On the Use of Entity Embeddings from Pre-Trained Language Models for Knowledge Graph Completion

Abstract

Recent work has found that entity representations can be extracted from pre-trained language models to develop knowledge graph completion models that are more robust to the naturally occurring sparsity found in knowledge graphs. In this work, we explore how to best extract and incorporate those embeddings. We explore the suitability of the extracted embeddings for direct use in entity ranking and introduce both unsupervised and supervised processing methods that can lead to improved downstream performance. We then introduce supervised embedding extraction methods and demonstrate that we can extract more informative representations. We also examine the effect of language model selection and find that the choice of model can have a significant impact. We then synthesize our findings and develop a knowledge graph completion model that significantly outperforms recent neural models. [1]

1 Introduction

Knowledge graphs (KG) are structured representations of knowledge that contain a collection of factual relations between entities. KGs have been shown to be useful in a variety of tasks such as representation learning (Liu et al., 2018), question answering (Sun et al., 2019a; Shen et al., 2019; Thirukovalluru et al., 2021), and entity linking (Thai et al., 2021).

However, the difficulty of curating knowledge at scale means that existing KGs are highly incomplete. This has led to the widespread study of knowledge graph completion (KGC) which aims to develop automated solutions that can suggest new facts to add to the KG (Yang et al., 2015; Trouillon et al., 2016; Shang et al., 2018; Dettmers et al., 2018; Sun et al., 2019b; Balazevic et al., 2019; Vashishth et al., 2020a). Much of the work in this area has been performed on a collection of benchmark datasets that are curated to have unusually dense connectivity. This simplifies the task but has also led to the development of KGC methods that are heavily reliant on that dense connectivity for strong performance (Pujara et al., 2017).

Recent work has begun to focus on more realistic settings where the KG does not exhibit dense connectivity. That work has demonstrated that textual entity embeddings can be extracted from pre-trained language models to develop KGC models with greater robustness to sparsity (Malaviya et al., 2020; Lovelace et al., 2021; Wang et al., 2021).

The most recent work (Lovelace et al., 2021; Wang et al., 2021) has fixed the textual entity embeddings during the training process. This reduces the reliance of the KGC model on existing knowledge within the graph and improves robustness to sparsity.

This prior work, however, diverged in their selection of language model, their method of extracting entity representations, and their use of the entity representations for candidate ranking. The work focused primarily on developing neural ranking architectures to effectively utilize the textual embeddings once they are extracted, leaving the effects of these divergent choices unclear.

In this work, we perform a comprehensive exploration of how to best extract entity representations from pre-trained language models and process them for use in downstream KGC architectures. We explore three primary research questions which we outline below.

RQ1: Is the textual embedding space sufficient for use in entity ranking? Mu and Viswanath (2018); Ethayarajh (2019); Li et al. (2020) have observed that textual embedding spaces tend to be highly anisotropic, with most vectors occupying a narrow cone within the space.

[1] We will make our code publicly available.

Furthermore, processing the word embeddings to be more isotropic, i.e. uniformly distributed with respect to direction, leads to significant improvements on semantic similarity benchmarks (Mu and Viswanath, 2018; Li et al., 2020). Given that entity ranking relies upon a similar measure of similarity, anisotropic embeddings could degrade KGC performance as well. We find that a similar problem does extend to KGC and introduce unsupervised and supervised approaches that transform the space to be more suitable for use in entity ranking.

RQ2: Can we extract more informative entity representations from pre-trained language models? Recent KGC work has extracted entity representations from pre-trained language models in an unsupervised manner. However, the knowledge for different downstream tasks is encoded differently by language models (Tenney et al., 2019; Toshniwal et al., 2020), suggesting that unsupervised extraction may be suboptimal. We explore supervised embedding extraction techniques to develop more informative entity representations.

RQ3: How sensitive is the downstream KGC performance to the selection of language model? We explore this question along two primary axis. First, we examine whether scaling up the language model leads to improved entity representations. Second, we additionally examine the effect of domain-specific pretraining. We find that while scaling up the language model can be helpful, domain specialization is particularly effective.

We synthesize our findings and utilize the most effective representation processing and extraction techniques with a recently proposed neural ranking architecture. Even though we make no modifications to the ranking architecture, our representation extraction and processing techniques lead to significant improvements across multiple datasets. The findings and analysis from this work provide useful guidelines for developing and utilizing effective textual entity representations for KGC.

2 Related Work

Malaviya et al. (2020); Lovelace et al. (2021); Wang et al. (2021) have found that pretrained language models can be used to extract entity representations to improve KGC in settings where the KG is highly incomplete and the existing knowledge is insufficient to learn meaningful entity representations.

Malaviya et al. (2020) and Wang et al. (2021) focused on commomsense KGC and developed methods utilizing graph neural networks in conjunction with shallow convolutional decoders. Lovelace et al. (2021) explored biomedical, encyclopedic, and commonsense KGC and introduced a deep convolutional model that outperformed existing shallow convolutional KGC architectures. All of these works focused on developing neural ranking architectures that used textual embeddings. We focus in this work on the complementary questions related to the extraction and use of entity representations.

Petroni et al. (2019) introduced the LAMA benchmark which utilizes fill-in-the-blank prompts to query the models for factoid knowledge. They found that language models are surprisingly effective at recalling relational knowledge even in a fully unsupervised setting. Follow-up work has found that language models are sensitive to the choice of prompt and that factual recall can be significantly improved with appropriate prompting (Jiang et al., 2020; Shin et al., 2020; Haviv et al., 2021). This motivates us to explore whether we can introduce supervision to extract embeddings that better represent the knowledge necessary for KGC.

3 Task Formulation

Given a set of entities $\mathcal{E}$ and relations $\mathcal{R}$, a KG can be defined as a collection of entity-relation-entity triplets $\mathcal{K} = \{(e_i, r_j, e_k)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ where $e_i, e_k \in \mathcal{E}$ and $r_j \in \mathcal{R}$. The aim of KGC is to develop a model that accepts a query consisting of a head entity and relation, $(e_i, r_j, ?)$, and ranks all candidate entities $e_k \in \mathcal{E}$ to resolve the query. An effective KGC model should rank correct candidates more highly than incorrect candidates.

