A Framework for Adapting Pre-Trained Language Models to Knowledge Graph Completion

Justin Lovelace[*]
Computer Science Department
Cornell University
jl3353@cornell.edu

Carolyn Penstein Rosé
Language Technologies Institute
Carnegie Mellon University
cp3a@andrew.cmu.edu

Abstract

Recent work has demonstrated 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 conduct a comprehensive exploration of how to best extract and incorporate those embeddings into knowledge graph completion models. 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 that can extract more informative representations. 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 are valuable resources with applications in different areas such as representation learning (Liu et al., 2018), question answering (Sun et al., 2019; 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; Dettmers et al., 2018). KGC is typically formulated as ranking problem where an incomplete fact is used as a query to retrieve entities that complete the fact.

Recent work has utilized pre-trained language models to develop approaches that are more robust to the naturally occurring sparsity within knowledge graphs. These approaches utilize textual entity descriptions to develop entity representations that are less reliant on graph connectivity.

Such work either fine-tunes the language model directly during training to encode the entities (e.g. Yao et al. (2019)) or extracts a set of entity embeddings prior to training which can then be used to train a KGC model using standard training procedures (e.g. Lovelace et al. (2021)).

While fine-tuning language models often improves downstream performance (Rogers et al., 2020), it increases the computational overhead of computing entity representations. As a result, standard KGC training procedures that involve evaluating a large number of negative candidates for each positive instance are typically infeasible. Sampling only a small set of negative candidates enables training, but can harm performance.

Approaches that extract entity embeddings prior to training (Lovelace et al., 2021; Wang et al., 2021a) do not introduce any overhead for computing entity representations and are able to take advantage of standard training protocols. However, such approaches do not utilize any supervision to adapt the pre-trained language model to KGC.

While both lines of previous work have demonstrated their approaches effectiveness at retrieving sparsely connected entities, they still lag behind KGC models that do not incorporate any textual information on standard benchmark datasets.

In this work, we develop a framework for adapting pre-trained language models to KGC that takes advantage of the strengths of both prior lines of work. We accomplish this by decoupling the entity representations used for computing the query representation and the entity representations used for retrieval (see Figure 1).

For candidate ranking, we extract and cache en-

[*] Work conducted while at Carnegie Mellon University.

[1] https://github.com/justinlovelace/LM-KG-Completion

tity representations from a pre-trained language model prior to training. We then introduce lightweight unsupervised and supervised embedding processing techniques that improve the suitability of the space for candidate retrieval without sacrificing the scalability necessary to use standard KGC training procedures. The embedding processing techniques introduced in this work lead to significant performance improvements across datasets from diverse domains.

This decoupling also enables us to scalably fine-tune pre-trained language models to extract more informative entity representations for the query. However, naively fine-tuning the language model overfits the knowledge graph and actually degrades performance. We find that parameter-efficient fine-tuning methods such as prompt-tuning mitigate this and improve downstream performance.

We synthesize our findings and utilize the most effective candidate representation processing and entity extraction techniques with a recently proposed neural ranking architecture. Although we do not make any modifications to the ranking architecture, our representation extraction and processing techniques lead to significant improvements across four diverse datasets. The findings and analysis from this work provide useful guidelines for developing and utilizing effective textual entity representations for KGC.

The rest of our paper is organized as follows. We discuss related work in Section 2, present a formal description of our task in Section 3, and describe the datasets used in this work in Section 4. We introduce unsupervised and supervised techniques to improve the suitability of entity embeddings for candidate ranking in Section 5. We then introduce supervised methods to extract more informative representations for the query entity in Section 6 and explore the effect of language model selection in Section 7. Finally, we synthesize our findings in Section 8 and compare against recent work on our datasets. Our contributions are as follows.

- We develop a novel framework for adapting pre-trained language models for KGC that significantly improves performance for both sparsely connected and widely studied benchmark datasets.
- We demonstrate that the embeddings extracted from pre-trained language models are suboptimal for entity ranking and introduce unsupervised and supervised processing techniques that transform the textual embedding space to be more suitable for candidate retrieval.
- We demonstrate that parameter-efficient fine-tuning methods can be applied scalably to extract more informative query entity representations.

2 Related Work

Yao et al. (2019) adapted a pre-trained language model to KGC by fine-tuning it for triplet classification, i.e. predicting whether a given fact is true. However, such an approach scales poorly to the widely studied ranking formulation and is not competitive with simpler approaches.

Follow-up work has developed more scalable frameworks utilizing siamese encoders to independently encode the query and candidate entities (Wang et al., 2021b; Li et al., 2022; Daza et al., 2021). While this is an improvement, it still cannot scale to the tens of thousands of negative candidates typically considered during training. Clouatre et al. (2021) take a different approach and adapt the MLM objective to perform candidate retrieval by aggregating the logits for a number of mask tokens, eliminating the need to directly encode negative candidate entities. Although these approaches generally improve upon Yao et al. (2019), they still lag behind simpler models on standard benchmarks.

Malaviya et al. (2020); Lovelace et al. (2021); Wang et al. (2021a) have taken a different approach and extracted entity embeddings from pre-trained language models prior to training. This eliminates the overhead of computing entity representations during training, enabling the use of standard training procedures. The focus of this line of work has been on developing neural ranking architectures that can effectively utilize the extracted textual embeddings. We focus on the complementary questions of how to best extract and use entity representations with existing neural architectures.

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 a 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.

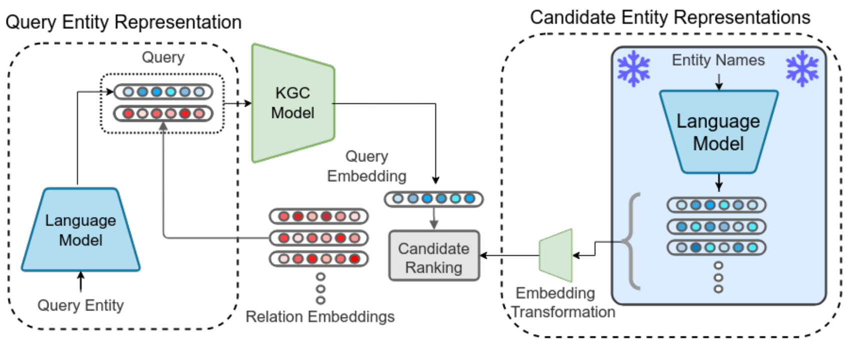

Figure 1: Overview of our proposed framework.

