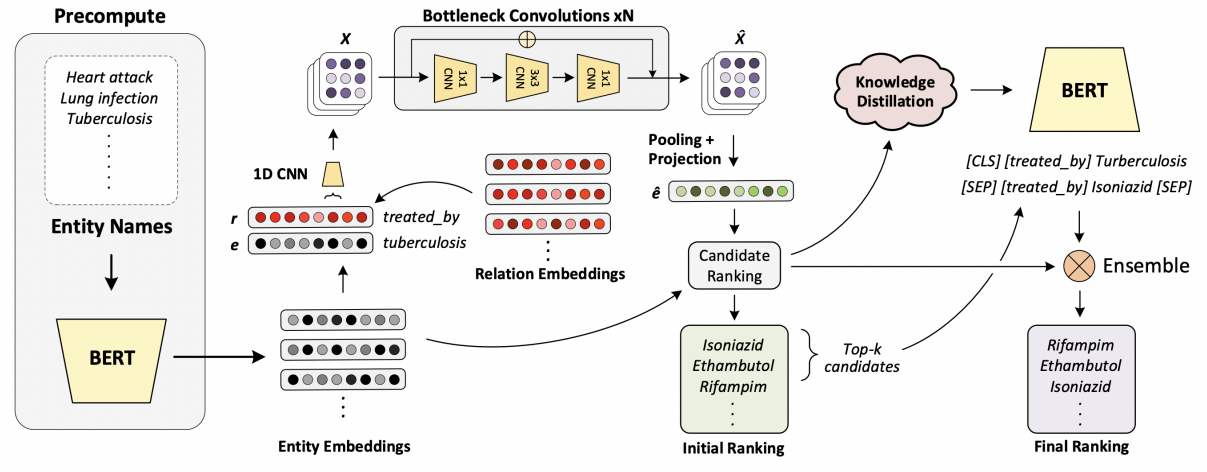

Figure 1: Model Architecture (Lovelace et al., 2021)

incomplete fact, $(e_1, r, ?)$, the model computes a score for all candidate entities e_i that exist in the graph (with a higher score assigned to correct entities).

4.1.1 Textual Entity Representation

In the original paper, the authors fine-tune BERT using the MLM objective with the set of entity identifiers in the KG to obtain entity embeddings.

Variation 1: We propose to replace this method of generating entity embedding by using "Phrase-BERT" (Wang et al., 2021) instead. This approach provides an advantage as the other baselines rely heavily on lexical overlap, but Phrase-BERT's nearest neighbors (embeddings that are similar) are both semantically similar and lexically diverse.

The novelty of Phrase-BERT consists in designing two separate fine-tuning tasks on top of BERT to improve its ability to produce meaningful phrase embeddings. Since the pre-trained BERT model relies heavily on lexical overlap to determine phrase similarity, the first fine-tuning objective relies on an automatically generated dataset of lexically diverse phrasal paraphrases to encourage the model to move beyond string matching. The second objective encourages the model to encode contextual information into phrase embeddings by relying on a phrase-in-context dataset extracted from the huge-scale Books3 corpus (Gao et al., 2021). Both cases rely on contrastive objectives similar to Sentence-BERT (Reimers and Gurevych, 2019) for fine-tuning. For example "Jack of all trade" and the paraphrase "a worker with versatile skills" are placed close in the embedding space.

The Phrase-BERT embeddings have been released through HuggingFace and we obtain the embeddings for the CN100K dataset using the sentence-transformers library. Since the CN100K dataset contains commonsense knowledge about the world, we hypothesized that using Phrase-BERT instead of MLM fine-tuned BERT would help the ranking ensemble perform better. We discuss the results in Section 6. However, due to the more general nature of Phrase-BERT we decided not to replace BERT with Phrase-BERT for the SNOMED CT Core (which contains medical domain-specific phrases with a standard meaning) and FB15K-237 (which contains textual longer descriptions as an entity) datasets.

4.1.2 Model architecture

After obtaining the entity embeddings, (Lovelace et al., 2021) stack the pre-computed entity embedding $e \in \mathbb{R}^{1\times d}$ with the learned relation embedding of the same dimension $r \in \mathbb{R}^{1\times d}$ to produce a feature vector of length d with 2 channels $q \in \mathbb{R}^{2\times d}$.

