Scientific Language Models for Biomedical Knowledge Base Completion: An Empirical Study

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Biomedical knowledge graphs (KGs) hold rich information on entities such as diseases, drugs, and genes. Predicting missing links in these graphs can boost many important applications, such as drug design and repurposing. Recent work has shown that general-domain language models (LMs) can serve as "soft" KGs, and that they can be fine-tuned for the task of KG completion. In this work, we study *scientific* LMs for KG completion, exploring whether we can tap into their latent knowledge to enhance biomedical link prediction. We evaluate several domain-specific LMs, fine-tuning them on datasets centered on drugs and diseases that we represent as KGs and enrich with textual entity descriptions. We integrate the LM-based models with KG embedding models, using a router method that learns to assign each input example to either type of model and provides a substantial boost in performance. Finally, we demonstrate the advantage of LM models in the inductive setting with novel scientific entities. Our datasets and code are made publicly available.[1]

1 Introduction

Understanding complex diseases such as cancer, HIV, and COVID-19 requires rich biological, chemical, and medical knowledge. This knowledge plays a vital role in the process of discovering therapies for these diseases — for example, identifying targets for drugs [20] requires knowing what genes or proteins are involved in a disease, and designing drugs requires predicting whether a drug molecule will interact with specific target proteins. In addition, to alleviate the great costs of designing new drugs, drug repositioning [22] involves identification of *existing* drugs that can be re-purposed for other diseases. Due to the challenging combinatorial nature of these tasks, there is need for automation with machine learning techniques. Given the many links between biomedical entities, recent work [6, 7] has highlighted the potential benefits of *knowledge graph* (KG) data representations, formulating the associated tasks as *KG completion* problems — predicting missing links between drugs and diseases, diseases and genes, and so forth.

The focus of KG completion work — in the general domain, as well as in biomedical applications — is on using graph structure to make predictions, such as with KG embedding (KGE) models and graph neural networks [40, 10]. In parallel, recent work in the general domain has explored the use of pretrained language models (LMs) as "soft" knowledge bases, holding factual knowledge latently encoded in their parameters [25, 26]. An emerging direction for using this information for the task of KG completion involves fine-tuning LMs to predict relations between pairs of entities based on their textual descriptions [39, 17, 35, 12]. In the scientific domain, this raises the prospect of using LMs trained on millions of research papers to tap into the scientific knowledge that may be embedded in

[1] Redacted for anonymity.

Submitted to NeurIPS 2021 AI for Science Workshop.

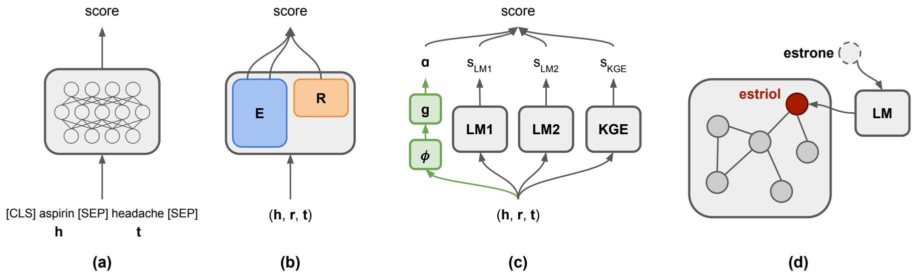

Figure 1: Our main methods for biomedical KG completion: (a) LM fine-tuning; (b) KGE models; (c) an approach that combines both; and (d) using an LM to impute missing entities in a KGE model.

their parameters. While this text-based approach has been evaluated on general domain benchmarks derived from WordNet [23] and Freebase [5], to our knowledge it has not been applied to the task of scientific KG completion.

Our contributions. We perform an extensive study of LM-based KG completion in the biomedical domain, focusing on three datasets centered on drugs and diseases, two of which have not been used to date for the KG completion task. To enable exploration of LM-based models, we collect missing entity descriptions, obtaining them for over 35k entities across all datasets. We evaluate a range of KGE models and *domain-specific* scientific LMs pretrained on different biomedical corpora [3, 19, 1, 15]. We conduct analyses of predictions made by both types of models and find them to have complementary strengths, echoing similar observations made in recent work in the general domain [35] and motivating integration of both text and graph modalities. Unlike previous work, we train a router that selects for each input instance which type of model is likely to do better, finding it to often outperform average-based ensembles. Integration of text and graph modalities provides substantial relative improvements of 13–36% in mean reciprocal rank (MRR), and routing across multiple LM-based models further boosts results. Finally, we demonstrate the utility of LM-based models when applied to entities unseen during training, an important scenario in the rapidly evolving scientific domain. Our hope is that this work will encourage further research into using scientific LMs for biomedical KG completion, tapping into knowledge embedded in these models and making relational inferences between complex scientific concepts.

2 Task and Methods

We begin by presenting the KG completion task and the approaches we employ for predicting missing links in biomedical KGs. An overview of our approaches is illustrated in Figure 1.

2.1 KG Completion Task

Formally, a KG consists of entities $\mathcal{E}$, relations $\mathcal{R}$, and triples $\mathcal{T}$ representing *facts*. Each triple $(h, r, t) \in \mathcal{T}$ consists of head and tail entities $h, t \in \mathcal{E}$ and a relation $r \in \mathcal{R}$. An entity can be one of many types, with the type of an entity e denoted as $T(e)$. In our setting, each entity is also associated with some text, denoted as $\text{text}(e)$ for $e \in \mathcal{E}$. The task of *KG completion* or *link prediction* involves receiving a triple $(h, r, ?)$ (where $?$ can replace either the head or tail entity) and scoring all candidate triples $\{(h, r, t') \mid t' \in \mathcal{S}\}$ such that the correct entity that replaces $?$ has the highest score. Each KG completion model in our experiments learns a function f that computes a ranking score $s = f(x)$ for a given triple $x = (h, r, t)$. Models are trained to assign a high ranking score to correct positive triples from the set of known facts $\mathcal{T}$ and a low ranking score to triples that are likely to be incorrect. To do so, we use the max-margin loss function. We also explore the inductive setting, with nodes not seen during training time (see Appendix B.2).