Neural KGC models embed the head entity and relation and compute 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 $\mathbf{e_k} \in \mathbb{R}^d$. We follow Lovelace et al. (2021) and use textual descriptors to extract the entity embeddings from pre-trained language models while learning relation embeddings during training.

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 (Dettmers et al., 2018). We validate the significance of improvements in the MRR with the 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 KGC datasets that cover diverse domains such as commonsense, biomedical, and encyclopedic knowledge. For the commonsense KG dataset, we work with the CN-82K dataset introduced by (Wang et al., 2021a) 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). We additionally utilize the widely studied WN18RR (Dettmers et al., 2018) dataset which is derived from WordNet. Dataset statistics are reported in the appendix in Table 7.

5 Candidate Retrieval

Mu and Viswanath (2018); Ethayarajh (2019); Li et al. (2020) have observed that textual embedding spaces tend to be highly anisotropic, i.e. most vectors occupy a narrow cone within the space, which limits their expressiveness. Furthermore, approaches that improve the isotropy, i.e. the uniformity with respect to direction, of the embedding space lead to significant improvements on semantic similarity benchmarks (Mu and Viswanath, 2018; Li et al., 2020; Gao et al., 2021). Given that entity ranking relies upon a similar scoring mechanism, the existing embedding space may be similarly sub-optimal for candidate retrieval.

5.1 Embedding Quality Metrics

We measure two primary aspects of the embedding space to analyze the effect of different processing techniques: the anisotropy of the space and the alignment of the space with the knowledge contained within the graph. We note that these aspects correspond to the notions of uniformity and alignment from work in constrastive learning (Wang and Isola, 2020; Gao et al., 2021).

5.1.1 Effective Dimension

We utilize a measure of anisotropy introduced by Cai et al. (2021) called the **ϵ-effective-dimension**. 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 we measure the minimum number of PCA components necessary to explain 80% of the variance in the embedding space.

5.1.2 Knowledge Alignment

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 example, all entities that satisfy the query (abdomen, `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

$$\mathrm{Sim}(e_i, e_j) = \textstyle\sum_{e_k \in \mathcal{E}, r_l \in \mathcal{R}} \mathbb{1}(e_k, r_l, e_i) \times \mathbb{1}(e_k, r_l, e_j)$$

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 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.

5.1.3 Lexical Alignment

As a complementary measure to knowledge alignment, we also measure the lexical alignment of the embedding space by calculating the Spearman's rank correlation, ρ, between the Jaccard Similarity of the entity descriptions and the inner product between centered entity embeddings.

5.2 Embedding Processing Techniques

5.2.1 Unsupervised Techniques

Normalization As a simple baseline, we normalize each entity embedding, $\mathbf{e_i} \in \mathbb{R}^d$, by centering the embedding space and scaling 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 a 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 the base probability distribution of the flow model.

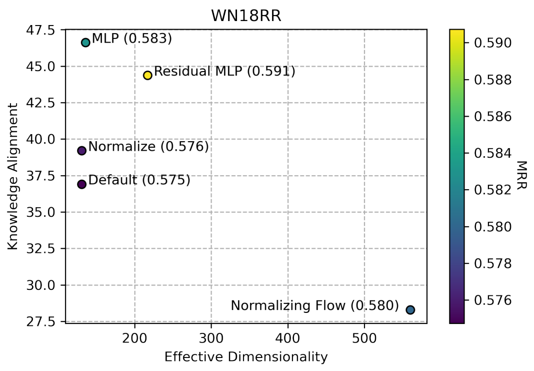

Figure 2: Intrinsic evaluation of embedding processing techniques. We note the MRR for each approach in parenthesis.

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

$$-\log(p_x(\mathbf{x_i})) = \\ -\log(p_u(T^{-1}(\mathbf{x_i}))) - \log|\det(J_{T^{-1}}(\mathbf{x_i}))|$$

We define $T^{-1}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ 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 parameterize $\mathbf{W}$ using its LU decomposition (Kingma and Dhariwal, 2018). We select a multivariate Guassian centered on the origin with identity convariance for the base distribution. Thus, the normalizing flow learns to map the embedding space to an isotropic Gaussian.

5.2.2 Supervised Techniques

We explore two inexpensive supervised techniques that learn to transform the embedding space. For both techniques, we preprocess the set of entity embeddings by centering and scaling them to have unit norm prior to the transformation.

MLP We consider an MLP with one hidden layer followed by normalization. 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 that uses a residual connection with the original embedding. A processed entity embedding, $\mathbf{e_i}$, would then be transformed as $\tilde{\mathbf{e}}_\mathbf{i} = \frac{(\mathbf{e_i} + MLP(\mathbf{e_i}))}{\|(\mathbf{e_i} + MLP(\mathbf{e_i}))\|_2}$.

	SNOMED CT Core				CN-82K				FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	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	.575	.503	.606	.716
Normalization	.487	.381	.544	.692	.192	.128	.211	.317	.348***	.264	.381	.514	.576	.501	.608	.726
Normalizing Flow	**.508*****	**.401**	**.566**	**.713**	**.194****	**.129**	**.213**	**.320**	**.352*****	**.265**	**.385**	**.527**	**.580***	**.509**	**.607**	**.721**
MLP	.539***†	.431	.598	.749	.200***†	.132	.222	.339	.374***†	.282	.407	.561	.583**	.510	.613	.730
Residual MLP	**.549*****†	**.445**	**.507**	**.752**	**.209*****†	**.138**	**.230**	**.350**	**.375*****†	**.283**	**.408**	**.564**	**.591*****†	**.518**	**.616**	**.735**

Table 1: Comparison of candidate transformation techniques. The highest metrics for unsupervised and supervised techniques are bolded. We indicate a significant improvement over the default embeddings with $*, **, ***(p < 0.05, 0.005, 5e{-}5)$ and over the normalizing flow with † $(p < 5e{-}5)$.