Neural KGC models use the embedded head entity and relation to produce a query vector $f_\theta(\mathbf{e_i}, \mathbf{r_j}) = \mathbf{q}$ where $f_\theta(\cdot)$ is a neural network and $\mathbf{e_i}, \mathbf{r_j}, \mathbf{q} \in \mathbb{R}^d$. Scores for each candidate, $e_k \in \mathcal{E}$, are computed as the inner product between the query vector and the candidate entity embedding $y_k = \mathbf{q}\mathbf{e_k}^\intercal$ where $\mathbf{e_k} \in \mathbb{R}^d$. The line of work that we build on (Malaviya et al., 2020; Lovelace et al., 2021; Wang et al., 2021) uses textual descriptors to extract the entity embeddings from pre-trained language models.

We evaluate the KGC models with standard ranking metrics: Mean Reciprocal Rank (MRR), Hits at 1 (H@1), Hits at 3 (H@3), and Hits at 10 (H@10). We follow standard procedure and consider both forward and reverse relations and use the filtered evaluation setting (Bordes et al., 2013; Dettmers et al., 2018). We validate the significance of improvements in the MRR with paired bootstrap significance testing (Berg-Kirkpatrick et al., 2012) and correct for multiple hypothesis testing with the Benjamini/Hochberg method (Benjamini and Hochberg, 1995).

4 Datasets

We work with commonsense, biomedical, and encyclopedic KGC datasets. For the commonsense KG dataset, we work with the CN-82K dataset introduced by (Wang et al., 2021) which is derived from ConceptNet. For the biomedical KGC dataset, we work with the SNOMED-CT Core dataset introduced by Lovelace et al. (2021). For the encyclopedic dataset, we utilize the widely used benchmark KGC dataset, FB15k-237 (Toutanova and Chen, 2015). Dataset statistics are reported in Table 11.

5 RQ1: Sufficiency of Embedding Space for Entity Ranking

We evaluate whether the entity embeddings released by Lovelace et al. (2021) are well-suited for use in the candidate scoring process. In their work, Lovelace et al. (2021) used the entity embeddings directly in the entity ranking process following the standard neural KGC completion formulation. Thus the scalar score for entity i is $y_i = \mathbf{q}\mathbf{e_i}^\intercal$.

5.1 Embedding Space Analysis

5.1.1 Global and Local Cosine Similarity

We follow Ethayarajh (2019) and measure the anisotropy using the expected cosine similarity between randomly selected entities, i.e. $\mathbb{E}_{i,j\in|E|:i\neq j}[\cos(\mathbf{e}_i, \mathbf{e}_j)]$ where $\cos(\cdot,\cdot)$ denotes the cosine similarity. We would expect $\mathbb{E}_{i,j\in|E|:i\neq j}[\cos(\mathbf{e}_i, \mathbf{e}_j)] \approx 0$ in an isotropic space.

Recent work (Cai et al., 2021) has found that the embedding spaces from pre-trained language models contain embedding clusters that are locally isotropic when re-centered. We also compute the similarity metric after re-centering the embedding space to evaluate the local isotropy.

We report the expected cosine similarity for the entity embeddings released by Lovelace et al. (2021) in Table 1. The global cosine similarity is high across all datasets, but the similarity is near zero after re-centering. Therefore the global similarity arises from the presence of a large mean vector which is consistent with findings from past work (Mu and Viswanath, 2018; Cai et al., 2021).

We can examine the effect of this mean vector, $\mathbf{c} \in \mathbb{R}^d$. For a given query vector, $\mathbf{q_i}$, the score for some entity, e_j, can be decomposed as $y_j = \mathbf{q_i}(\mathbf{w_j}^\intercal + \mathbf{c}^\intercal) = \mathbf{q_i}\mathbf{w_j}^\intercal + \mathbf{q_i}\mathbf{c}^\intercal = \mathbf{q_i}\mathbf{w_j}^\intercal + b_i$ where $\mathbf{w_j}$ is an entity-specific vector and $b_i = \mathbf{q_i}\mathbf{c}^\intercal$ acts as a query-specific bias term. Perhaps surprisingly, the large mean vector actually has no effect on the relative rankings of the entities. We later explore experimentally whether this large mean vector degrades performance.

5.1.2 Effective Dimension

A complementary measure of anisotropy is the **ϵ-effective-dimension** from Cai et al. (2021). We first apply PCA to the matrix of entity embeddings. The ratio of the variance explained by k principal components can then be calculated as $r_k = \sum_{i=0}^{k-1}\sigma_i / \sum_{j=0}^{m-1}\sigma_j$, where σ_i is the i-th largest eigenvalue of the covariance matrix of the embeddings. The **ϵ-effective-dimension** is then $d(\epsilon) = \text{argmin}_k r_k \geq \epsilon$. We set $\epsilon = 0.8$ which means that $d(0.8)$ measures the minimum number of PCA components necessary to explain 80% of the variance in the embedding space.

We illustrate the value of this complementary measure of anisotropy with an example. Consider an embedding space that is normally distributed across a 2-dimensional plane centered in the d-dimensional embedding space. The expected cosine similarity would be 0, but the effective dimensionality of the embedding space would be $2 \ll d$ which would reveal the anisotropy. Our findings in Table 1 demonstrate that effective dimensionalities of the embeddings are far smaller than $d = 768$.

5.1.3 Knowledge Alignment

We measure the alignment between the embedding space and the KG. Past work (Zhang et al., 2020) has observed that for some set of facts $\{(e_i, r_j, e_k)\}_{k=1}^n$, we would expect $\{e_k\}_{k=1}^n$ to be similar in some way. For instance, all entities that satisfy the query $(\texttt{abdomen}, \texttt{finding\_site\_of}, ?)$ are abdominal conditions. The inner product scoring means that this similarity should be encoded within the entity embedding space to enable retrieving the set of correct entities with a single query vector.

To evaluate the alignment of the embedding space and the KG, we define the similarity between two entities as

Dataset	$\mathbb{E}[\cos(\cdot,\cdot)]$ Global	Local	$d(0.8)$	ρ
CN-82K	0.62	< 0.01	190	28.9
SNOMED-CT Core	0.81	< 0.01	110	22.6
FB15k-237	0.88	< 0.01	68	34.1

Table 1: Analysis of entity embedding spaces.