(Lovelace et al., 2021) apply 1D convolution with a kernel of width 1 and $f \times f$ filters along the length of the feature vector to produce a 2D spatial feature map $X \in \mathbb{R}^{f\times f\times d}$ with d channels, representing the incomplete query triple $(e_i, r_j, ?)$.

The spatial feature map, $X \in \mathbb{R}^{f\times f\times d}$, is analogous to a square image with a side length of f and d channels so they apply the ResNet (He et al., 2016) model which outputs a feature map $\hat{X} \in \mathbb{R}^{f\times f\times 4d}$. This feature representation is average pooled over the spatial dimension $(f \times f)$ which produces a summary feature vector $\hat{x} \in \mathbb{R}^{4d}$. The authors finally apply a fully connected layer followed by a PReLU nonlinearity to project the feature vector

back to the original embedding dimensionality d. The final vector is denoted $\hat{e}$ and the scores for candidate entities are computed by using the dot product with candidate entity embeddings. See Figure 1.

4.1.3 Training

For training, (Lovelace et al., 2021) use a 1vsAll training strategy with the binary cross-entropy loss function. The authors 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. Then, scores are computed for all entities as described in Section 4.1.2 and apply a sigmoid operator to induce a probability for each entity. They treat all entities other than e_k as negative candidates and then compute the binary cross-entropy loss.

4.2 Entity Re-Ranking

4.2.1 Re-Ranking Network

(Lovelace et al., 2021) use the ranking network to extract the top-k entities for every unique training query and then train a re-ranking network to rank these k entities. The student re-ranking network is designed 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 allows to directly fine-tune BERT during the training of the re-ranking network which often improves performance. See Figure 2.

They use relation tokens so ("head name", r_i , "tail name") would be represented as "`[CLS] [REL i]` head name `[SEP] [REL i]` tail name `[SEP]`". They use learned linear combination of the `[CLS]` embedding from each layer as the final feature representation for the prediction.

4.2.2 Knowledge Distillation

Since the ranking model takes incomplete facts as input while the re-ranking model takes complete facts as input, the models are inherently learning different tasks (i.e, the prediction of rankings based on an incomplete fact vs. based on a set of complete facts). While typical attempts at knowledge transfer by model weights would not work for this reason, the two models have the same target, so we can still distill knowledge of good prediction distributions. In particular, the re-ranking network makes predictions based on the top k selections of the ranking network and thus also has access to the ranking networks' knowledge in the form of its predictions.

The mechanism of knowledge distillation relies on a setup with two models, one "Teacher" model that has already learned a task and another "Student" model that aims to quickly learn the task via the Teacher's knowledge. We train the "Student" re-ranking model after the "Teacher" ranking model has been trained. For each input x_i and its true label y_i, we first turn the Teacher's predictions into normalized logits:

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

for a temperature T, which is used to create softer targets, increasing the Student model's sensitivity to correctly predicting classifications with lower probabilities (Hinton et al., 2015). We then calculate the loss for this input given the Student model output $f_S(x_i)$ as:

$$L_{KD}(y_i,x_i)=\lambda L(s_i,f_S(x_i))+(1-\lambda)L(y_i,f_S(x_i)) \quad (2)$$

In (Lovelace et al., 2021), this loss is a weighted average between typical (binary cross entropy) losses computed between the Student prediction and both the teacher's normalized logits, **s**(soft target), and the noisy training labels, **y** (hard target). Notice that for the BCE loss, equation (2) is equivalent to:

$$L_{KD}(y_i, x_i) = L_{BCE}((1-\lambda)y_i + \lambda s_i, f_S(x_i)) \quad (3)$$

This induces the Student model to have an objective function that pushes the Student predictions towards a target of $(1-\lambda)y_i+\lambda s_i$. This target maintains the distributional information learned by the ranking model, while still pushing the re-ranking model towards ranking the correct answers higher if possible.