2.2 Methods

KG embedding (KGE) models. For each entity $e \in \mathcal{E}$ and each relation $r \in \mathcal{R}$, KG embedding (KGE) models learn a vector representation $E(e) \in \mathbb{R}^m$ and $R(r) \in \mathbb{R}^n$. For a given triple (h, r, t), each model computes the ranking score $f(h, r, t)$ as a simple function of these embeddings. We include a variety of different KGE models in our experiments, including TransE [8], DistMult [38], ComplEx [33], and RotatE [30].

LM-based models. KGE methods do not capture the rich information available from textual descriptions of nodes. To address this limitation, previous KG completion approaches have incorporated textual representations [32, 36], most recently with approaches such as KG-BERT [39] that fine-tune the BERT language model (LM) [13] for the task of KG completion. Our focus in this work is on LMs pretrained on corpora of biomedical documents (e.g., PubMedBERT [15]; see Appendix C.1.2 for full details). To score a triple using an LM, we use a cross-encoder approach [17] (Fig. 1a), where we encode the text of the head and tail entities together as $v = \text{LM}(\texttt{[CLS]}\ \text{text}(h)\ \texttt{[SEP]}\ \text{text}(t)\ \texttt{[SEP]})$, where v is the contextualized representation of the `[CLS]` token at the last layer. We use the approach of Kim et al. [17] and incorporate two additional losses for each LM: a binary triple classification loss to identify if a triple is positive or negative, and a multi-class relation classification loss.[2]

2.3 Integrating KGE and LM: Model Averaging vs. Routing

We study integration of graph-based and text-based methods (Figure 1c), exploring whether learning to route input instances adaptively to a *single* model can improve performance over previous approaches that compute a weighted average of ranking scores [35]. We also explore the more general setup of combining more than two models. More formally, for a given triple $x = (h, r, t)$, let $\phi(x)$ be its feature vector. We can learn a function $g(\phi(x))$ that outputs a set of weights $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k], \sum_i \alpha_i = 1, \alpha_i > 0\ \forall i$. These weights can be used to perform a weighted average of the ranking scores $\{s_1, \ldots, s_k\}$ for a set of k models we wish to combine, such that the final ranking score is $s = \sum_i \alpha_i s_i$. We use a variety of graph-, triple-, and text-based features to construct the feature vector $\phi(x)$ (full list in Appendix C.1.3 , Table 6). For the function $g(\cdot)$, we experiment with an **input-dependent weighted average** that outputs arbitrary weights $\boldsymbol{\alpha}$ and a **router** that outputs a constrained $\boldsymbol{\alpha}$ such that $\alpha_i = 1$ for some i and $\alpha_j = 0, \forall j \neq i$ (i.e., $\boldsymbol{\alpha}$ is a one-hot vector). In practice, we implement the router as a classifier which selects a single KG completion model for each example by training it to predict which model will perform better. For the input-dependent weighted average we train a multilayer perceptron (MLP) using the max-margin ranking loss.

3 Experiments and Results

3.1 Datasets

We use three datasets in the biomedical domain that cover a range of sizes comparable to existing general domain benchmarks, each pooled from a broad range of biomedical sources. Our datasets include **RepoDB** [9], a collection of drug-disease pairs intended for drug repositioning research; **MSI** (multiscale interactome; [28]), a recent network of diseases, proteins, genes, drug targets, and biological functions; and **Hetionet** [16], a heterogeneous biomedical knowledge graph which following Alshahrani et al. [2] we restrict to interactions involving drugs, diseases, symptoms, genes, and side effects.[3] In order to apply LMs to each dataset, we scrape entity names (when not provided by the original dataset) as well as descriptions from the original online sources used to construct each KG (see Table 3 in the appendix). While Hetionet has previously been explored for the task of KG completion as link prediction (though not LMs) [2, 7], to our knowledge neither RepoDB nor MSI have been represented as KGs and used for evaluating KG completion models.

3.2 Link Prediction Results

We report performance in Table 1. For each LM that has been fine-tuned for KG completion, we add the prefix "KG-" (e.g., KG-PubMedBERT). While LMs perform competitively with KGE models

[2] See details in Appendix C.

[3] More information on each dataset is available in Appendix A.1.

		RepoDB			Hetionet			MSI		
		MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
KGE	ComplEx	62.3	71.1	85.6	45.9	53.6	77.8	40.3	44.3	57.5
	DistMult	62.0	70.4	85.2	46.0	53.5	77.8	29.6	34.1	53.6
	RotatE	58.8	65.9	79.8	50.6	58.2	79.3	32.4	35.3	49.8
	TransE	60.0	68.6	81.1	50.2	58.0	79.8	32.7	36.5	53.8
LM (fine-tuned)	RoBERTa	51.7	60.3	82.3	46.4	53.6	76.9	30.1	33.3	50.6
	SciBERT	59.7	67.6	88.5	50.3	57.1	79.1	34.2	37.9	55.0
	BioBERT	58.2	65.8	86.8	50.3	57.5	79.4	33.4	37.1	54.8
	Bio+ClinicalBERT	55.7	64.0	84.1	43.6	49.1	72.6	32.6	36.1	53.5
	PubMedBERT-abs	60.8	70.7	89.5	50.8	58.0	80.0	34.3	38.0	55.3
	PubMedBERT-full	59.9	69.3	88.8	51.7	58.7	80.8	34.2	37.7	55.1
Two models (router)	Best pair of KGE	62.2	70.4	83.7	56.1	65.5	85.4	45.2	50.6	66.2
	Best KGE + LM	70.6	80.3	94.3	59.7	68.6	87.2	48.5	54.4	70.1
Two models (input-dep. avg.)	Best pair of KGE	65.2	74.3	87.6	65.3	75.3	90.2	39.8	44.9	62.0
	Best KGE + LM	65.9	74.4	91.5	**70.3**	**78.7**	**92.2**	40.6	44.6	61.2
Three models (router)	2 KGE + 1 LM	**72.7**	81.6	95.2	62.6	71.7	89.4	50.9	**57.1**	**73.2**
	1 KGE + 2 LM	72.1	**82.5**	**95.7**	62.1	71.9	89.5	**51.2**	57.0	73.0

Table 1: KG completion results. Underlined values denote the best result within a model category, while bold values denote the best result for each dataset.