$\text{Sim}(e_i, e_j) = \sum_{e_k \in \mathcal{E}, r_l \in \mathcal{R}} \mathbb{1}(e_k, r_l, e_i) \times \mathbb{1}(e_k, r_l, e_j)$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, and $\mathbb{1}(e_k, r_l, e_i)$ evaluates to one if the fact is contained within the test set of the KG and zero otherwise. We report the knowledge aligment as the Spearman's rank correlation, ρ, between our KG-induced measure of similarity and the inner product between centered entity embeddings. Table 1 shows there is a significant ($p \ll 1\text{e}{-10}$) positive alignment across all datasets[2].

5.2 Embedding Processing Techniques

We explore unsupervised and supervised methods to improve the isotropy and alignment of the space.

5.2.1 Unsupervised Techniques

KGC training typically involves computing scores across all candidate entities (or at least a large sample of negative candidates) for each example. Therefore, unsupervised preprocessing techniques are more scalable than learning supervised transformations over the entire set of embeddings.

Normalization We normalize each entity embedding, $\mathbf{e_i} \in \mathbb{R}^d$, by centering the embedding space and reducing each vector to unit norm as $\tilde{\mathbf{e}}_\mathbf{i} = \frac{\mathbf{e_i} - \mathbf{c}}{\|\mathbf{e_i} - \mathbf{c}\|_2}$ where $\boldsymbol{c} \in \mathbb{R}^d$ is the mean of the entity embeddings.

Normalizing Flow We learn a normalizing flow to transform the anisotropic embedding space to an isotropic space, similar to Li et al. (2020). We briefly introduce normalizing flows, but we refer the reader to Papamakarios et al. (2021) for a comprehensive overview.

Normalizing flows can be used to transform an unknown distribution into a known probability distribution. Given $\mathbf{x} \in \mathbb{R}^d$ with an unknown true distribution $\mathbf{x} \sim p_x^*(\mathbf{x})$, we can define a joint distribution over $\mathbf{x}$ following the generative process of $\mathbf{x} = T(\mathbf{u}), \mathbf{u} \sim p_u(\mathbf{u})$ where $p_u(\mathbf{u})$ is called the base probability distribution of the flow model.

Normalizing flows constrain the transformation, T, to be a diffeomorphism which allows us to write the density of $\mathbf{x}$ in terms of $p_u(\mathbf{u})$ and the Jacobian determinant of T^{-1} as $p_x(\mathbf{x}) = p_u(T^{-1}(\mathbf{x}))|\det(J_{T^{-1}}(\mathbf{x}))|$. We can then fit the flow by minimizing the negative log-likelihood of observed samples $\{\mathbf{x}_n\}_{n=1}^N$ as $-\log(p_x(\mathbf{x})) = -\log(p_u(T^{-1}(\mathbf{x_i}))) - \log|\det(J_{T^{-1}}(\mathbf{x_i}))|$. If T^{-1}, $\det(J_{T^{-1}}(\cdot))$, and $p_u(\cdot)$ are tractable then gradient-based optimization is straightforward.

In this work we define T^{-1} as a linear projection followed by a scalar shift: $T^{-1}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ where $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^d$. To ensure the invertibility of $\mathbf{W}$ and to simplify the computation of the Jacobian determinant, we use the trick introduced by Kingma and Dhariwal (2018) and parameterize $\mathbf{W}$ using its LU decomposition.

We select a multivariate Guassian centered on the origin with an identity convariance matrix for the base distribution which provides a closed-form solution for $p_u(\mathbf{u})$. Thus the normalizing flow learns to transform the anisotropic entity embedding distribution to an isotropic Gaussian.

5.2.2 Supervised Techniques

We explore inexpensive supervised techniques that learn to transform the embedding space.

MLP We consider an MLP with one hidden layer followed by normalization. We process the set of entity embeddings by centering and scaling them to have unit norm before feeding them to the MLP. Thus a processed entity embedding, $\mathbf{e_i}$, is transformed as $\tilde{\mathbf{e}}_\mathbf{i} = \frac{MLP(\mathbf{e_i})}{\|MLP(\mathbf{e_i})\|_2}$.

Residual MLP We consider an MLP utilizing a residual connection with the original embedding. We similarly center and scale the embeddings to have unit norm. A processed entity embedding, $\mathbf{e_i}$, would then be transformed as $\tilde{\mathbf{e}}_\mathbf{i} = \frac{(\mathbf{e_i} + MLP(\mathbf{e_i}))}{\|(\mathbf{e_i} + MLP(\mathbf{e_i}))\|_2}$. Given the strong performance of the original set of embeddings, optimizing the residual mapping may be more effective.

5.3 Experiments

We conduct experiments to evaluate the different embedding processing methods. We utilize BERT-ResNet with the default hyperparameters from Lovelace et al. (2021) as our neural ranking architecture, $f_\theta(\cdot,\cdot)$. We only apply the transformation, $g_\theta(\mathbf{e_k}) = \tilde{\mathbf{e}}_\mathbf{k}$ where $\tilde{\mathbf{e}}_\mathbf{k} \in \mathbb{R}^d$, to the embedding matrix used for candidate ranking. Therefore, we still compute the query as $f_\theta(\mathbf{e_i}, \mathbf{r_j}) = \mathbf{q}$ with the original embeddings, but the score is now computed as $y_k = \mathbf{q}\tilde{\mathbf{e}}_\mathbf{k}^\intercal$.

[2] All rank correlations reported in this work are similarly significant ($p \ll 1\text{e}{-10}$). We omit future mentions of that significance for brevity.

