Cosine Similarity as Logits?: Few-shot Knowledge Graph Completion with Embedding Vectors of a Generative PLM and its Application in Knowledge Probing

Tomoyuki Jinno[1] Kazuki Hayashi[1] Yusuke Sakai[1] Hidetaka Kamigaito[1] Taro Watanabe[1]
[1]Nara Institute of Science and Technology
jinno.tomoyuki.jx3@naist.ac.jp
{hayashi.kazuki.hl4, sakai.yusuke.sr9, kamigaito.h, taro}@is.naist.jp

Abstract

The Knowledge graph completion (KGC) task aims to predict missing relations in knowledge graphs (KGs). Recently, text-based KGC approaches have gained attention but they present challenges: encoder-based methods require fine-tuning making it non-ideal when an ideal KG for training cannot be obtained, such as when KG is sparse or predicting new relation-types. Meanwhile, decoder-based methods make prediction by generating tokens, where entity disambiguation becomes a challenge. KGC is also used in knowledge proving, which aims to evaluate the knowledge retrieval capability of pre-trained language models (PLMs), but existing probes for generative PLM capable of ranking all multi-token and single-token entities are computationally inefficient. To address these problems, we propose DEER, an encoder-based few-shot KGC, leveraging a generative PLM that achieves a linear inference time complexity. Our experiment shows that DEER outperforms a fine-tuned KGC model in a relationally inductive setting and aligns with an existing knowledge-probing method, positioning it as a possible alternative.

1 Introduction

The knowledge graph completion (KGC) task aims to precit missing relations in existing knowledge graphs (KGs). A relation is represented as a triplet consisting of (*head-entity*, *relation-type*, *tail-entity*), hence the aim of KGC is to predict the *tail-entity* given a partially-filled triplet, (*head-entity*, *relation-type*, ?). Text-based KGC methods have recently gained popularity which can be categorized into encoder-based and decoder-based approaches.

Encoder-based approach, such as SimKGC [1], reformulates the KGC task as a document ranking task, treating the partially-filled triplet as a query and tail-entity as an answer. An encoder model is used to encode the query and tail-entity candidates as a vector which are then re-ranked by their similarity. In contrast, decoder-based approach treats the task as a text-generation task, generating name of tail-entities given a partially filled triplet using a generative pre-trained language model (PLM) [2, 3].

Encoder models often have fewer parameters than decoder models. This limits the parametric knowledge and inference performance of encoder-based approach [4, 5], necessitating fine-tuning on a KG, rather than a zero-shot or few-shot inference. This makes the approach non-ideal when a suitable KG for training is difficult to acquire, such as when the KG is sparse, evolves over time or when predicting new relation-types [6].

In contrast, predicting large number of tail-entity candidates using decoder-based approach is often inefficient. Hence, they typically predict only the most likely or top-n tail entities. In addition, linking the generated output to the correct entity, such as disambiguating identically named entities, presents a challenge [7].

KGC is also employed for knowledge probing, which aims to assess the factual knowledge retrieval capabilities of PLMs. LAMA [8] was the first of such method, but it fails to probe multi-token entities. Rank based knowledge probe for generative PLMs compatible with multi-token entities remain under explored [9] with BEAR [10] being the only method to our knowledge. However, it infers all combinations of queries and answers, making it computationally expensive, limiting KG size used for probing.

In this paper, we propose DEcoder Embedding-based Relational probe (DEER), a few-shot KGC model de-

Table 1 Comparison of key features with text-based KGC.

Encoder-Based	Decoder-Based	Ours
Fine-Tuned	**Few-Shot**	Few-Shot
Vector-Based	Token-Based	Vector-Based
Fully Ranked	Top-n	Fully Ranked
Sub-Billion Params	**Super-Billion**	Super-Billion

signed to overcome the limitations of both encoder-based and decoder-based methods, while retaining their advantages. DEER enables fully ranked, multi-entity-compatible knowledge probing and KGC under linear time complexity.

Table 1 compares the key characteristics of different KGC methods. As illustrated in Figure 1, our model leverages a generative language model, but differs from typical decoder-based methods by inferring using vector representations of entities, making it an encoder-based approach. These vectors are obtained from a generative PLM with the Prompt-Based Method with Explicit One Word Limitation (PromptEOL) [11], which acquires sentence embeddings from a generative PLM without additional training needs.

This work addresses the following questions:

1. How does DEER perform in relationally inductive settings?
2. Does it correlate with the prediction of LAMA?

Through answering these questions, we aim to demonstrate the generalization capability of our method in relationally sparse KGs and validate the use of DEER for knowledge probing by showing its agreement with LAMA.