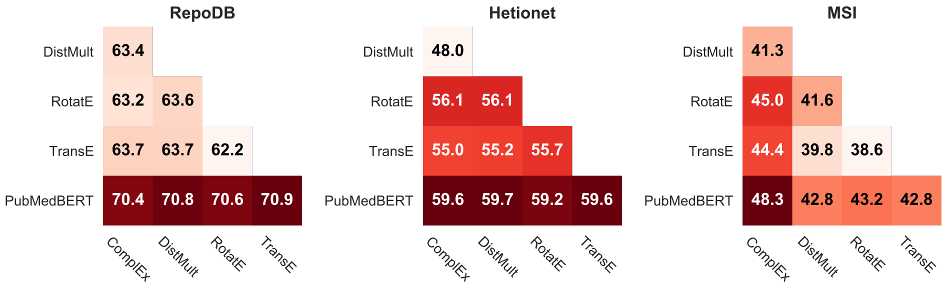

Figure 2: Test set MRR for all pairs of models, using an MLP router. The best combination of a KGE model and KG-PubMedBERT always outperforms the best pair of KGE models.

and even outperform some, they generally do not match the best KGE model on RepoDB and MSI. This echoes results in the general domain for link prediction on subsets of WordNet and Freebase [39, 35]. Combining each class of models boosts results by a large relative improvement of 13–36% in MRR across datasets. Moreover, the best-performing combination always includes a KGE model and KG-PubMedBERT rather than two KGE models (Fig. 2 in the appendix), showing the unique benefit of using LMs to augment models relying on KG structure alone.

Routing vs. Averaging We also compare the router and input-dependent weighted average approaches of integrating a pair of models, with the router-based approach outperforming on RepoDB and MSI. This presents routing as a promising alternative for integrating KGE and LM models. The three-model combinations provide the best performance for RepoDB and MSI.

4 Conclusion and Discussion

We perform the first empirical study of scientific language models (LMs) applied to biomedical knowledge graph (KG) completion. We evaluate *domain-specific* biomedical LMs, fine-tuning them to predict missing links in KGs that we construct by enriching biomedical datasets with textual entity descriptions. We find that LMs and more standard KG embedding models have complementary strengths, and propose a routing approach that integrates the two by assigning each input example to either type of model to boost performance. We also demonstrate the utility of LMs in the inductive setting with entities not seen during training, an important scenario in the scientific domain. Our findings provide a promising direction for biomedical knowledge completion tasks, and for literature-based scientific discovery [31, 14].

References

[1] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly Available Clinical BERT Embeddings. In *2nd Clinical Natural Language Processing Workshop*, 2019.

[2] Mona Alshahrani, Maha A. Thafar, and Magbubah Essack. Application and evaluation of knowledge graph embeddings in biomedical data. *PeerJ Computer Science*, 7, 2021.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A Pretrained Language Model for Scientific Text. In *EMNLP*, 2019.

[4] Rajarshi Bhowmik and Gerard de Melo. Explainable Link Prediction for Emerging Entities in Knowledge Graphs. In *SEMWEB*, 2020.

[5] Kurt Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A Collaboratively Created Graph Database For Structuring Human Knowledge. In *SIGMOD Conference*, 2008.

[6] Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, Andreas Bender, Charles Tapley Hoyt, and William Hamilton. A Review of Biomedical Datasets Relating to Drug Discovery: A Knowledge Graph Perspective. *arXiv:2102.10062*, 2021.

[7] Stephen Bonner, Ian P Barrett, Cheng Ye, Rowan Swiers, Ola Engkvist, and William L Hamilton. Understanding the Performance of Knowledge Graph Embeddings in Drug Discovery. *arXiv:2105.10488*, 2021.

[8] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *NIPS*, 2013.

[9] Adam S. Brown and Chirag J. Patel. A standard database for drug repositioning. *Scientific Data*, 4, 2017.

[10] David Chang, Ivana Balažević, Carl Allen, Daniel Chawla, Cynthia Brandt, and Andrew Taylor. Benchmark and Best Practices for Biomedical Knowledge Graph Embeddings. In *19th SIGBioMed Workshop on Biomedical Language Processing*, pages 167–176, 2020.

[11] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. *KDD*, 2016.

[12] Daniel Daza, Michael Cochez, and Paul T. Groth. Inductive Entity Representations from Text via Link Prediction. In *WWW*, 2021.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

[14] Vishrawas Gopalakrishnan, Kishlay Jha, Wei Jin, and Aidong Zhang. A survey on literature based discovery approaches in biomedical domain. *Journal of biomedical informatics*, 93:103141, 2019.

[15] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing. *arXiv:2007.15779*, 2020.

[16] Daniel S. Himmelstein and Sergio E. Baranzini. Heterogeneous Network Edge Prediction: A Data Integration Approach to Prioritize Disease-Associated Genes. *PLoS Computational Biology*, 11, 2015.

[17] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models. In *COLING*, 2020.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

[19] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36:1234 – 1240, 2020.

[20] Mark A Lindsay. Target discovery. *Nature Reviews Drug Discovery*, 2(10):831–838, 2003.

[21] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692*, 2019.