	SNOMED CT Core				CN-82K				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Default Embeddings	.488	.383	.543	.689	.190	.127	.208	.314	.339	.259	.370	.500
Normalization	.487	.381	.544	.692	.192	.128	.211	.317	$.348^{**}$	.264	.381	.514
Normalizing Flow	**$.508^{**}$**	**.401**	**.566**	**.713**	**$.194^{*}$**	**.129**	**.213**	**.320**	**$.352^{**}$**	**.265**	**.385**	**.527**
MLP	$.539^{**\dagger}$	.431	.598	.749	$.200^{**\dagger}$	.132	.222	.339	$.374^{**\dagger}$	.282	.407	.561
Residual MLP	**$.549^{**\dagger}$**	**.445**	**.507**	**.752**	**$.209^{**\dagger}$**	**.138**	**.230**	**.350**	**$.375^{**\dagger}$**	**.283**	**.408**	**.564**

Table 2: Comparison of embedding processing techniques. The highest metrics for unsupervised and supervised techniques are bolded. We indicate a significant improvement over the default embeddings with $*(p < 0.01), **\,(p < 5\text{e}{-6})$ and over the normalizing flow with $\dagger$ $(p < 1\text{e}{-5})$.

	SNOMED CT Core			CN-82K			FB15k-237		
	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ
Default Embeddings	0.91	110	22.6	0.62	190	28.9	0.81	68	34.1
Normalization	< 0.01	109	21.6	< 0.01	190	28.8	< 0.01	66	35.6
Normalizing Flow	< 0.01	545	14.3	< 0.01	554	12.7	< 0.01	544	10.3
MLP	0.67	135	26.2	0.45	130	32.3	0.44	118	40.4
Residual MLP	0.63	183	27.4	0.34	228	33.6	0.41	147	41.5

Table 3: Intrinsic evaluation of embedding processing techniques.

	SNOMED CT Core				CN-82K				FB15k-237			
	x-to-one		x-to-many		x-to-one		x-to-many		x-to-one		x-to-many	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Default Embeddings	.823	.925	.259	.495	.267	.410	.112	.213	.714	.827	.229	.389
Normalizing Flow	.831 +1.0%	.936 +1.2%	.280 +7.9%	.522 +5.5%	.272 +1.7%	.418 +2.0%	.118 +5.5%	.225 +5.7%	.714 +0.0%	.838 +1.3%	.245 +4.3%	.418 +7.7%
Residual MLP	**.852** +3.6%	**.950** +2.7%	**.329** +26.9%	**.576** +16.4%	**.294** +10.1%	**.462** +12.8%	**.128** +14.5%	**.246** +15.8%	**.729** +2.0%	**.843** +2.0%	**.272** +18.7%	**.465** +19.8%

Table 4: Effect of embedding transformations across relation types.

5.4 Impact Of Embedding Space Transformations

	SNOMED CT Core		CN-82K	
	Edit Distance	KG Alignment	Edit Distance	KG Alignment
Default	-23.0	22.6	-21.3	28.9
Residual MLP	-17.8	27.4	-20.2	33.6

Table 5: Effect of supervised transformation on rank correlation (ρ) with edit distance and the KG.

We report the effect of the different transformations on downstream performance in Table 2 and report the intrinsic embedding metrics in Table 3.

For the unsupervised techniques, the normalizing flow consistently leads to significant performance improvements. However, the simpler normalization technique is not as effective. The embedding metrics show that normalization reduces the global similarity but has limited effects on the other metrics, suggesting that the large common mean vector has a minimal impact on performance. This also suggests that the cosine similarity metric may not be very informative for KGC.

The normalizing flow reduces the global similarity, but it also dramatically increases the effective dimensionality and decreases the knowledge alignment of the space. This suggests that a tradeoff may exist between our measures of isotropy and knowledge alignment. Depsite that tradeoff, optimizing solely for isotropy is effective. This confirms that the anisotropy of the original space does harm performance.

For the supervised techniques, both the MLP and Residual MLP lead to significantly improved performance, with the Residual MLP consistently outperforming the MLP. Both transformations consistently improve the knowledge alignment of the embedding spaces. Compared to the MLP, the Residual MLP produces a more isotropic space with a greater effective dimensionality that is better aligned with the KG. The improvement in both the isotropy and the knowledge alignment of the embedding space from end-to-end supervision provides further evidence that they are desireable characteristics for candidate ranking.

We also compare the effect on KG alignment with lexical overlap. Although lexical overlap can be meaningful, it also likely introduces spurious signals. We report the Spearman's rank correlation, ρ, between the edit distance between entity names and the inner product between centered entity embeddings in Table 5. The Residual MLP strengthens the KG alignment while reducing the correlation with lexical overlap, suggesting that it learns to highlight relevant information while discarding spurious correlations.

5.5 Performance by Relation Type

If the similarity between entities that resolve the same query is not represented in the embedding space, then the model would struggle to handle queries that retrieve multiple entities. To evaluate whether our tranformations alleviate that weakness, we categorize relations with at least 300 training examples as either x-to-one relations or x-to-many relations by computing the average number of tail entities associated with each query for the relation. If the number is less than 1.5, then we categorize it as a x-to-one relation. If the number is greater than 3, then we categorize it as a x-to-many relation.

We report the metrics and relative improvements obtained by our transformations in Table 4. The relative improvement is greater for x-to-many relations across all transformations and datasets. Thus the tranformations improve the model's ability to handle queries that retrieve multiple entities.

6 RQ2: Embedding Extraction

Previous work that utilizes embeddings from language models diverge in their embedding extraction method. We explore the efficacy of the unsupervised representation extraction techniques used in prior work and additionally introduce supervised representation extraction techniques.

6.1 Embedding Extraction Techniques

6.1.1 Unsupervised Techniques

[CLS] Token: We extract the embedding of the `[CLS]` token from the final layer following prior work (Malaviya et al., 2020; Wang et al., 2021).

Mean Pooling: We mean pool across all tokens and layers following Lovelace et al. (2021).

MLM Pretraining: Recent work (Malaviya et al., 2020; Wang et al., 2021; Lovelace et al., 2021) has pretrained the language model using the MLM objective upon the set of entity names. We ablate the impact of this pretraining stage.

6.1.2 Supervised Techniques

Fine-tuning the language model is ineffective because the limited vocabulary of entities leads to rapid overfitting. We instead explore supervised representation extraction techniques that introduce supervision over the frozen language model.

Linear Probe: We learn a linear projection (Toshniwal et al., 2020) that is applied to every hidden state of the frozen model. We then max-pool across the tokens in each layer to produce a single feature vector for every layer. We aggregate these features using a learned linear combination across layers.