	SNOMED CT Core				CN-82K				FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
CLS Token	.472	.371	.521	.671	.157	.104	.171	.259	.351	.266	.383	.525	.549	.488	.567	.675
+ Pretraining	.489*	.385	.540	.695	.189*	.126	.207	.314	**.356***	**.270**	**.388**	**.530**	**.587***	**.515**	**.618**	**.732**
Mean Pooling	.503	.397	.559	.705	.184	.124	.202	.303	.352	.266	.385	.525	.577	.508	.603	.719
+ Pretraining	**.509***	**.403**	**.566**	**.713**	**.195***	**.130**	**.216**	**.323**	.352	.265	.385	.527	.580	.509	.607	.721

Table 2: Ablation of embedding extraction techniques. We indicate significant improvements from the pretraining procedure with $*(p < 5e{-}5)$.

5.3 Experiments

We evaluated the different embedding processing methods using the textual entity embeddings released by Lovelace et al. (2021)[2]. We also 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 compute the score as $y_k = f_\theta(\mathbf{e_i}, \mathbf{r_j}) g_\theta(\mathbf{e_k})^\intercal$.

5.4 Impact Of Embedding Space Transformations

We report the effect of the different transformations on downstream performance in Table 1 and display the intrinsic embedding metrics for WN18RR in Figure 2. Figures for the other datasets are presented in the appendix and show similar findings.

The normalization baseline is generally ineffective, which is consistent with its limited effect on the embedding metrics. The normalizing flow greatly increases the effective dimensionality but decreases the knowledge alignment of the space. This suggests that there may be a trade-off between isotropy and alignment of the space, which is consistent with observations from work in contrastive learning (Gao et al., 2021). Despite that trade-off, optimizing solely for isotropy significantly improves performance across all datasets, confirming that the anisotropy of the original space hurts performance.

For the supervised techniques, 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. Given its strong performance, the Residual MLP seems to best balance the trade-off between the knowledge alignment and isotropy of the embeddings.

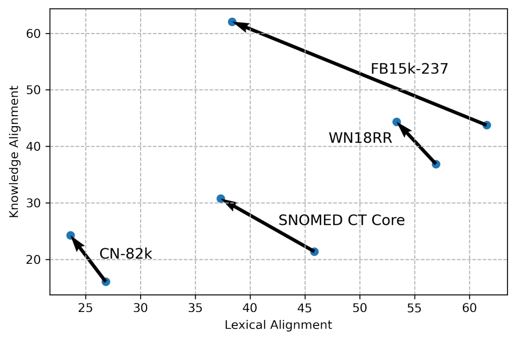

Figure 3: Effect Of Residual MLP on knowledge and lexical alignment.

We contrast the effect of the Residual MLP on knowledge and lexical alignment in Figure 3. The Residual MLP strengthens the KG alignment while reducing the lexical alignment across all datasets, demonstrating that it learns to emphasize relevant information while discarding spurious information.

5.5 Embedding Extraction Ablation

For this ablation, we used the most effective unsupervised processing technique, the normalizing flow, for candidate ranking. We ablate the efficacy of the following embedding extraction choices.

[CLS] Token: We extract the embedding of the

[2]Lovelace et al. (2021) did not work with WN18RR, so we developed embeddings following their procedure. We examine embedding extraction methods in detail in Section 5.5

	SNOMED CT Core				CN-82K				FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Unsupervised Extraction	.509	.403	.566	.713	.195	.130	.216	.323	.356	.270	.388	.530	.587	.515	.618	.732
Finetuning	.496	.386	.555	.709	.186	.124	.203	.307	.347	.260	.379	.522	.579	.509	.606	.721
Linear Probe	**.516**†††	.408	**.575**	**.722**	.195	.130	.215	.324	**.358**†	**.272**	**.392**	**.530**	**.598**††	**.524**	**.630**	**.746**
Prompt-tuning	.515†††	**.410**	.573	.719	**.201**†††	**.136**	**.222**	**.333**	.357	.271	**.392**	.528	.597††	.523	**.630**	.744

Table 3: Comparison of query entity extraction techniques. We indicate significant improvements over the best unsupervised approach with $\dagger, \dagger\dagger, \dagger\dagger\dagger (p < .05, 5e{-}4, 5e{-}5)$.

[CLS] token from the final layer following prior work (Malaviya et al., 2020; Wang et al., 2021a).
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., 2021a; Lovelace et al., 2021) has pretrained the language model using the MLM objective on the set of entity names. We ablate the impact of this choice.

We report the KGC metrics in Table 2. The MLM pretraining often results in significant improvements in downstream performance. The optimal unsupervised extraction technique varies based on the dataset, with mean-pooling being most effective for the SNOMED CT Core dataset and the CN-82K dataset while the [CLS] embedding is most effective for the other two datasets. However, we observe that mean pooling after MLM pre-training is reasonably effective across all datasets.

6 Query Entity Extraction

We explore supervised techniques to extract more informative representations from pre-trained language models for the query entity.

Fine-tuning: We fine-tune the language model during training and extract the entity representation by mean pooling across the intermediate states in each layer and aggregating across layers with a learned linear combination.

Linear Probe: We freeze the language model and apply a learned linear projection (Toshniwal et al., 2020) to every hidden state of the 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.

Prompt-tuning 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 extract entity representations by mean pooling across intermediate states in each layer and aggregate across layers with a learned linear combination.

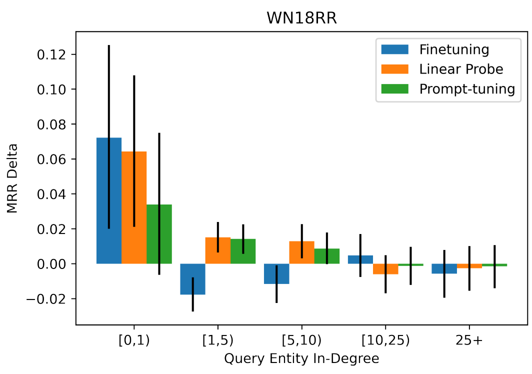

Figure 4: Effect of supervised extraction techniques compared to the unsupervised baseline. Error bars indicate 95% confidence intervals.

6.1 Experiments

To isolate the effect of the query embedding extraction technique, we use the normalizing flow for candidate ranking with the most effective embeddings from our prior ablation for each dataset.

The supervised extraction 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 $f_\theta(\hat{\mathbf{e}}_\mathbf{i}, \mathbf{r}_\mathbf{j}) = \hat{\mathbf{q}}$. Therefore, the score is computed as $y_k = f_\theta(h_\theta(e_i), \mathbf{r}_\mathbf{j}) g_\theta(\mathbf{e}_\mathbf{k})^\intercal$.