4.2.3 Student-Teacher Ensemble

Since the re-ranking model is still operating with little information compared to the ranking model, it may make worse predictions at times. This issue is partially offset by the use of ensembling, which stabilizes the final model predictions as a linear combination of the ranking model and re-ranking model's predictions:

$$\hat{s}_{ik:(i+1)k} = \alpha(softmax(f_S(x_{ik:(i+1)k}))) + (1-\alpha)(softmax(f_T(x_i)_{0:k}) \quad (4)$$

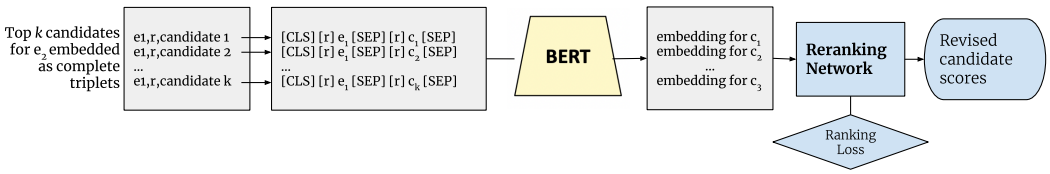

Figure 2: Illustration of re-ranking network in (Lovelace et al., 2021) and this paper

Here, α controls the impact of the Student re-ranking model. We sweep over $\alpha \in [0, 1]$ by increments of 0.1 during validation stage to find the re-ranking α that maximizes performance (achieves the best validation Mean Reciprocal Rank (MRR)).

4.2.4 Training

For our experiments, reproducing (Lovelace et al., 2021) 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, 2014) 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 MRR on validation data has not improved for 3 epochs.

Variation 2: The learning objective of neural ranking models can be broadly categorized into three groups: *pointwise, pairwise*, and *listwise*. See Figure 3 for an illustration of the differences. (Lovelace et al., 2021) use the Binary Cross Entropy (BCE) with logits loss, which is pointwise, for both the ranking and re-ranking. We modify the leaning objective for the re-ranking network to be of the pairwise and listwise (Guo et al., 2020) types, respectively. For the re-ranking, each data point is represented by $x_{i,j,k} = (e_i, r_j, e_k)$, a full candidate fact (where e_k is the candidate for entity 2 given (e_i, r_j)), and the true label $y_{i,j,k} = 1$ if e_k is a good candidate for $(e_i, r_j, ?)$ and $y_{i,j,k} = 0$ otherwise. Let $f(x_{i,j,k})$ be the output of the re-ranking network.

Pointwise Ranking Objective. The loss functions of pointwise learning objectives are computed based on each data point independently. The loss can be formulated as:

$$L(f, \vec{x}, \vec{y}) = \sum_{i,j} \sum_{k} L(f(x_{i,j,k}), y_{i,j,k}) \quad (5)$$

Pairwise Ranking Objective. Pairwise ranking objectives focus on optimizing the relative preferences between entities rather than their labels, more specifically considering how good is an entity 2 (k_1) compared to another entity 2 (k_2) for the triplet $(e_i, r_j, ?)$. In contrast to pointwise methods where the final ranking loss is the sum of loss on each triplet, pairwise loss functions are computed based on the permutations of all possible triplets. The loss can be formulated as:

$$L(f, \vec{x}, \vec{y}) = \sum_{i,j} \sum_{\substack{(k_1,k_2) \\ y_{i,j,k_1} \succ y_{i,j,k_2}}} L(f(x_{i,j,k_1}) - f(x_{i,j,k_2})) \quad (6)$$

where e_{k_1} is preferred to e_{k_2} as a candidate for entity 2 in $(e_i, r_j, ?)$, i.e. $y_{i,j,k_1} \succ y_{i,j,k_2}$.

Listwise Ranking Objective. Instead of comparing two candidates each time (like in the pairwise case), listwise loss functions compute ranking loss with each triplet and their candidate entity 2 e_2 list together. The loss can be formulated as:

$$L(f, \vec{x}, \vec{y}) = \sum_{i,j} L(\{f(x_{i,j,k}), y_{i,j,k} | e_k \in K\}) \quad (7)$$

where K denotes the set of possible candidates for e_2.

5 Experiments

First, we reproduce the main experiment of the paper using BERT-ResNet for the ranking network and knowledge distillation and re-ranking for the final student-teacher ensemble with pointwise loss objective (binary cross entropy). We present the results in Table 1.