Prompting: We learn continuous prompts that we prepend to the language model inputs at every layer to prompt the frozen model (Li and Liang, 2021). We parameterize the prompt embeddings in a low-dimensional space and learn an MLP with one hidden layer to project them to the dimensionality of the language model. We extract the entity representation by mean pooling across all intermediate states in each layer and aggregate across layers with a learned linear combination.

6.2 Experiments

We conduct experiments to evaluate the different entity extraction techniques. To isolate the effect of the embedding extraction technique, we use the most effective unsupervised processing technique, the normalizing flow, for candidate ranking. For the unsupervised techniques, we therefore compute the query as $f_\theta(\mathbf{e_i}, \mathbf{r_j}) = \mathbf{q}$ and the score as $y_k = \mathbf{q}\tilde{\mathbf{e}}_\mathbf{k}^\intercal$.

The supervised techniques introduce an additional function, $h_\theta(e_i) = \hat{\mathbf{e}}_\mathbf{i}$ where $\hat{\mathbf{e}}_\mathbf{i} \in \mathbb{R}^d$, to extract entity representations for computing the query $f_\theta(\hat{\mathbf{e}}_\mathbf{i}, \mathbf{r_j}) = \hat{\mathbf{q}}$. The score is then computed similarly to the unsupervised setting as $y_k = \hat{\mathbf{q}}\tilde{\mathbf{e}}_\mathbf{k}^\intercal$.

6.2.1 Impact of Embedding Extraction Techniques

We report the KGC metrics in Table 6 and report the intrinsic embedding metrics in Table 7. For unsupervised embedding extraction, the MLM pretraining improves downstream performance. That improvement corresponds to an improved KG alignment which outweighs a minor reduction in the effective dimensionality.

The optimal unsupervised extraction technique varies based on the dataset and that variance is reflected in the embedding metrics. For instance, the mean-pooled embeddings have far greater effective dimensionalities for the SNOMED CT Core dataset and the CN-82K dataset and lead to the strongest downstream performance. For the FB15k-237 dataset, however, the `[CLS]` embeddings have the greatest effective dimensionality and lead to the strongest performance. The supervised embedding extraction techniques do lead to improved performance over the unsupervised baselines, although we do not observe a clear winner among them.

	SNOMED CT Core				CN-82K				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Unsupervised Extraction Techniques												
CLS Token	.472	.371	.521	.671	.157	.104	.171	.259	.351	.266	.383	.525
+ Pretraining	$.489^{***}$	.385	.540	.695	$.189^{***}$	.126	.207	.314	$\mathbf{.356}^{***}$	**.270**	**.388**	**.530**
Mean Pooling	.503	.397	.559	.705	.184	.124	.202	.303	.352	.266	.385	.525
+ Pretraining	$\mathbf{.509}^{***}$	**.403**	**.566**	**.713**	$\mathbf{.195}^{***}$	**.130**	**.216**	**.323**	.352	.265	.385	.527
Supervised Extraction Techniques												
Linear Probe	$.516^{\dagger\dagger\dagger}$	.408	.575	**.722**	.195	.130	.215	.324	$.358^{\dagger}$	.272	.392	.530
+ Pretraining	$\mathbf{.517}^{*\dagger\dagger\dagger}$	**.410**	**.576**	**.722**	$.199^{**\dagger\dagger}$	.133	.220	.329	$\mathbf{.359}^{\dagger\dagger}$	**.273**	**.392**	**.532**
Prompting	$.515^{\dagger\dagger\dagger}$	**.410**	.573	.719	$.201^{\dagger\dagger\dagger}$	.136	.222	.333	.357	.271	.392	.528
+ Pretraining	$.513^{\dagger\dagger\dagger}$	.406	.571	.718	$\mathbf{.202}^{\dagger\dagger\dagger}$	**.137**	**.223**	**.335**	.356	.271	.387	.525

Table 6: Comparison of embedding extraction techniques. We indicate significant improvements from the pretraining procedure with $*(p < .05), **\,(p < .01), ***(p < 5\mathrm{e}{-5})$ and over the best unsupervised approach with $\dagger(p < .05), \dagger\dagger\,(p < .005), \dagger\dagger\dagger, (p < 5\mathrm{e}{-6})$.

	SNOMED CT Core			CN-82K			FB15k-237		
	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ
CLS Token	0.93	112	21.1	0.82	119	18.1	0.64	74	29.8
+ Pretraining	0.43	56	23.7	0.69	108	29.9	0.58	83	31.7
Mean Pooling	0.93	126	21.5	0.68	206	26.0	0.82	69	34.7
+ Pretraining	0.91	112	23.1	0.62	189	29.2	0.81	69	34.8

Table 7: Intrinsic evaluation of embedding extraction techniques.

	SNOMED CT Core				CN-82K				FB15k-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Unsupervised Embedding Extraction & Residual MLP												
BERT-base	.531	.425	.588	.736	.210	**.139**	.232	.352	.373	.282	.406	.559
BERT-large	$.545^{*}$	.441	.601	.749	**.212**	**.139**	**.234**	**.356**	**.375**	**.282**	**.410**	**.563**
PubMedBERT	$\mathbf{.549}^{\ddagger}$	**.444**	**.606**	**.754**	–	–	–	–	–	–	–	–
Prompting & Residual MLP												
BERT-base	.530	.423	.587	.736	$.214^{\dagger\dagger}$	.142	.237	.361	$\mathbf{.376}^{\dagger}$	**.284**	**.410**	**.562**
BERT-large	$.541^{*}$	.434	.599	.749	$\mathbf{.216}^{\dagger\dagger}$	**.144**	**.238**	**.361**	.373	.280	.409	.561
PubMedBERT	$\mathbf{.550}^{\ddagger}$	**.443**	**.611**	**.755**	–	–	–	–	–	–	–	–

Table 8: Effect of language model selection. We indicate significant improvements from the larger language model with $*(p < 5\mathrm{e}{-6})$; from prompting with $\dagger(p < 0.05), \dagger\dagger\,(p < 0.001)$; and from specialization with $\ddagger(p < 5\mathrm{e}{-6})$.