6.1.1 Impact of Embedding Extraction Techniques

We report the KGC metrics in Table 3. Fine-tuning the language model during training actually degrades performance across all datasets, although it does minimize the training loss more effectively than other approaches. We break down the effect of different techniques in Figure 4 by the connectivity of the query entity for the WN18RR dataset. We observe that the performance degradation is more pronounced for queries with lower connectivity although this degradation doesn't extend to unseen query entities. This suggests that fine-tuning the language model leads to overfitting for entities with limited information. The figures for the other datasets show similar trends and are presented in the appendix.

	SNOMED CT Core				CN-82K				FB15k-237				WN18RR			
	MRR	H@1	H@3	H@10	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	.590	.518	.616	.735
BERT-large	$.545^{**}$	.441	.601	.749	**.212**	**.139**	**.234**	**.356**	**.375**	**.282**	**.410**	**.563**	$\mathbf{.597}^{*}$	**.524**	**.624**	**.743**
PubMedBERT	$\mathbf{.549}^{\ddagger}$	**.444**	**.606**	**.754**	–	–	–	–	–	–	–	–	–	–	–	–
Prompt-tuning & Residual MLP																
BERT-base	.530	.423	.587	.736	$.214^{\dagger\dagger}$	.142	.237	.361	$\mathbf{.376}^{\dagger}$	**.284**	**.410**	**.562**	$.599^{\dagger}$	.525	.632	.749
BERT-large	$.541^{**}$	.434	.599	.749	$\mathbf{.216}^{\dagger\dagger}$	**.144**	**.238**	**.361**	.373	.280	.409	.561	$\mathbf{.608}^{**\dagger\dagger}$	**.538**	**.636**	**.751**
PubMedBERT	$\mathbf{.550}^{\ddagger}$	**.443**	**.611**	**.755**	–	–	–	–	–	–	–	–	–	–	–	–

Table 4: Effect of language model selection. We indicate significant improvements from the larger language model with $*, **$ $(p < .05, 5\text{e}{-5})$; from prompting with $\dagger, \dagger\dagger$ $(p < 0.05, .005)$; and from specialization with $\ddagger (p < 5\text{e}{-5})$.

The parameter-efficient supervised techniques do, however, lead to significantly improved performance across all datasets, although there is not a clear winner between them. These techniques mitigate the overfitting problem while enabling beneficial adaptation to the downstream task. Figure 4 shows that the benefits of supervision are greatest for sparsely connected query entities. For densely connected query entities, the impact is generally negligible, potentially because the graph already contains sufficient information about the entity.

We note that sparsely connected entities were filtered out of the FB15k-237 KG during the curation of the dataset, producing an artificially dense KG (Lovelace et al., 2021). This artificial density limits the benefit of techniques which improve performance for sparsely connected entities. Therefore, our analysis also explains the limited topline improvements for the FB15k-237 dataset.

7 Effect of 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). In this section, we examine the effect of these two aspects on downstream KGC performance.

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

We report the results of these experiments in Table 4. When using unsupervised extraction techniques, the larger language model consistently improves performance, 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 extracting the query entity is dataset-dependent and is helpful for CN82K and WN18RR.

The supervised extraction and larger language models do lead to lower training loss, but that improvement does not consistently translate to stonger test performance. Thus, the mixed results likely arise from overfitting which could potentially be mitigated with careful regularization. Domain-specific pretraining is particularly effective, with PubMedBERT consistently outperforming other models.

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 5. Our embedding extraction and processing techniques outperform recent work, with the supervised techniques being particularly effective. In Table 5 we also compare against a selection of baselines on the FB15K-237 and WN18RR datasets. 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. Although this seems natural, this was actually not the case with previous work. Therefore, our method integrates textual information in a way that leads to competitive performance even for these widely studied benchmark datasets.

	SNOMED CT Core				CN-82K				Additional Information
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	Text
DistMult (Yang et al., 2015)	.293	.226	.318	.426	.0280	–	.0290	.0560	✗
ComplEx (Trouillon et al., 2016)	.302	.224	.332	.456	.0260	–	.0270	.0500	✗
ConvE (Dettmers et al., 2018)	.271	.191	.303	.429	.0801	–	..0867	.1313	✗
BERT-ConvTransE (Malaviya et al., 2020)	–	–	–	–	.1626	–	.1795	.2751	✓
InductivE (Wang et al., 2021a)	–	–	–	–	.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 + Prompt-tuning + Normalizing Flow	.515	.410	.573	.719	.201	.136	.222	.333	✓
BERT-ResNet + Residual MLP	.549	**.444**	.606	.754	.212	.139	.234	.356	✓
BERT-ResNet + Prompt-tuning + Residual MLP	**.550**	.443	**.611**	**.755**	**.216**	**.144**	**.238**	**.361**	✓

	FB15K-237				WN18RR				Additional Information	
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	Graph Structure	Text
RESCAL† (Nickel et al., 2011)	.357	–	–	.541	.467	–	–	.517	✗	✗
TransE† (Bordes et al., 2013)	.313	–	–	.497	.228	–	–	.520	✗	✗
DistMult† (Yang et al., 2015)	.343	–	–	.531	.452	–	–	.531	✗	✗
ComplEx† (Trouillon et al., 2016)	.348	–	–	.536	.475	–	–	.547	✗	✗
ConvE† (Dettmers et al., 2018)	.339	–	–	.521	.442	–	–	.504	✗	✗
CompGCN (Vashishth et al., 2020)	.355	.264	.390	.535	.479	.443	.494	.546	✓	✗
HittER (Chen et al., 2021)	.373	.279	.409	.558	.503	.462	.516	.584	✓	✗
KG-BERT‡ (Yao et al., 2019)	.236	.145	.258	.420	.242	.110	.280	.524	✗	✓
BERT-TransE (Daza et al., 2021)	.235	.150	.253	.411	.325	.144	.431	.679	✗	✓
MLMLM (Clouatre et al., 2021)	.259	.187	.282	.403	.502	.439	.542	.611	✗	✓
StAR (Wang et al., 2021b)	.296	.205	.322	.482	.401	.243	.491	.709	✗	✓
LP-BERT (Li et al., 2022)	.310	.223	.336	.490	.482	.343	.563	.752	✗	✓
BERT-ResNet (Lovelace et al., 2021)	.346	.262	.379	.514	.575	.503	.606	.716	✗	✓
BERT-ResNet + Normalizing Flow	.356	.270	.388	.530	.587	.515	.618	.732	✗	✓
BERT-ResNet + Prompt-tuning + Normalizing Flow	.357	.271	.392	.528	.599	.527	.630	.743	✗	✓
BERT-ResNet + Residual MLP	.375	.282	**.410**	**.563**	.597	.524	.624	.743	✗	✓
BERT-ResNet + Prompt-tuning + Residual MLP	**.376**	**.284**	**.410**	.562	**.608**	**.538**	**.636**	.751	✗	✓