[22] Huimin Luo, Min Li, Mengyun Yang, Fang-Xiang Wu, Yaohang Li, and Jianxin Wang. Biomedical data and computational models for drug repositioning: a comprehensive review. *Briefings in bioinformatics*, 22(2):1604–1619, 2021.

[23] George A. Miller. WordNet: a lexical database for English. *Commun. ACM*, 38:39–41, 1995.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.

[25] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastien Riedel. Language Models as Knowledge Bases? In *EMNLP*, 2019.

[26] Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. How Context Affects Language Models' Factual Predictions. In *AKBC*, 2020.

[27] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*, 2019.

[28] Camilo Ruiz, Marinka Zitnik, and Jure Leskovec. Identification of disease treatment mechanisms through the multiscale interactome. *Nature communications*, 2021.

[29] Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *ESWC*, 2018.

[30] Zhiqing Sun, Zhihong Deng, Jian-Yun Nie, and Jian Tang. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*, 2019.

[31] Don R. Swanson. Fish oil, Raynaud's syndrome, and undiscovered public knowledge. *Perspectives in biology and medicine*, 30(1):7–18, 1986.

[32] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing Text for Joint Embedding of Text and Knowledge Bases. In *EMNLP*, 2015.

[33] Théo Trouillon, Johannes Welbl, S. Riedel, Éric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *ICML*, 2016.

[34] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR*, 2020.

[35] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *WWW*, 2021.

[36] Zhigang Wang and Juan-Zi Li. Text-Enhanced Representation Learning for Knowledge Graph. In *IJCAI*, 2016.

[37] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation Learning of Knowledge Graphs with Entity Descriptions. In *AAAI*, 2016.

[38] Bishan Yang, Wen tau Yih, Xiadong He, Jianfeng Gao, and Li Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*, 2015.

[39] Liang Yao, Chengsheng Mao, and Yuan Luo. KG-BERT: BERT for Knowledge Graph Completion. *arXiv:1909.03193*, 2019.

[40] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):i457–i466, 2018.

Dataset	#Entities	#Rel	#Positive Edges			Avg. Desc. Length
			Train	Dev.	Test	
RepoDB	2,748	1	5,342	667	668	49.54
Hetionet (our subset)	12,733	4	124,544	15,567	15,568	44.65
MSI	29,959	6	387,724	48,465	48,465	45.13
WN18RR	40,943	11	86,835	3,034	3,134	14.26
FB15k-237	14,541	237	272,115	17,535	20,466	139.32

Table 2: Statistics for our datasets and a sample of general domain benchmarks.

Dataset	Link	Sources
RepoDB	http://apps.chiragjpgroup.org/repoDB/	DrugBank UMLS
Hetionet	https://github.com/hetio/hetionet	DrugBank Disease Ontology Entrez SIDER MeSH
MSI	https://github.com/snap-stanford/multiscale-interactome	DrugBank Gene Ontology Entrez UMLS

Table 3: Links and sources of entity names and descriptions for each dataset.

A Dataset Construction

A.1 Sources

While Hetionet has previously been explored for the task of KG completion as link prediction using KGE models (though not LMs) [2, 7], to our knowledge neither RepoDB nor MSI have been represented as KGs and used for evaluating KG completion models despite the potential benefits of this representation [6], especially in conjunction with textual information.

RepoDB Drugs in RepoDB have statuses including *approved*, *terminated*, *withdrawn*, and *suspended*. We restrict our KG to pairs in the *approved* category.

Hetionet was constructed using data from various publicly-available scientific repositories. Following Alshahrani et al. [2], we restrict the KG to the *treats*, *presents*, *associates*, and *causes* relation types. This includes interactions between drugs and the diseases they treat, diseases and their symptoms, diseases and associated genes, and drugs and their side effects. We use this subset of the full Hetionet dataset to avoid scalability issues that arise when training large Transformer-based language models, inspired by benchmark datasets such as FB15K [8], a subset of the Freebase knowledge base.

MSI includes diseases and the proteins they perturb, drug targets, and biological functions designed to discover drug-disease treatment pairs through the pathways that connect them via genes, proteins, and their functions. We include all entities and relation types in the dataset.

We collect each of the datasets from the links listed in Table 3. For missing entity names and all descriptions, we write scripts to scrape the information from the resources listed above using the entity identifiers provided by each of the datasets.

A.2 Transductive Splits

We construct an 80%/10%/10% training/development/test transductive split for each KG by removing edges from the complete graph while ensuring that all nodes remain in the training graph. We also construct inductive splits, where each positive triple in the test test has one or both entities unseen during training.

To construct transductive splits for each dataset, we begin with the complete graph, and repeat the following steps:

1. Randomly sample an edge from the graph.
2. If the degree of both nodes incident to the edge is greater than one, remove the edge.
3. Otherwise, replace the edge and continue.

The above steps are repeated until validation and test graphs have been constructed of the desired size while ensuring that no entities are removed from the training graph. We construct 80%/10%/10% training/validation/test splits of all datasets.

A.3 Inductive Splits

To construct inductive splits for each dataset, we follow the procedure outlined in the "Technical Details" section of the appendix of Daza et al. [12]. We similarly construct a 80%/10%/10% training/validation/test split of each dataset in the inductive setting.

A.4 Negative Validation/Test Triples

A.5 Evaluation

At test time, each positive triple is ranked against a set of negatives constructed by replacing either the head or tail entity by a fixed set of entities of the same type. When constructing the edge split for each of the three datasets, we generate a fixed set of negatives for every positive triple in the validation and test sets, each corresponding to replacing the head or tail entity with an entity of the same type and filtering out negatives that appear as positive triples in either the training, validation, or test set. For each positive triple, we use its rank to compute the mean reciprocal rank (MRR), Hits@3 (H@3), and Hits@10 (H@10) metrics.