Language Model	$\mathbb{E}[\cos(\cdot,\cdot)]$	$d(0.8)$	ρ
BERT-base	65.6	132	18.0
BERT-large	62.6	135	18.6
PubMedBERT	90.7	112	23.1

Table 9: Analysis of SNOMED-CT Core embeddings.

7 RQ3: Language Model Selection

Further performance improvements can often be gained by scaling up the size of the language model Devlin et al. (2019) or from using specialized, domain-specific language models Gu et al. (2020). We examine the effect of those two aspects on downstream KGC performance in this section.

7.1 Experiments

To evaluate the potentially differential improvements across entity extraction techniques, we conduct experiments with both unsupervised and supervised extraction techniques while using our best candidate ranking approach, the Residual MLP. We conduct experiments with BERT-base and BERT-large for all three KGs using the uncased versions. To evaluate the effect of specialization, we use PubMedBERT, which is the same size as BERT-base, for the biomedical SNOMED-CT Core dataset.

7.2 Effect of Language Model Selection

We report results for these experiments in Table 8. When using unsupervised extraction techniques, the larger language model achieves better performance across all datasets, but the differences can be minor. For the supervised extraction techniques, the larger language model actually degrades performance over the unsupervised extraction techniques in some cases. The effect of using supervision for entity extraction and candidate ranking is dataset-dependent and is helpful for the CN82K dataset.

The mixed results from utilizing larger language models and introducing additional supervision could arise from an increased risk of overfitting. The supervised extraction and larger language

	SNOMED CT Core				CN-82K			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
BERT-ConvTransE (Malaviya et al., 2020)	–	–	–	–	.1626	–	.1795	.2751
InductivE (Wang et al., 2021)	–	–	–	–	.2035	–	.2265	.3386
BERT-DeepConv (Lovelace et al., 2021)	.479	.374	.532	.685	–	–	–	–
BERT-ResNet (Lovelace et al., 2021)	.492	.389	.544	.694	.190	.127	.208	.318
BERT-ResNet + Normalizing Flow	.509	.403	.566	.713	.195	.130	.216	.323
BERT-ResNet + Prompting + Normalizing Flow	.515	.410	.573	.719	.201	.136	.222	.333
BERT-ResNet + Residual MLP	.549	**.444**	.606	.754	.212	.139	.234	.356
BERT-ResNet + Prompting + Residual MLP	**.550**	.443	**.611**	**.755**	**.216**	**.144**	**.238**	**.361**

	FB15K-237				Additional Information	
	MRR	H@1	H@3	H@10	Graph Neighborhood	Text
RESCAL† Nickel et al. (2011)	.357	–	–	.541	✗	✗
TransE† Bordes et al. (2013)	.313	–	–	.497	✗	✗
DistMult† Yang et al. (2015)	.343	–	–	.531	✗	✗
ComplEx† Trouillon et al. (2016)	.348	–	–	.536	✗	✗
ConvE† Dettmers et al. (2018)	.339	–	–	.521	✗	✗
CompGCN (Vashishth et al., 2020b)	.355	.264	.390	.535	✓	✗
HittER (Chen et al., 2021)	.373	.279	.409	.558	✓	✗
BERT-DeepConv (Lovelace et al., 2021)	.327	.246	.354	.488	✗	✓
BERT-ResNet (Lovelace et al., 2021)	.346	.262	.379	.514	✗	✓
BERT-ResNet + Normalizing Flow	.356	.270	.388	.530	✗	✓
BERT-ResNet + Prompting + Normalizing Flow	.357	.271	.392	.528	✗	✓
BERT-ResNet + Residual MLP	.375	.282	**.410**	**.563**	✗	✓
BERT-ResNet + Prompting + Residual MLP	**.376**	**.284**	**.410**	.562	✗	✓

Table 10: Comparison against baseline methods and recent work. We indicate that the results are from the comprehensive replication study by Ruffinelli et al. (2020) with a †. Other results are taken from the original work.

models do lead to lower training loss, but that improvement does not consistently translate to stonger test performance.

Domain-specific pretraining is particularly effective, with PubMedBERT consistently outperforming other models. Table 9 shows that the biomedical language model is better-aligned with the biomedical KG, albeit with slightly lower effective dimensionality. This suggests that our proposed KG alignment metric may provide insight into the suitability of a language model *a priori*.

8 Comparison Against Recent Work

We synthesize our findings to develop a KGC model and compare against recent work. We again simply repurpose the BERT-ResNet ranking architecture with the default hyperparameters from Lovelace et al. (2021) to demonstrate the impact of the decisions explored in this work.

We report results across the two sparser datasets in Table 10. Our embedding extraction and processing techniques outperform recent work, with the supervised techniques being particularly effective. In Table 10 we compare against a selection of baselines on the FB15K-237 dataset. We also denote whether the models utilize additional graph information or textual information.

Our KGC model is very effective and outperforms the models that do not incorporate any additional information. While this seems natural, this was not actually the case with the prior work by Lovelace et al. (2021). Therefore, our method integrates textual information a way that improves performance even for densely-connected KGs.

Our KGC model also outperforms the recent work by Chen et al. (2021) which obtained strong performance gains from incorporating the graph neighborhood into the ranking decision. A natural extension is to explore how to incorporate both textual and graph information.

9 Conclusion

In this work, we have explored various techniques to improve the suitability of entity embeddings for candidate ranking (Section 5), explored different methods to extract entity embeddings from language models (Section 6), and have explored the effect of language model selection (Section 7).

By synthesizing the insights from our research questions, we were able to develop a KGC model that significantly outperforms recent work without making any modifications to the neural ranking architecture. The findings and analysis from this work provide a useful framework for evaluating and selecting effective entity representations for KGC. Our work also demonstrates the necessity of carefully controlling for choices regarding entity embeddings when conducting work in this area.

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.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate - a practical and powerful approach to multiple testing. *J. Royal Statist. Soc., Series B*, 57:289 – 300.

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.

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.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *International Conference on Learning Representations*.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. HittER: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10395–10407, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. 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 55–65, Hong Kong, China. Association for Computational Linguistics.

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.

Adi Haviv, Jonathan Berant, and Amir Globerson. 2021. BERTese: Learning to speak to BERT. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3618–3623, Online. Association for Computational Linguistics.