Table 5: Comparison against baselines and recent work. We indicate that the results are from Ruffinelli et al. (2020) with a † and from the work by Daza et al. (2021) with a ‡. The baselines for SNOMED CT Core and CN82K are taken from Lovelace et al. (2021) and Wang et al. (2021a) respectively, except for the BERT-ResNet result for CN82K which is from our implementation. The WN18RR result for BERT-ResNet is also from our implementation. Other results are taken from the original work. Dashes indicate that the metric was not reported by the prior work.

8.1 Complementarity of Textual Approach

To evaluate the complementarity of textual and non-textual approaches, we train a transformer model similarly to Chen et al. (2021). We refer the reader to the appendix for full details regarding this model. We then ensemble this model with our most effective model from Table 5, computing candidate scores as a convex combination of the two sets of scores. We tune the ensemble weight with the validation set. We also explore using an independent weight for each relation. As a baseline comparison, we ensemble our best configuration across two random seeds.

We report the results of this experiment in Table 6. We observe that ensembling is consistently effective, particularly the relation-specific ensembling. On the WN18RR dataset where the textual approach is already highly effective, ensembling the different approaches does not outpeform the self-ensemble. However, for the FB15k-237 dataset where the performance of the different approaches is closer, ensembling the textual and non-textual models does meaningfully improve performance over the self-ensemble. This demonstrates that textual approaches can complement existing methods.

	FB15K-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Transformer	.367	.272	.404	.554	.486	.446	.503	.564
Our Framework	.376	.284	.410	.562	.608	.538	.636	.751
Alt. Seed	.377	.285	.412	.564	.605	.533	.634	.749
Simple Ensemble								
Self-Ensemble	.384***	.292	.420	.570	.613***	.540	.641	.760
Transformer Ensemble	.388***†‡‡‡	.295	.425	.576	.609*	.539	.638	.755
Relation-Specific Ensemble								
Self-Ensemble	.391***	.303	.424	.571	**.616*****‡‡	**.544**	**.642**	**.758**
Transformer Ensemble	**.400*****†‡‡‡	**.310**	**.435**	**.582**	.612*‡	.543	.640	.756

Table 6: Ensembling Results. We indicate significant improvements over our framework with $*, **, ***(p < .05, 5e{-}4, 5e{-}5)$; from the transformer ensemble with $\dagger(p < 5e{-}5)$; and from relation-specific ensembling with $\ddagger, \ddagger\ddagger, \ddagger\ddagger\ddagger(p < .005, 5e{-}4, 5e{-}5)$.

9 Conclusion

We present a framework for adapting pre-trained language models for KGC. The key insight driving the development of our framework was that decoupling the entity representations used for computing the query representation and the entity representations used for candidate retrieval enabled us to better integrate the information from pre-trained language models while maintaining the scalability necessary to train performant KGC models.

We introduced unsupervised and supervised techniques to improve the suitability of entity embeddings for candidate ranking (Section 5), introduced methods to extract entity embeddings from language models (Section 6), and explored the effect of language model selection (Section 7).

By synthesizing the insights from our exploration, we developed a KGC model that significantly outperforms recent work while simply repurposing an existing ranking architecture. While innovations in neural ranking architecture have been valuable, our work demonstrates the importance of developing more informative entity representations. The findings and analysis from this work provide a useful framework for adapting pre-trained language models for knowledge graph completion.

10 Limitations

10.1 Training Overhead

We report and discuss the number of trainable parameters and training times across our different configurations in detail in the appendix [3]. We present the main takeaways in this section.

The supervised techniques like the Residual MLP and prompt-tuning introduce additional parameters and can increase the training time compared to the BERT-ResNet baseline. However, both the Residual MLP and prompt-tuning are very parameter-efficient. When utilizing BERT-base, the Residual MLP increases the number of trainable parameters by 3.6% and prompt-tuning increases it by 1.2%. The increases are similar when utilizing BERT-large (3.6% and 1.1% respectively). Directly fine-tuning BERT-base, for comparison, increases the number of trainable parameters by 331.2%.

The residual MLP, while lightweight, does increase the training time per iteration. For the candidate transformation experiment on the WN18RR dataset (Section 5), the baseline completes one epoch in 3m56s while the Residual MLP increases this to 5m44s. However, the Residual MLP also accelerates convergence, offsetting the per-iteration slowdown. Although it takes a similar amount of time to train the baseline for 6 epochs as it does to train the Residual MLP model for 4 epochs, the Residual MLP actually outperforms the baseline at that time despite training for fewer iterations.

Therefore, the baseline is only more effective in the earliest stage of training before being surpassed by the Residual MLP model. For the WN18RR dataset, this breakeven point occurs within only 29m of training. This trend holds across all datasets, with the worst breakeven point being only 1h43m. Therefore the accelerated convergence offsets the increased per-iteration cost for all but the shortest of training times.

Techniques such as prompt-tuning require the application of a language model, which increases the time per iteration. For the query extraction experiment on the WN18RR dataset (Section 6), the baseline completes one epoch in 3m54s, while prompt-tuning increases this to 8m47s. When controlling for wall clock time, we observe a similar trend where the baseline is more effective early in training before being surpassed by prompt-tuning. However, the breakeven point occurs much later (e.g. at 14h1m for WN18RR). Therefore, in settings with limited training budgets, the performance improvement from prompt-tuning may not justify the additional training cost.