In order to perform a ranking-based evaluation for each dataset in both the transductive and inductive settings, we generate a set of negative triples to be ranked against each positive triple. To generate negative entities to replace both the head and tail entity of each validation and test positive, we follow the procedure below:

1. Begin with the set of all entities in the knowledge graph.
2. Remove all entities that do not have the same entity type as the entity to be ranked against in the positive triple.
3. Remove all entities that would result in a valid positive triple in either the training, validation, or test sets.
4. Randomly sample a fixed set of size m from the remaining set of entities.

We use a value of $m = 500$ for RepoDB and MSI, and a value of $m = 80$ for Hetionet (due to the constraints above, the minimum number of valid entities remaining across positive triples for Hetionet was 80). Using a fixed set of entities allows for fair comparison when assessing performance of subsets of the test set, such as when examining the effect of subsets where descriptions are present for neither, one, or both entities (Table 7).

B Results

Since the gradient boosted decision trees (GBDT) router achieves the best validation set performance in most cases across classifiers and integration methods, we use this method for combinations of more than two models, such as multiple LMs with a single KGE model.

Relation	RotatE better	KG-PubMedBERT better
Disease *presents* Symptom	Disease: **mediastinal cancer**; a cancer in the mediastinum. Symptom: **hoarseness**; a deep or rough quality of voice.	Disease: **stomach cancer**; a gastrointestinal cancer in the stomach. Symptom: **weight loss**; decrease in existing body weight.
Compound *treats* Disease	Compound: **methylprednisolone**; a prednisolone derivative glucocorticoid with higher potency. Disease: **allergic rhinitis**; a rhinitis that is an allergic inflammation and irritation of the nasal airways.	Compound: **altretamine**; an alkylating agent proposed as an antineoplastic. Disease: **ovarian cancer**; a female reproductive organ cancer that is located in the ovary.
Compound *causes* Side Effect	Compound: **cefaclor**; semi-synthetic, broad-spectrum anti-biotic derivative of cephalexin. Side Effect: **tubulointerstitial nephritis**; *no description*	Compound: **perflutren**; a diagnostic medication to improve contrast in echocardiograms. Side Effect: **palpitations**; irregular and/or forceful beating of the heart.

Table 4: Examples from Hetionet where one model ranks the shown positive pair considerably higher than the other. LMs often perform better when there is semantic relatedness between head and tail text, but can be outperformed by a KGE model when head/tail entity text is missing or unrelated. Entity descriptions cut to fit.

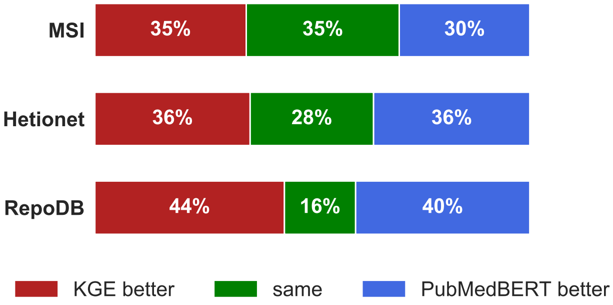

Figure 3: Fraction of test set examples where each model performs better.

Interpreting model routing. We compute average feature gain for all datasets, using a GBDT router implemented with XGBoost [11] (see Fig. 5 in the appendix). We find that the most salient features are the ranking scores output by each model, which is intuitive as these scores reflect each model's confidence. Graph features like node degree and PageRank also factor into the classifier's predictions, as well as textual features such as entity text length and edit distance between entity names. General concepts such as *Hypertensive disease* and *Infection of skin and/or subcutaneous tissue* are central nodes for which we observe KGE models to often do better. KGE models also tend to do better on entities with short, non-descriptive names (e.g., *P2RY14*), especially when no descriptions are available. Generally, these patterns are not clear-cut, and non-linear or interaction effects likely exist. It remains an interesting challenge to gain deeper understanding into the strengths and weaknesses of LM-based and graph-based models.

B.1 Comparing model errors.

By examining a selected set of examples in Table 4, we can observe cases where information in text provides LMs an advantage and where a lack of context favors KGE models.

KG-PubMedBERT is able to make connections between biomedical concepts – like the fact that a disease that affects the *stomach* might cause *weight loss* – and align related concepts expressed with different terminology – like connecting *antineoplastic* with *cancer* (a type of *neoplasm*), or recognizing that an *echocardiogram* is a technique for imaging the *heart*. In contrast, RotatE offers an advantage when the descriptions do not immediately connect the two terms (*mediastinal cancer*, *hoarseness*), where a description may be too technical or generic to be informative (*methylpred-*

	RepoDB			Hetionet			MSI		
	MRR	H@3	H@10	MRR	H@3	H@10	MRR	H@3	H@10
DKRL	15.6	15.9	28.2	17.8	18.5	31.9	13.3	14.1	22.4
KG-PubMedBERT	**38.8**	**43.4**	**67.5**	**21.6**	**22.3**	**42.8**	**20.2**	**21.7**	**32.2**
ComplEx	0.8	0.4	1.6	3.6	0.7	2.8	0.5	0.1	0.4
NN-ComplEx, frozen LM	20.1	22.3	31.2	18.1	18.4	32.8	15.8	16.9	23.4
NN-ComplEx, fine-tuned	26.9	30.3	39.4	13.9	12.9	25.5	14.6	15.4	21.4

Table 5: Inductive KG completion results. NN-ComplEx refers to the version of ComplEx with unseen entity embeddings replaced using an LM to find the 1-nearest neighbor, either with PubMedBERT frozen or fine-tuned for KG completion (KG-PubMedBERT).

nisolone, *allergic rhinitis*), or where no description is available (*cefaclor*, *tubulointerstitial nephritis*).[4] Furthermore, Fig. 3 shows that KG-PubMedBERT outperforms the best KGE model on a substantial fraction of the test set examples for each dataset.[5] These observations motivate an approach that leverages the strengths of both types of models by identifying examples where each model might do better, which leads to our results for model integration.