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 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*.

Durk P Kingma and Prafulla Dhariwal. 2018. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2395–2405, Melbourne, Australia. Association for Computational Linguistics.

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 Rosé. 2021. Robust knowledge graph completion with stacked convolutions and a student re-ranking network. 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 1016–1029, Online. Association for Computational Linguistics.

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*.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

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.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. 2021. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

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.

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.

Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. 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 2442–2451, Hong Kong, China. Association for Computational Linguistics.

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

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019a. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. 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 2380–2390, Hong Kong, China. Association for Computational Linguistics.

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

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Dung Thai, Raghuveer Thirukovalluru, Trapit Bansal, and Andrew McCallum. 2021. Simultaneously self-attending to text and entities for knowledge-informed text representations. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 241–247, Online. Association for Computational Linguistics.

Raghuveer Thirukovalluru, Mukund Sridhar, Dung Thai, Shruti Chanumolu, Nicholas Monath, Sankaranarayanan Ananthakrishnan, and Andrew McCallum. 2021. Knowledge informed semantic parsing for conversational question answering. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 231–240, Online. Association for Computational Linguistics.

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.

Shubham Toshniwal, Haoyue Shi, Bowen Shi, Lingyu Gao, Karen Livescu, and Kevin Gimpel. 2020. A cross-task analysis of text span representations. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 166–176, Online. Association for Computational Linguistics.

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, 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*.

Bin Wang, Guangtao Wang, Jing Huang, Jiaxuan You, Jure Leskovec, and C-C Jay Kuo. 2021. Inductive learning on commonsense knowledge graph completion. *International Joint Conference on Neural Networks (IJCNN)*.

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.

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*.

Zhanqiu Zhang, Jianyu Cai, and Jie Wang. 2020. Duality-induced regularizer for tensor factorization based knowledge graph completion. In *Advances in Neural Information Processing Systems*, volume 33, pages 21604–21615. Curran Associates, Inc.

A Dataset Information

We report the dataset statistics across all datsets used in this work in Table 11. For all three datasets, we utilize the textual descriptions used by Lovelace et al. (2021). For SNOMED CT Core and CN82k, these consist of short entity names. For FB15k-237, the descriptions are short paragraphs describing the entity. Unless otherwise stated, we utilize Pubmed-BERT to extract embeddings for the SNOMED CT Core dataset and utilize the uncased version of BERT-base for the other two datasets.

B Evaluation Metrics

We present a rigorous formulation of our evaluation metrics. We consider both forward and inverse relations for the datasets examined in this work. For the CN82k and FB15k-237 datasets, we follow standard procedure and introduce an inverse fact, (e_l, r_j^{-1}, e_i), for every fact, (e_i, r_j, e_l), in the dataset. The SNOMED CT Core dataset already contains inverse relations so manually adding inverse facts in unecessary. We let $\mathcal{T}$ denote the set of all facts in the test set.

The Mean Reciprocal Rank (MRR) is defined as

$$\text{MRR} = \frac{1}{|\mathcal{T}|} \sum_{(e_i, r_j, e_l) \in \mathcal{T}} \frac{1}{\text{rank}(e_l)}$$

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

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

where $I[P]$ is 1 if the condition P is true and is 0 otherwise. When computing 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. If the correct entity is tied with some other entity, then we compute its rank as the average rank of all entities with that score.

C Implementation Details

We outline our implementation details below. We begin by outlining the details shared across all experiments and then outline the details specific to the experiments performed for each of the three research questions.

C.1 Training Procedure

We train all ranking models for a maximum of 200 epochs and terminate training if the validation MRR has not improved for 20 epochs. We evaluate the model with the highest validation MRR upon the test set.

We use a batch size of 64 with the 1vsAll training 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). We set the learning rate to 1e-3 and set the weight decay coefficient to 1e-4. We reduce the learning rate by a factor of 0.5 if the validation MRR has plateaued for 3 epochs. We use label smoothing with a value of 0.1, clip gradients to a max value of 1.

C.2 BERT-ResNet

We reuse the reported hyperparameters from Lovelace et al. (2021) for the BERT-ResNet ranking architecture which we redescribe here. We set $f = 5$ where f is the hyperparameter that controls the side length of the spatial feature map produced by the initial 1D convolution. We set $N = 2$ where N controls the depth of the convolutional network. Our BERT-ResNet model then consists of $3N = 6$ bottleneck convolutional blocks. The dimensionality of the model is simply determined by the dimensionality of the language model, e.g. $d = 768$ for experiments with BERT-base and PubmedBERT and $d = 1024$ for experiments with BERT-large. We apply dropout with drop probability 0.2 after the embedding layer and apply 2D dropout (Tompson et al., 2015) with the same probability before the convolutions. We apply dropout with probability 0.3 after the final fully connected layer. These hyperparameter values are simply the default values reported by Lovelace et al. (2021).

C.3 RQ1: Sufficiency of Embedding Space for Entity Ranking

We describe implementation details pertinent to the experiments conducted in Section 5. To isolate the impact of the structure of the entity embedding space, we utilize a single shared bias term across all entities instead of the per-entity bias term utilized by Lovelace et al. (2021). Thus the entity ranking is determined entirely by the query vector and the entity embeddings. All future experiments also use this shared bias term.

For all of our embedding processing techniques, we decouple the entity embeddings fed to the convolutional model and the entity embeddings used for candidate ranking. All of our transformations

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
CN82K	78,088	34	100,000	1,200	1,200

Table 11: Dataset statistics

are only applied to the entity embeddings used for candidate ranking.

C.3.1 Normalizing Flow

We define the normalizing flow with the transformation $T^{-1}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ where $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{d}$[3]. To ensure the invertibility of $\mathbf{W}$ and to simplify the computation of the Jacobian determinant, we follow Kingma and Dhariwal (2018) and parameterize $\mathbf{W}$ using its LU decomposition. so $\mathbf{W} = \mathbf{PL}(\mathbf{U} + \text{diag}(\mathbf{s}))$ where $\mathbf{P} \in \mathbb{R}^{d \times d}$ is a permutation matrix, $\mathbf{L} \in \mathbb{R}^{d \times d}$ is a lower triangular matrix with ones on the diagonal, $\mathbf{U} \in \mathbb{R}^{d \times d}$ is a strictly upper triangular matrix, and $\mathbf{s} \in \mathbb{R}^{d}$ is a vector. During the training process, we fix $\mathbf{P}$ and learn the parameters for $\mathbf{L}$, $\mathbf{U}$, and $\mathbf{s}$.