2 Background

PromptEOL Embedding PromptEOL uses a generative PLM for sentence embedding. Given a sequence of input tokens $x_1, x_2, \ldots, x_n$, the last hidden state, $\mathbf{h}_n$, corresponding to the token x_n, is used as the sentence embedding vector. More specifically, $\mathbf{h}_n$ is the vector typically used for next token prediction by applying a final dense layer followed by a softmax function. This process is expressed as $\mathbf{z}_n = W\mathbf{h}_n + \mathbf{b}$, where $x_{n+1} = \arg\max(\text{softmax}(\mathbf{z}_n))$. To embed a sentence, it uses the following prompt template: *This sentence: "S" means in one word "*, where S is replaced with the target sentence.

Knowledge Probing LAMA was the first knowledge probe that evaluated PLMs on the KGC task. It first restricts tail-entity candidates to single-token entities and by prompting the model with cloze-style questions, ranks the

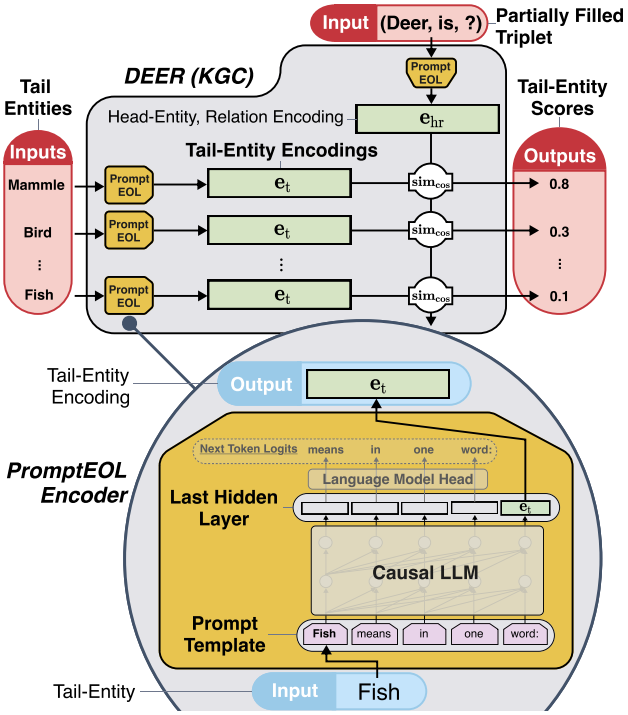

Figure 1 An illustration of the DEER architecture.

tail-entities by the log likelihood of the token corresponding to their name, at the masked position in the prompt. The mean precision at rank k (P@k), also known as Hit@k was used to score the models. Since only a single masked token is used, it cannot handle multi-token entities.

KAMEL [12] was introduced to address the problem, which probes multi-token entities by autoregressively generating the tail-entity names. It uses exact string match on the output for evaluation, which limits evaluation metrics to non-rank based scores (P@1). BEAR was later proposed to support P@k scores while also handling multi-token entities. However, BEAR is computationally expensive, requiring PLMs to process $\mathcal{O}(qa)$ inputs, where q is the number of partially filled triplets and a is the number of tail-entity candidates. Since all methods above are token based, they also cannot disambiguate identically named tail entities. In contrast, our approach only requires $\mathcal{O}(q + a)$, whilst also supporting tail-entity disambiguation.

Knowledge Graph Completion A KG is defined as a set of triplets $\mathcal{T} \ni (h, r, t)$, where h represents the head entity, r the relation type and t the tail entity, with $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ the set of relation types. Therefore, KGC aims to learn a correct map, $f : (h, r, ?) \rightarrow t$. KGC models are trained on $\mathcal{T}_{\text{train}}$ and tested against $\mathcal{T}_{\text{test}}$, ensuring $\mathcal{T}_{\text{train}} \cap \mathcal{T}_{\text{test}} = \emptyset$. We

Template 1 Tail Entity Encoding Template where e_{name} and $e_{\text{description}}$ are replaced by their textual representations.

1: e_{name} - $e_{\text{description}}$
2: This sentence: "{word}" means in one word: "{one word}"
3: This sentence: "e_{name}" means in one word: "

further define subsets of KGC as follows.

Inductive KGC Setting In this setting, all entities found in the test set are never found in the training set, thereby $\mathcal{E}_{\text{test}} \cap \mathcal{E}_{\text{train}} = \emptyset$ and $\mathcal{R}_{\text{test}} \subseteq \mathcal{R}_{\text{train}}$.