B.2 Inductive KG Completion

KGE models are limited to the *transductive* setting where all entities seen during evaluation have appeared during training. *Inductive* KG completion is important in the biomedical domain, where we may want to make predictions on novel entities such as emerging biomedical concepts or drugs/proteins mentioned in the literature that are missing from existing KGs. Due to their ability to form compositional representations from entity text, LMs are well-suited to this setting. In addition to using LMs fine-tuned for KGC, we try a simple technique using LMs to“fill in” missing KGE embeddings without explicitly using the LM for prediction (Fig. 1d).

For our inductive KG completion experiments, we use ComplEx as the KGE model and KG-PubMedBERT as our LM-based model, and compare the performance of each method to ComplEx with entity embeddings imputed using the method described in Section B.2. We use either the untrained PubMedBERT or the fine-tuned KG-PubMedBERT as the LM for retrieving nearest-neighbor (NN) entities (see examples in Table 9 in the appendix). We also compare to DKRL [37], which constructs entity representations from text using a CNN encoder and uses the TransE scoring function. We use PubMedBERT's token embeddings as input to DKRL and train with the same multi-task loss. While other methods for inductive KG completion exist, such as those based on graph neural networks [29, 34, 4], they require the unseen entity to have *known connections* to entities that were seen during training in order to propagate information needed to construct the new embedding. In our inductive experiments, we consider the more challenging setup where every test set triple has at least one entity with no known connections to entities seen during training, such that graph neural network-based methods cannot be applied. This models the phenomenon of rapidly emerging concepts in the biomedical domain, where a novel drug or protein may be newly studied and discussed in the scientific literature without having been integrated into existing knowledge bases.

As seen in Table 5, ComplEx unsurprisingly performs poorly as it attempts link prediction with random embeddings for unseen entities. DKRL does substantially better, with KG-PubMedBERT further increasing MRR with a relative improvement of 21% (Hetionet) to over 2x (RepoDB). Our strategy for replacing ComplEx embeddings for unseen entities performs comparably to or better than DKRL in most cases, with untrained PubMedBERT encodings generally superior to using KG-PubMedBERT's encodings. In either case, this simple strategy for replacing the untrained entity embeddings of a KGE model shows the ability of an LM to augment a structure-based method for KG completion that is typically only used in the transductive setting, even without using the LM to compute ranking scores.

[4] Table 7 in the appendix shows the drop in performance when one or both entities are missing descriptions.

[5] See MRR breakdown by relation type in Fig. 4 in the appendix.

C Training

C.1 Transductive Setting

For all individual models, we train the models on the training set of each dataset while periodically evaluating on the validation set. We save the model with the best validation set MRR, then use that model to evaluate on the test set. We also perform hyperparameter tuning for all models, and use validation set MRR to select the final set of hyperparameters for each model.

C.1.1 Knowledge Graph Embeddings

We use the max-margin ranking loss for all KGE methods. We use a batch size of 512 for all models. We train models for 10,000 steps (958 epochs) on RepoDB, 50,000 steps (205 epochs) on Hetionet, and 50,000 steps (66 epochs) on MSI. We evaluate on the validation set every 500 steps for RepoDB and 5,000 steps for Hetionet and MSI. We use the Adam optimizer for training. We perform a hyperparameter search over the following values:

- Embedding dimension: 500, 1000, 2000
- Margin for max-margin loss: 0.1, 1
- Learning rate: 1e-3, 1e-4
- Number of negative samples per positive: 128, 256
- Parameter for L3 regularization of embeddings: 1e-5, 1e-6

C.1.2 Language Models

Pretrained scientific LMs. We explore various pretrained LMs, with their initialization, vocabulary, and pretraining corpora described below. In particular, we study a range of LMs trained on different scientific and biomedical literature, and also on clinical notes.

- **BioBERT** [19] Initialized from BERT and using the same general domain vocabulary, with additional pretraining on the PubMed repository of scientific abstracts and full-text articles.
- **Bio+ClinicalBERT** [1] Initialized from BioBERT with additional pretraining on the MIMIC-III corpus of clinical notes.
- **SciBERT** [3] Pretrained from scratch with a domain-specific vocabulary on a sample of the Semantic Scholar corpus, of which biomedical papers are a significant fraction but also papers from other scientific domains.
- **PubMedBERT** [15] Pretrained from scratch with a domain-specific vocabulary on PubMed. We apply two versions of PubMedBERT, one trained on PubMed abstracts alone (PubMedBERT-abstract) and the other on abstracts as well as full-text articles (PubMedBERT-fulltext).

We also use **RoBERTa** [21] – pretrained from scratch on the BookCorpus, English Wikipedia, CC-News, OpenWebText, and Stories datasets – as a strongly-performing general domain model for comparison. For all LMs, we follow Kim et al. [17] and use the multi-task loss consisting of binary triple classification, multi-class relation classification, and max-margin ranking loss, with a margin of 1 for the max-margin loss. For triple classification, given the correct label $y \in \{0, 1\}$ (positive or negative triple) we apply a linear layer to the `[CLS]` token representation v to output the probability p of the triple being correct as $p = \sigma(W_{\text{triple}} v)$, and use the binary cross entropy loss $\mathcal{L}_{\text{triple}}(x) = -y \log(p) - (1 - y) \log(1 - p)$. For relation classification over R relation types, we apply a linear layer to v to calculate a probability distribution q over relation classes with $q = \text{softmax}(W_{\text{rel}} v)$, and use the cross entropy loss with one-hot vector $y \in \{0, 1\}^R$ as the correct relation label: $\mathcal{L}_{\text{rel}}(x) = -\sum_{i=1}^{R} y_i \log q_i$. The final loss is the equally-weighted sum of all three losses: $\mathcal{L}(x) = \mathcal{L}_{\text{rank}}(x) + \mathcal{L}_{\text{triple}}(x) + \mathcal{L}_{\text{rel}}(x)$.