5.1 Phrase-BERT embeddings

We replace the precomputed fine-tuned BERT embeddings of the entity names in (Lovelace et al.,

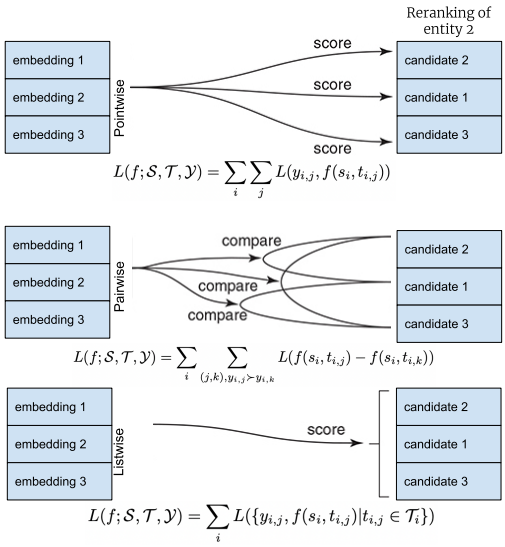

Figure 3: Illustration of re-ranking with pointwise, pairwise, and listwise losses

2021) with embeddings generated by Phrase-BERT of the entity names extracted from the CN100K dataset. We train the convolutional ranking network for 200 epochs and terminate early if the results have not increased for 20 epochs. We use hyperparameters `BATCH SIZE = 64`, `DROPOUT = 0.3`, `WEIGHT DECAY = 0.0001`, `LEARNING RATE = 0.001`. We present the results of the ranking network in Table 2.

5.2 Pointwise, Pairwise, and Listwise Loss Functions

We replace the BCE pointwise loss objective used by (Lovelace et al., 2021) with other objectives as described in Section 4.2.4. We experiment with the following losses:

- Pointwise: BCE, RMSE
- Pairwise: approxNDCG, lambdaLoss (Wang et al., 2018), rankNet, neuralNDCG (Pobrotyn and Bialobrzeski, 2021)
- binary ListNET , ListMLE (Xia et al., 2008), ListNET (Cao et al., 2007)

We use the AllRank (Pobrotyn et al., 2020) framework for the implementations of the above functions. We present the results in Table 3.

6 Results and Discussion

For evaluation, we use the following metrics: the mean rank (MR), which computes the arithmetic mean over all individual ranks, the mean reciprocal rank (MRR), which is the arithmetic mean of reciprocal ranks, and H(hits)@k, which describes the fraction of true entities whose rank is less than or equal to k (with 'filtering' such that this rank is calculated ignoring other true entities).

Table 2 presents the results for Variation 1, where we replaced the fine-tuned BERT embeddings for entities with embeddings generated by Phrase-BERT. The ranking trained with this change has worse performance than in (Lovelace et al., 2021). We hypothesize that this might be due to the fact that we did not fine-tune Phrase-BERT on our dataset and directly used the phrase embeddings for our entity names. While we expected the Phrase-BERT embeddings to be effective in CN-100K as its entities are "commonsense" phrases, it is reasonable to believe that the Phrase-BERT might generate more effective embeddings after being fine tuned on this specific KG similarly to the base BERT model. Further work could investigate how to appropriately fine-tune Phrase-BERT for KGC tasks.

	MR	MRR	H@1	H@3	H@5	H@10
Our Reproduction	413	.196	.134	.207	.254	.317
(Lovelace et al., 2021)	413	.198	.136	.211	-	.317

Table 1: Reproduction of (Lovelace et al., 2021) results using BERT-ResNet + Reranking + KD + TE on FB15k-237-Sparse

	MR	MRR	H@1	H@3	H@5	H@10
Our results with Phrase-BERT	282	.500	.375	.575	.645	.731
(Lovelace et al., 2021) with BERT	169	.550	.426	.628	-	.769

Table 2: Ranking Network on CN100K dataset using different types of embeddings for textual entity representation

	MR	MRR	H@1	H@3	H@5	H@10
BCE (Pointwise)	413	.196	.134	.207	.254	.317
RMSE (Pointwise)	413	.195	.133	.206	.252	.317
approxNDCG (Pairwise)	413	.192	.129	.203	249	.317
lambdaLoss (Pairwise)	413	.193	.130	.204	.251	.317
rankNet (Pairwise)	413	.192	.129	.203	.250	.317
neuralNDCG (Pairwise)	413	.195	.133	.205	.251	.317
binary ListNet (Listwise)	413	.195	.132	.206	.250	.317
ListMLE (Listwise)	413	.193	.130	.205	.250	.317
ListNET (Listwise)	413	.196	.133	.208	.254	.317

Table 3: Test Metrics for the BERT-ResNet + Reranking + KD + TE ensemble on FB15K-237-Sparse dataset with pointwise, pairwise, and listwise losses

Table 3 shows results for Variation 2, where we tried various loss functions for the re-ranking objective functions including pairwise and listwise

loss functions. While we theorized that pairwise and listwise loss functions might have a advantage due to their effectiveness for ranking applications where they can consider candidates together instead of independently, meaning they can assess actual rankings and assess relative differences between candidates.