We train the Normalizing Flow on the set of entity embeddings with a batch size of 64 for a maximum of 500 epochs using a learning rate of 1e-3 with the Adam optimizer (Kingma and Ba, 2015). We clip gradients to a max value of 1 and use the checkpoint that acheived the lowest training loss to transform the embeddings for candidate ranking. We normalize the transformed embeddings to have unit norm before use in candidate ranking so an entity embedding, $\mathbf{e_i}$, is transformed as $\tilde{\mathbf{e}}_\mathbf{i} = \frac{T^{-1}(\mathbf{e_i})}{\|T^{-1}(\mathbf{e_i})\|_2}$.

C.3.2 MLP and Residual MLP

For the supervised transformations, we set the dimensionality of the hidden layer to match the dimensionality of the entity embeddings. We use a ReLU nonlinearity and apply dropout with drop probability 0.1 after the first projection. We found it necessary to reduce the learning rate for the MLP to stabilize training so we set the learning rate to 1e-4 for the MLP parameters. All other hyperparameters remained fixed.

[3]This transformation consistently outperformed more expressive nonlinear flows (e.g. GLOW (Kingma and Dhariwal, 2018)) in our preliminary experiments. It's possible that a more comprehensive exploration of flow architectures and hyperparameter choices would lead to improvements over our design, but we leave such an exploration to future work.

C.4 RQ2: Embedding Extraction

We describe implementation details pertinent to the experiments conducted in Section 6. We use the HuggingFace Transformers library (Wolf et al., 2020) to work with pretrained language models. For this set of experiments, we utilize the normalizing flow technique for candidate ranking to isolate the effect of the extraction techniques. For the supervised extraction experiments, we utilize the most effective unsupervised embeddings with the normalizing flow for candidate ranking.

C.4.1 MLM Pre-training

We fine-tune the language models using the MLM pretraining objective over the set of textual entity identifiers. We fine-tune the language models for 3 epochs with a batch size of 32 and a learning rate of 3e-5. We use a linear learning rate warmup for first 10% of the total training steps. For SNOMED-CT Core and CN82K, we set the maximum sequence length to 64. For FB15k-237, we set the maximum sequence length to 256 to account for the longer entity descriptions. All other hyperparameters follow the default values from Huggingface.

C.4.2 Linear Projection

We learn a linear projection that is applied to every hidden state of the frozen model as $\tilde{\mathbf{h}}_{\mathbf{l,j}} = \mathbf{h_{l,j}}\mathbf{W}^\intercal + \mathbf{b}$ where $\mathbf{h_{l,j}} \in \mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d \times d}$, and $\mathbf{b} \in \mathbb{R}^d$. We then max-pool across every token in each layer to produce a single feature vector for each layer , $\tilde{\mathbf{h}}_\mathbf{l}$. and aggregate these features using a learned linear combination across layers $\tilde{\mathbf{e}}_\mathbf{i} = \sum_{l=1}^{L} \lambda_l \cdot \tilde{\mathbf{h}}_\mathbf{l}$ where $\lambda_l = \text{softmax}(\mathbf{a})_l$ and $\mathbf{a} \in \mathbb{R}^L$ is a learned vector of scalars. We set the learning rate for the parameters for embedding extraction to 5e-5.

C.4.3 Prompting

We learn continuous prompts that we prepend to the language model inputs at every layer to prompt the frozen model (Li and Liang, 2021). We parameterize the prompt embeddings, $\mathbf{p}_{i,j} \in \mathbb{R}^{d'}$, in a low-dimensional space where $d' < d$, and learn an MLP with one hidden layer to project them to the dimensionality of the language model. We set $d' = 256$ in this work and apply dropout with drop probability 0.1 before the MLP and after the first projection. The dimensionality of the hidden layer is set to $d/2$. We also apply a shared layer normalization layer to the output of the MLP.

Therefore the input to the i^{th} layer of the language model is $\mathbf{s}_i = [\text{LN}(\text{MLP}(\mathbf{p}_{i,0})), \ldots, \text{LN}(\text{MLP}(\mathbf{p}_{i,k})), \mathbf{x}_{i,0}, \ldots, \mathbf{x}_{i,n}]$ where $\text{LN}(\text{MLP}(\mathbf{p}_{i,j})) \in \mathbb{R}^d$ and $\mathbf{x}_{i,j} \in \mathbb{R}^d$ are the transformed prompt token and tokenized entity embedding respectively for the j^{th} position at the i^{th} layer. We use $k = 3$ prompt tokens across all experiments in this work. We extract the entity representation by mean pooling across all intermediate states in each layer and aggregate across layers with a learned linear combination. We set the learning rate for the parameters for embedding extraction to 5e-5.

C.5 RQ3: Language Model Selection

We describe implementation details pertinent to the experiments conducted in Section 7. For the unsupervised embedding extraction, we utilize mean-pooled embeddings from language models with additional MLM pretraining upon the set of entity names. For the prompting, we utilize the language model without any MLM pretraining. All other hyperparameters are kept constant from earlier sections.

D Validation Results

We report the validation results corresponding to the results reported in Table 10 in Table 12

	SNOMED CT Core				CN-82K				FB15K-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
BERT-ResNet + Normalizing Flow	.510	.403	.568	.714	.196	.133	.216	.323	.362	.279	.393	.529
BERT-ResNet + Prompting + Normalizing Flow	.517	.411	.574	.719	.202	.137	.223	.329	.361	.278	.394	.530
BERT-ResNet + Residual MLP	.551	.445	.608	.754	.213	.142	.235	.356	.378	.286	.414	.564
BERT-ResNet + Prompting + Residual MLP	.551	.444	.612	.757	.218	.146	.240	.363	.377	.287	.410	.564

Table 12: Validation results corresponding to results reported in Table 10.