We train for 40 epochs on RepoDB, and 10 epochs on Hetionet and MSI. We evaluate on the validation set every epoch for RepoDB, and three times per epoch for Hetionet and MSI. For RepoDB, Hetionet, and MSI we use 32, 16, and 8 negative samples per positive, respectively. We use the Adam optimizer for training. We perform a hyperparameter search over the following values:

- Batch size: 16, 32
- Learning rate: 1e-5, 3e-5, 5e-5

C.1.3 Integrated Models

Global weighted average. For the global weighted average, we compute ranking scores for positive and negative examples as the weighted average of ranking scores output by all KG completion models being integrated. Specifically, for a set of ranking scores $s_1, \ldots, s_k$ output by k models for an example, we learn a set of weights $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_k]$ to compute the final ranking score as $s = \sum_{i=1}^{k} \alpha_i s_i$, where the same weight vector $\boldsymbol{\alpha}$ is used for all examples. We search for each α_i over the grid [0.05, 0.95] with steps of 0.05, ensuring that all α_i's sum to 1. We choose values that maximize validation set MRR, then apply them to the test set.

Router. For the router-based method, we train a classifier to select a single model out of a set of KG completion models to use for computing ranking scores for a positive example and its associated negatives. The class to be predicted for a particular example corresponds to which model performs best on that example (i.e., gives the best rank), with an additional class for examples where all models perform the same. We explore a number of different classifiers, including logistic regression, decision tree, gradient boosted decision tree (GBDT), and multilayer perceptron (MLP), finding that GBDT and MLP classifiers perform the best. As input to the classifier, we use a diverse set of features computed from each positive example (listed in Table 6) as well as each model's ranking score for the positive example. Classifiers are trained on the validation set and evaluated on the test set for each dataset. We additionally perform hyperparameter tuning over the following values for each classifier:

Logistic regression:

- Penalty: L1, L2
- Regularization parameter: 9 values evenly log-spaced between 1e-5 and 1e3

Decision tree:

- Max depth: 2, 4, 8
- Learning rate: 1e-1, 1e-2, 1e-3

GBDT:

- Number of boosting rounds: 100, 500, 1000
- Max depth: 2, 4, 8
- Learning rate: 1e-1, 1e-2, 1e-3

MLP:

- Number of hidden layers: 1, 2
- Hidden layer size: 128, 256
- Batch size: 64, 128, 256
- Learning rate: 1e-1, 1e-2, 1e-3

We perform five-fold cross-validation on the validation set and use validation set accuracy to choose the best set of hyperparameters for each classifier. We use Scikit-Learn [24] to implement the logistic regression and MLP classifiers, and XGBoost [11] to implement the decision tree and GBDT classifiers, using default parameters other than the ones listed above.

Input-dependent weighted average. The input-dependent weighted average method of integrating KG completion models operates similarly to the global weighted average, except that the set of weights can vary for each positive example and are a function of its feature vector (the same set of weights is used for all negative examples used to rank against each positive example). We train an MLP to output a set of weights that are then used to compute a weighted average of ranking scores for a set of KG completion models. The MLP is trained on the validation set and evaluated on the test set for each dataset. We use the max-margin ranking loss with a margin of 1. In order to compare to the MLP trained as a router, we train the MLP using the Adam optimizer [18] for 200 epochs with early stopping on the training loss and a patience of 10 epochs (the default settings for an MLP classifier in Scikit-Learn). We perform a hyperparameter search over the following values (matching the values for the MLP router where applicable):

- Number of hidden layers: 1, 2
- Hidden layer size: 128, 256
- Batch size: 64, 128, 256
- Learning rate: 1e-1, 1e-2, 1e-4
- Number of negatives (for max-margin loss): 16, 32

We select the best hyperparameters by MRR on a held-out portion of the validation set.

Features for integrated models. Both the router and input-dependent weighted average methods of model integration use a function to outputs weights based on a feature vector of an example. A complete list of the features used by each method can be found in Table 6. We also use the ranking score for the positive example from each KG completion model being integrated as additional features.

entity type
relation type
head/tail node in-/out-degree
head/tail node PageRank
Adamic-Adar index of edge
edit dist. between head/tail entity names
length of text in chars.
presence of word “unknown” in name/desc.
missing desc.
number/ratio of punctuation/numeric chars.
tokens-to-words ratio of entity name/desc.

Table 6: Complete list of features used by router classifiers.

C.2 Inductive Setting

C.2.1 Inductive Nearest Neighbor Baseline

Given a set of entities $\mathcal{E}$ for which a KGE model has trained embeddings and a set of unknown entities $\mathcal{U}$, for each $e \in \mathcal{E} \cup \mathcal{U}$ we encode its text using an LM to form $v_e = \text{LM}(\texttt{[CLS]}\ \text{text}(e)\ \texttt{[SEP]}), \forall e \in \mathcal{E} \cup \mathcal{U}$, where v_e is the [CLS] token representation at the last layer. We use the cosine similarity between embeddings to replace each unseen entity's embedding with the closest trained embedding as $E(u) = E(\operatorname*{argmax}_{e \in \mathcal{E}} \text{cos-sim}(v_e, v_u))$ where e is of the same type as u, i.e., $T(e) = T(u)$.

C.2.2 Knowledge Graph Embeddings and Language Models

For the KGE and LM models, we follow the same training procedure for the inductive splits as for the transductive splits. We perform hyperparameter tuning over the same grids of hyperparameters, periodically evaluate on the validation set and save the checkpoint with the best validation set MRR, and use the set of hyperparameters corresponding to the highest validation set MRR to evaluate on the test set.