Noting that a higher score indicates better performance on the task for all metrics aside from "MR", we see that these results go against the theorized advantage of pairwise or listwise loss functions. In particular, the pointwise BCE loss actually has the best performance across most metrics (aside from H@5, where it is outperformed marginally by ListNET). This result is unexpected for which we consider two possible causes.

First, the task may not be suited to pairwise and listwise functions. In particular, the task is not a natural ranking problem. The overall task for our system is to rank candidates, but the target output is simply an indicator of which candidates are correct as opposed to an actual ranking. Indeed while the re-ranking model does has a richer target output derived from knowledge distillation (which could be considered to induce a specific target ranking on the top k candidates), this target output is still not based on ranking data that in any way considers relative relevance between candidates as the outputs from the ranking model still independently assess each candidate.

Second, it is possible that our setup was not representative of the actual potential effectiveness of each loss function. In particular, we had to reformat our data for simultaneous input into loss functions in a way that may have affected the effectiveness of our gradient descent procedure (for example, since we had to group each set of top k candidate triplets, we may have lost some beneficial element of stochasticity that the original procedure had by ignoring these groupings). This hypothesis may be partially supported by the fact that our final implementation with BCE performed slightly worse that the original paper's implementation, which could suggest slightly worse training capabilities for our grouped data.

Additionally, the dataset we used, FB15K-237-SPARSE, was not the best representative of the task this model is intended to function well on. If we had more computation, we would have wanted to run these tests on the CN-100K or SNOMED CT Core database described in the original paper, but this would have taken about 12 hours/loss function, so it was unfeasible for our means.

7 Conclusion

We have replicated the work by (Lovelace et al., 2021) on Knowledge Graph Completion. First, we trained a ranking network based on BERT embeddings and a ResNet convolutional architecture on the FB15k-237-Sparse dataset. Second, we trained a re-ranking student network that re-ranked the top-k candidates resulted from ranking, using knowledge distillation. Our results are very similar to those from the original paper are presented in Table 1. We implement two variations. (1) We changed textual entity embeddings from being generated with BERT on an Masked Language Model (MLM) objective to being generated by Phrase-BERT and retraining the ranking network. We show these results in Table 2. We find that this change does not improve the results and performs worse. This could be due to not fine-tuning Phrase-BERT in a similar way that (Lovelace et al., 2021) fine-tuned BERT for their original approach. (2) We also implemented 9 different pointwise, pairwise, and listwise loss objectives for the re-ranking network. We found that the results were generally worse or the same as than the original baseline that used a pointwise loss (binary cross entropy), but we note that the listwise loss ListNET yields comparable results with BCE. One explanation for this result is the possibility of discrepancy based on our complete triplet grouping (see Appendix C) which might have a particular effect on the efficacy of Stochastic Gradient Descent.

Acknowledgments

We would like to thank Professor Narasimhan for his support in this class, as well as our advisor Tianyu Gao for his advice throughout the project. We also want to thank our friend Ian Nicolae for offering his with hardware to conduct our experiments. Lastly, we want to express our gratitude to Justin Lovelace, author of the paper we reproduced, for further explanations about the implementation of their experiments.

References

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca Passonneau. 2015. Abstractive multi-document summarization via phrase selection and

merging. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1587–1597, Beijing, China. Association for Computational Linguistics.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. volume 227, pages 129–136.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings.

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, page 601–610, New York, NY, USA. Association for Computing Machinery.

Kevin Donnelly. 2006. Snomed-ct: The advanced terminology and coding system for ehealth. *Studies in health technology and informatics*, 121:279–90.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network.

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.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6634–6647, Online. Association for Computational Linguistics.

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.

Yuncheng Li, Jianchao Yang, Yale Song, Liangliang Cao, Jiebo Luo, and Li-Jia 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. PMLR.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2015. Unifying distillation and privileged information.

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

Justin Lovelace, Denis Newman-Griffis, Shikhar Vashishth, Jill Fain Lehman, and Carolyn Penstein Rosé. 2021. Robust knowledge graph completion with stacked convolutions and a student re-ranking network. In *Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2019. Exploiting structural and semantic context for commonsense knowledge base completion. *CoRR*, abs/1910.02915.