We note that none of the techniques explored in our work introduce any overhead at inference time. After training, the improved entity representations from the Residual MLP or prompt-tuning can be computed and cached for inference, reducing the cost of computing entity embeddings to a simple lookup like the original BERT-ResNet model.

10.2 Availability of Textual Descriptions

The integration of pre-trained langauge models to improve KG entity representations is predicated upon the existence of informative textual descriptions for the entities within the graph. Although this assumption holds in many scenarios, it does not hold universally. For instance, clinical data from the Electronic Health Record can naturally be represented as a knowledge graph for applications such as question answering (Park et al., 2021).

Entities like medications and procedures would have well-defined names, but others such as those representing specific admissions events or hospital stays would be represented with a numerical ID and would not have natural textual representations. Although a hybrid approach that adaptively inte-

[3] All ranking models reported in this work were trained on a single NVIDIA GeForce GTX 1080 Ti.

grates textual information when available would likely be beneficial, the extension of our framework to such settings is left for future work.

11 Ethical Considerations

Knowledge graphs are valuable resources utilized by applications such as search engines (Sullivan, 2020) and automated voice assistants (Flint, 2021) to present information to users. While KGC models have the potential to improve the coverage of such resources, they also risk introducing inaccurate facts that could mislead users. The cost of such inaccuracies can vary significantly based on the information domain (e.g. film trivia vs. medical information).

Therefore, such tools should not be deployed without careful consideration of the potential harms or the development of appropriate mitigation strategies. One way to minimize such risks is to use KGC methods to accelerate the curation of likely candidate facts that must undergo further verification before their inclusion in the knowledge graph.

Acknowledgments

This research was funded in part by NSF grant IIS 1917955.

References

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.

Louis Clouatre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. 2021. MLMLM: Link prediction with mean likelihood masked language model. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4321–4331, Online. Association for Computational Linguistics.

Daniel Daza, Michael Cochez, and Paul Groth. 2021. Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*, WWW '21, page 798–808, New York, NY, USA. Association for Computing Machinery.

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.

Emma Flint. 2021. Alexa entities launches to general availability.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. 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.

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.

Da Li, Ming Yi, and Yukai He. 2022. LP-BERT: multi-task pre-training knowledge graph BERT for link prediction. *CoRR*, abs/2201.04843.

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.

Junwoo Park, Youngwoo Cho, Haneol Lee, Jaegul Choo, and E. Choi. 2021. Knowledge graph-based question answering with electronic health records. In *MLHC*.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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*.

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.

Danny Sullivan. 2020. A reintroduction to our knowledge graph and knowledge panels.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. 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.

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, and Partha Talukdar. 2020. 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. 2021a. Inductive learning on commonsense knowledge graph completion. *International Joint Conference on Neural Networks (IJCNN)*.

Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021b. Structure-augmented text representation learning for efficient knowledge graph completion. In *Proceedings of the Web Conference 2021*, WWW '21, page 1737–1748, New York, NY, USA. Association for Computing Machinery.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR.

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*.

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

A Dataset Information

We report the details for the datasets used in this work in Table 7. For SNOMED CT Core, CN82k, and FB15k-237 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 that describe the entity. For the WN18RR dataset, we utilize the entity descriptions released by Yao et al. (2019), which consist of the word and a short definition. Unless otherwise stated, we utilize PubmedBERT to extract embeddings for the SNOMED CT Core dataset and utilize the uncased version of BERT-base for the other three datasets.

B Evaluation Metrics

We present a mathematical 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 Model Configuration Details

C.1 Trainable Parameters

We report parameter counts for the WN18RR dataset across all the different configurations considered in this work in in Table 8. The parameter counts are identical across datasets with the exception of the relation parameters which depends upon the number of relations within each KG. The relation parameters make up a small portion of the overall parameters and are unaffected by the methods introduced in this work, so we simply report parameter counts for the WN18RR dataset for brevity.

The unsupervised Normalizing Flow technique can be applied prior to training and thus introduces zero additional trainable parameters for the ranking model. The supervised MLP and Residual MLP techniques introduce only 3.6% additional trainable parameters compared to the baseline model.

Directly fine-tuning the language model during training increases the number of trainable parameters by 331.2% because even the BERT-base model is over 3 times the size of the ranking model. The parameter-efficient methods, on the other hand, have a much more modest effect with the Linear Probe increasing the parameters by 3.0% and Prompt Tuning increasing the model size by 1.2%.

C.2 Training Time

We compare the training times across our different configurations. We report details for the candidate processing methods explored in Section 5 in Table 9. The normalizing flow technique has a negligible impact on training time because the unsupervised technique can be applied prior to training. The Residual MLP does increase the time per iteration as observed by the increased time needed to complete one epoch. However, the Residual MLP also accelerates convergence which largely offsets the aforementioned slowdown. Across all datasets, the Residual MLP outperforms the baseline even when controlling for wall clock time for all but the shortest of training times.

We report the training times for the query entity extraction methods explored in Section 6 in Table 10. The supervised methods introduce the application of a language model which also increases the time per iterations as seen by the time needed to complete one epoch. The effect on accelerating the convergence of the model is not as pronounced which means that in some cases, the supervised query extraction techniques do meaningfully increase the training time compared to the baseline.

Dataset	# Nodes	# Rels	# Train	# Valid	# Test
SNOMED-CT Core	77,316	140	502,224	71,778	143,486
CN82K	78,334	34	81,920	10,240	10,240
FB15K-237	14,451	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Table 7: Dataset statistics

Configuration	Trainable Params	Delta (%)
BERT-base		
BERT-ResNet	33.2M	-
+Normalizing Flow	33.2M	0%
+Fine-tuning	143.1M	331.2%
+Linear Probe	34.2M	3.0%
+Prompt Tuning	33.6M	1.2%
+MLP	34.4M	3.6%
+Residual MLP	34.4M	3.6%
+Prompt Tuning	34.8M	4.8%
BERT-large		
BERT-ResNet	58.9M	-
+Residual MLP	61.0M	3.6%
+Prompt Tuning	61.6M	4.7%

Table 8: Parameter Counts for WN18RR Models

D Additional Figures

D.1 Effect Of Embedding Processing Techniques

We report the embedding metrics across all datasets in Figure 5.

D.2 Effect Of Query Extraction Techniques

We report the performance of different query entity extraction techniques broken down by the connectivity of the query entity in Figure 6.