C.2.3 DKRL

In addition to the KGE and LM-based methods, we also train DKRL [37] for inductive KG completion as another text-based baseline for comparison. DKRL uses a two-layer CNN encoder applied to the word or subword embeddings of an entity's textual description to construct a fixed-length entity embedding. To score a triple, DKRL combines its entity embeddings constructed from text with a separately-learned relation embedding using the TransE [8] scoring function. The original DKRL model uses a joint scoring function with structure-based and description-based components; we restrict to the description-based component as we are applying DKRL in the inductive setting. We use PubMedBERT subword embeddings at the input layer of the CNN encoder, encode entity names

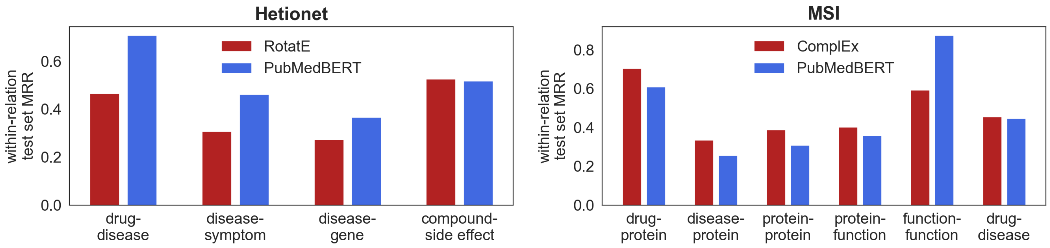

Figure 4: Test set MRR for the best KGE model compared to KG-PubMedBERT broken down by relation type for Hetionet and MSI.

and descriptions, and apply the same multi-task loss as for the LM-based models. To apply the triple classification and relation classification losses, for head and tail entity embeddings **h** and **t**, we apply a separate linear layer for each loss to the concatenated vector $[\mathbf{h}; \mathbf{t}; |\mathbf{h} - \mathbf{t}|]$, following previous work on models that use a bi-encoder to construct entity or sentence representations [35, 27]. We use the same number of training epochs and number of negatives per positive for DKRL as for the LM-based methods on each dataset. We use a batch size of 64, and perform a hyperparameter search over the following values:

- Learning rate: 1e-3, 1e-4, 1e-5
- Embedding dimension: 500, 1000, 2000
- Parameter for L2 regularization of embeddings: 0, 1e-3, 1e-2

D Additional Results

D.1 Transductive Setting, Individual Models

Missing entity descriptions. Table 7 shows test set MRR for KG-PubMedBERT on each dataset broken down by triples with either both, one, or neither entities having available descriptions. Across datasets, performance clearly degrades when fewer descriptions are available to provide context for the LM to generate a ranking score.

#entities with desc. in pair	MRR		
	RepoDB	**Hetionet**	**MSI**
None	N/A	25.6	25.1
One	59.5	43.6	25.4
Both	63.7	52.6	37.3

Table 7: Effect of descriptions on KG-PubMedBERT test set MRR.

Relation-level performance. Figure 4 shows test set MRR broken down by relation for the datasets with multiple relation types (Hetionet and MSI). KG-PubMedBERT performs better on all relation types except compound-side effect for Hetionet, and on the function-function relation for MSI.

D.2 Transductive Setting, Integrated Models

D.3 Inductive Setting

	RepoDB	**Hetionet**	**MSI**
Global avg.	70.4	55.8	42.1
Input-dep. avg.	65.9	**70.3**	40.6
Router	**70.6**	59.7	**48.5**

Table 8: Test set MRR for the best pair of a KGE model and KG-PubMedBERT for different methods of model integration.

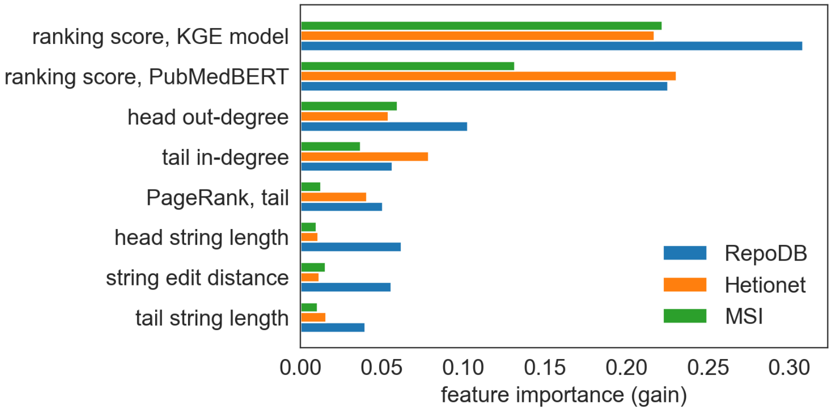

Figure 5: Feature importances for GBDT router for a selection of most important features. Ranking scores output by each model tend to be the most important, with other graph- and text-based features also contributing.

Imputation Model with Better Ranking	**Unseen Entity**	**KG-PubMedBERT nearest neighbors**	**PubMedBERT nearest neighbors**
PubMedBERT	eye redness	skin burning sensation, skin discomfort	conjunctivitis, throat sore
	ecchymosis	gas, thrombophlebitis	petechiae, macule
	estrone	vitamin a, methyltestosterone	estriol, calcitriol
KG-PubMedBERT	keratoconjunctivitis	conjunctivitis allergic, otitis externa	enteritis, parotitis
	malnutrition	dehydration, anaemia	meningism, wasting generalized
	congestive cardiomyopathy	diastolic dysfunction, cardiomyopathy	carcinoma breast, hypertrophic cardiomyopathy

Table 9: Samples of unseen entities and their nearest neighbors found by KG-PubMedBERT and PubMedBERT, for test set examples in the Hetionet inductive split where the PubMedBERT neighbor performs better than the KG-PubMedBERT neighbor (first three) and vice versa (last three). Each LM offers a larger improvement per example when its nearest neighbor is more semantically related to the unseen entity.