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.

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, page 1955–1961. AAAI Press.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. 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.

Przemyslaw Pobrotyn, Tomasz Bartczak, Mikolaj Synowiec, Radoslaw Bialobrzeski, and Jaroslaw Bojar. 2020. Context-aware learning to rank with self-attention. *ArXiv*, abs/2005.10084.

Przemyslaw Pobrotyn and Radoslaw Bialobrzeski. 2021. Neuralndcg: Direct optimisation of a ranking metric via differentiable relaxation of sorting. *ArXiv*, abs/2102.07831.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how BERT works. *CoRR*, abs/2002.12327.

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, page 926–934, Red Hook, NY, USA. Curran Associates Inc.

Richard Socher, Cliff Chiung-Yu Lin, A. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *ICML*.

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*.

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, page 2071–2080. JMLR.org.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. 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.

Shufan Wang, Laure Thompson, and Mohit Iyyer. 2021. Phrase-bert: Improved phrase embeddings from bert with an application to corpus exploration. In *Empirical Methods in Natural Language Processing*.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Mike Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of The 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*, pages 1313–1322.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1192–1199, New York, NY, USA. Association for Computing Machinery.

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 *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2659–2665. AAAI Press.

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

Mo Yu and Mark Dredze. 2015. Learning Composition Models for Phrase Embeddings. *Transactions of the Association for Computational Linguistics*, 3:227–242.

Zhihao Zhou, Lifu Huang, and Heng Ji. 2017. Learning phrase embeddings from paraphrases with GRUs. In *Proceedings of the First Workshop on Curation and Applications of Parallel and Comparable Corpora*, pages 16–23, Taipei, Taiwan. Asian Federation of Natural Language Processing.

A Datasets

See figures below:

Dataset	#Nodes	#Relations	#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 4: Datasets Statistics (Lovelace et al., 2021)

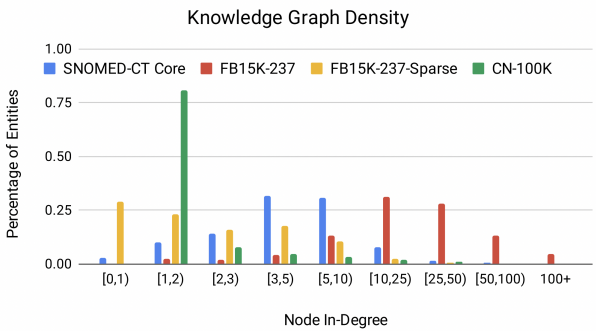

Figure 4: Degrees of entities in the training KGs (Lovelace et al., 2021)

B Training time

We performed our experiments using 2 GPUs: 1 Nvidia GeForce RTX 3060 Ti and 1 Nvidia GeForce RTX 3070 Ti.

The training time of the ranking model using Phrase-BERT and the convolutional architecture on the CN100K dataset is about 6.5 hours.

The total training time of the reranking network using all 9 types of losses described in Section 5.2 (1 at a time) on the FB15k-237-Sparse dataset is around 36h (an average of 4h/type of loss).

C Implementation

We release the implementation of the variants described in this paper at `https://github.com/ioanam25/robust-kg-completion`. The significant differences between the original implementation by (Lovelace et al., 2021) and ours are summarized as:

- (For the implementation of variant 1) During training, we grouped the top 10 (k) complete triplets output by the Ranking model for each incomplete query input. This let us input all 10 triplets to the model at the same time to then input their predictions and associated true labels to the loss functions together (which was necessary for the pairwise and listwise loss functions). We also reduced the batch size from 128 to 12 to account for the fact that each element of the batch was 10 triplets rather than 1.
- (For the implementation of variant 1) To actually implement this variant, we simply swapped the loss function used in Reranking from torch.nn.BCEWithLogitsLoss() to one of our loss functions implemented in the allRank framework (Pobrotyn et al., 2020).
- (For the implementation of variant 2) There were no actual changes to the code for this variant. Instead, we simply generated embeddings of all entities externally using Phrase-BERT, then swapped these in instead of the embeddings generated by the original paper's fine-tuned BERT model.

D Formula error

Our equations (2) and (3) use $1 - \lambda$, unlike the corresponding equations in the original paper, which use $\lambda - 1$ as we found their version to be incorrect (based on both theory and their code implementation).