E 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 experiments.

E.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.

E.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).

E.3 Candidate Retrieval

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,

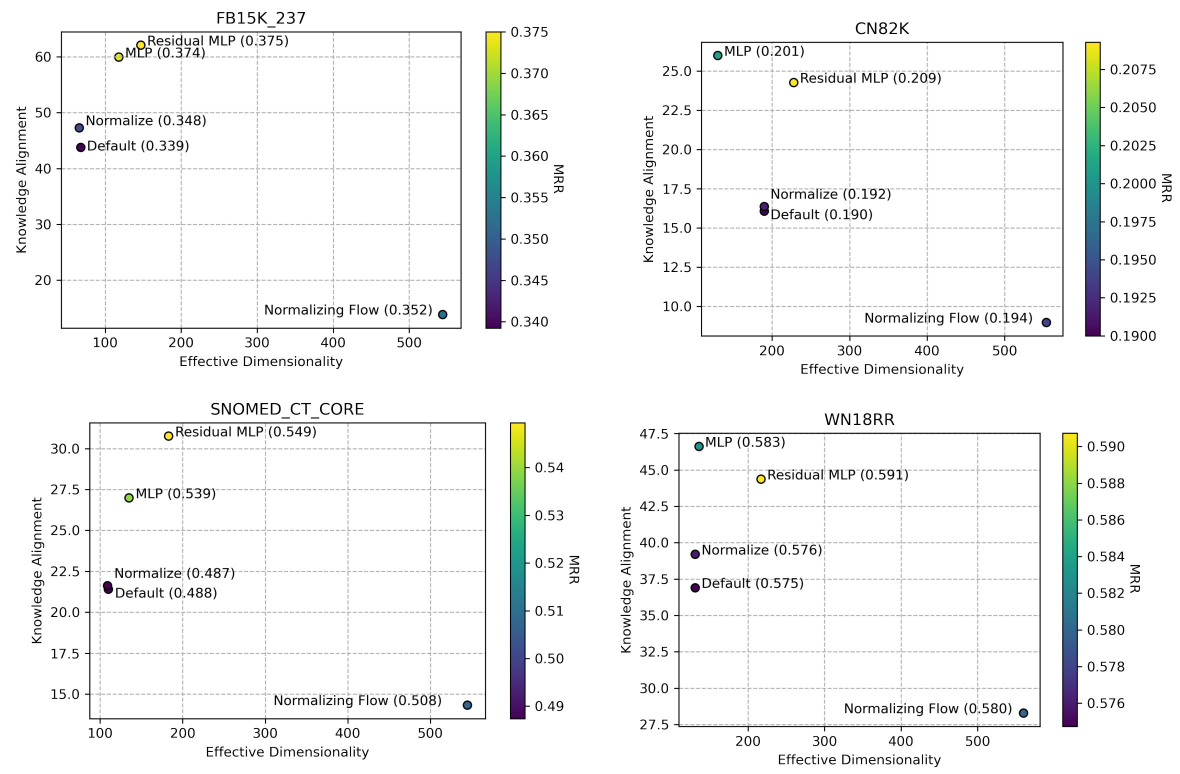

Figure 5: Intrinsic evaluation of embedding processing techniques. We note the MRR for each approach in parenthesis.

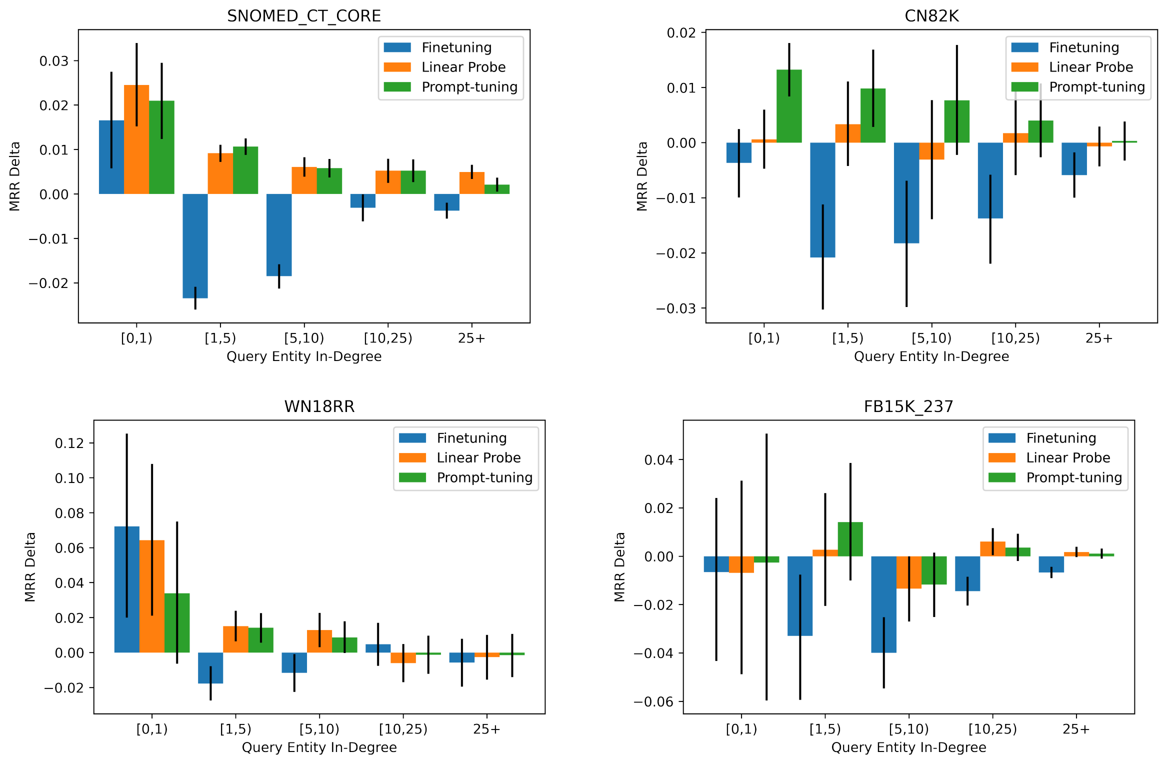

Figure 6: Performance delta of different extraction techniques across queries of varying connectivity. Error bars indicate 95% confidence intervals.

Configuration	SNOMED CT Core			CN-82K		
	Wall Clock Time			Wall Clock Time		
	Per Epoch	Best Validation MRR	Breakeven Point	Per Epoch	Best Validation MRR	Breakeven Point
BERT-ResNet	12m31s	22h57m49s	-	4m6s	5h33m38s	-
+Normalizing Flow	12m38s	28h38m39s	1h2m52s	4m7s	5h21m56s	1h6m55s
+Residual MLP	22m9s	52h5m26s	1h6m46s	7m20s	5h38m22s	1h43m9s
	FB15k-237			**WN18RR**		
Configuration	Wall Clock Time			Wall Clock Time		
	Per Epoch	Best Validation MRR	Breakeven Point	Per Epoch	Best Validation MRR	Breakeven Point
BERT-ResNet	11m52s	25h7m37s	-	3m56s	11h40m37s	-
+Normalizing Flow	11m50s	18h10m8s	35m32s	3m56s	10h10m7s	43m55s
+Residual MLP	14m31s	15h0m48s	14m31s	5m44s	11h5m2s	28m37s

Table 9: Run time for best supervised and unsupervised processing techniques from Section 5. We report the average wall clock time per epoch, the total time until the peak validation MRR, and the breakeven point where the configuration begins to outperform the baseline.

Configuration	SNOMED CT Core			CN-82K		
	Wall Clock Time			Wall Clock Time		
	Per Epoch	Best Validation MRR	Breakeven Point	Per Epoch	Best Validation MRR	Breakeven Point
BERT-ResNet	12m32s	34h17m41s	-	4m1s	5h25m20s	-
+Linear Probe	18m51s	42h8m20s	20h29m38s	6m25s	4h52m17s	4h52m17s
+Prompt-tuning	26m10s	60h48m38s	41h37m57s	8m42s	11h2m26s	5h40m58s
	FB15k-237			**WN18RR**		
Configuration	Wall Clock Time			Wall Clock Time		
	Per Epoch	Best Validation MRR	Breakeven Point	Per Epoch	Best Validation MRR	Breakeven Point
BERT-ResNet	11m52s	15h2m11s	-	3m54s	10h42m4s	-
+Linear Probe	23m19s	23h42m57s	N/A	6m3s	10h8m40s	6h36m41s
+Prompt-tuning	32m43s	47h22m32s	N/A	8m47s	20h36m0s	14h1m5s

Table 10: Run time for query entity extraction techniques from Section 6. We report the average wall clock time per epoch, the total time until the peak validation MRR, and the breakeven point where the configuration begins to outperform the baseline.

we decouple the entity embeddings fed to the convolutional model and the entity embeddings used for candidate ranking. All of our transformations are only applied to the entity embeddings used for candidate ranking.

E.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 $\mathbf{x}, \mathbf{b} \in \mathbb{R}^{d}$[4]. 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 $\tilde{\mathbf{e}}_\mathbf{i} = \frac{T^{-1}(\mathbf{e_i})}{\|T^{-1}(\mathbf{e_i})\|_2}$.

E.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

[4]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.

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. For the residual MLP, we also initialized the final linear layer to zeros so that the candidate embeddings were equivalent to the original embeddings at the start of training. All other hyperparameters remained fixed.

E.4 Embedding Extraction Ablation

We describe implementation details pertinent to the experiments conducted in Section 5.5. 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.

E.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, CN82K, and WN18RR 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.

E.5 Query Entity Extraction

E.5.1 Linear Projection

We learn a linear projection that is applied to every hidden state of the frozen model as $\tilde{\mathbf{h}}_{\mathbf{l},\mathbf{j}} = \mathbf{h}_{\mathbf{l},\mathbf{j}}\mathbf{W}^{\intercal} + \mathbf{b}$ where $\mathbf{h}_{\mathbf{l},\mathbf{j}} \in \mathbb{R}^{d}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$, 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.

E.5.2 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.

E.6 Effect of 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. All other hyperparameters are kept constant from earlier sections.

E.7 Ensembling

For our ensembling experiment, we train a transformer model that accepts a `[CLS]` token, the embedded query entity, and the embedded relation entity. This can be viewed as a simplified version of the HittER model from Chen et al. (2021) that doesn't utilize any additional graph context. The `[CLS]` embedding output from the final layer is used for candidate scoring.

We tune hyperparameters by running 20 trials of a random search over the grid of hyperparameters defined in Table 11. All models are trained for a maximum of 200 epochs with the AdamW optimizer. We linearly warm up the learning rate for the first 4000 steps before annealing it with a cosine decay schedule over the rest of training. We clip all gradient norms to 1 and apply early stopping with a patience of 50 epochs.

Hyperparameter	Search Range	Selected Value	
		FB15k-237	WN18RR
Learning Rate	[3e-3, 1e-3, 5e-4, 3e-4, 1e-4]	3e-4	3e-3
Weight Decay	[.3, .1, .03, .01, .001, 1e-4, 1e-5]	.01	0.1
Output Dropout	[.1, .2, .3, .4, .5, .6, .7]	.7	.5
Input Dropout	[.1, .2, .3, .4, .5, .6, .7]	.6	.5
Label Smoothing	[.1, .2, .3, .4, .5, .6]	.2	.2
Number Layers	[4,5,6]	6	5
Attention Heads	8	8	8
Embedding Dim	320	320	320
Feedforward Dim	1280	1280	1280

Table 11: Hyperparameter Search Space for Transformer Model

F Validation Results

We report the validation results corresponding to our final results reported in Table 5 in Table 12

	SNOMED CT Core				**CN-82K**				**FB15K-237**				**WN18RR**			
	MRR	H@1	H@3	H@10	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	.582	.511	.610	.729
BERT-ResNet + Prompt-tuning + Normalizing Flow	.517	.411	.574	.719	.202	.137	.223	.329	.361	.278	.394	.530	.591	.521	.618	.736
BERT-ResNet + Residual MLP	.551	.445	.608	.754	.213	.142	.235	.356	.378	.286	.414	.564	.592	.521	.621	.737
BERT-ResNet + Prompt-tuning + Residual MLP	.551	.444	.612	.757	.218	.146	.240	.363	.377	.287	.410	.564	.600	.531	.626	.742

Table 12: Validation results corresponding to results reported in Table 5.