Relationally Inductive KGC Setting In this setting, all the relation types found in the test set are never found in the training set, thereby $\mathcal{R}_{\text{train}} \cap \mathcal{R}_{\text{test}} = \emptyset$ and $\mathcal{E}_{\text{test}} \subseteq \mathcal{E}_{\text{train}}$.

Encoder-Based KGC Model These models generate embedding vectors for both the partial triplet and the tail entities. They define a similarity measure ϕ and learns two functions: $f_{\text{hr}} : (h, r, ?) \rightarrow \mathbf{e}_{\text{hr}}$ and $f_{\text{t}} : t \rightarrow \mathbf{e}_{\text{t}}$, such that tail entities, when sorted by $\phi(\mathbf{e}_{\text{hr}}, \mathbf{e}_{\text{t}})$ preserve the ranking of t as induced by $p(t|(h, r, ?))$, allowing for equivalent sorting of t by $\phi(\mathbf{e}_{\text{hr}}, \mathbf{e}_t)$ or $p(t|(h, r, ?))$.

3 Model Architecture

DEER is an encoder-based KGC model capable of predicting missing relations in a transductive, inductive, or relationally inductive setting. It creates both $\mathbf{e}_{hr}$ and $\mathbf{e}_t$ from a generative PLM using the PromptEOL method and employs cosine similarity as ϕ. The method requires two functions mapping entities to their textual name and textual description $f_{\text{name}} : e \rightarrow e_{\text{name}}$, $f_{\text{description}} : e \rightarrow e_{\text{description}}$ and another, mapping relations to their names, $f_{\text{relation}} : r \rightarrow r_{\text{name}}$. Following paragraphs describe the prompt templates used to acquire the embeddings, $\mathbf{e}_{hr}$ and $\mathbf{e}_t$.

Tail Entities Encoding Template Template 1 is used to encode the tail entities. This template is similar to the original PromptEOL template [11], but line 2 replaces the few-shot examples to prevent introducing bias, and adds line 1 to disambiguate identically named entities.

Head-Entity, Relation Encoding Template Here we propose two separate templates for generating $\mathbf{e}_{\text{hr}}$: first, a probing template designed to examine the knowledge recall ability of PLMs and second, a KGC template designed to maximize the KGC performance. Both templates are prefixed by an 8 shot example, S, generated from randomly sampled triplets from the training set.

Template 2 Probing Template used to generate e_{hr} .

1: h_{name} - $h_{\text{description}}$
2: (h_{name}, r_{name},

Probing Template For knowledge probing, the following template is used—(h_{name}, r_{name}—where h_{name} is replaced by the head entity's name and r_{name} by the relation's name. Note that no head-entity descriptions are provided to prevent the model from inferring relations of unmemorized entities. The example, S, is generated by concatenating the sampled triplets, each on a new line. Below is an example of a complete template, when h_{name}="deer" and r_{name}="hypernym":

(dress up, verb group, trick up)
⋮
(disfavour, hypernym, single out)
(deer, hypernym,

KGC Template When completing a KG, we additionally provide a head-entity description to enhance performance. This is achieved using Template 2, which is prefixed with an 8-shot example generated by the following template: $h_{i,\text{name}} - h_{i,\text{description}} \backslash \text{n}(h_{i,\text{name}}, r_{i,\text{name}}, t_{i,\text{name}})$.

4 Experiments and Results

In this work, we conduct two experiments. First, we perform a relationally inductive KGC experiment to demonstrate the advantage of our method in making out-of-distribution predictions. Second, we conduct the LAMA agreement experiment to investigate the degree of agreement with LAMA. WN18RR dataset [13] was used in both experiments and textual descriptions of entities provided by KG-BERT [14] were used as $f_{\text{description}}$. As in the original PromptEOL work, the OPT [1)] [15] was used as a PLM.

4.1 Experimental Setups

Relationally Inductive KGC Experiment A relationally inductive dataset was constructed using a transductive split of WN18RR. Triplets containing a specific relation type were removed from the training set and subsequently combined to form the test set. To represent a baseline performance of a fine-tuned encoder-base model, the SimKGC model was trained on the dataset with hyperparameters identical to the original work. The dataset was then used to evaluate the DEER performance with the

1) https://huggingface.co/facebook/opt-6.7b

Table 2 Relational Inductive Setting (Excluding 'Also See'). MR: mean rank, MRR: mean reciprocal rank, Iml: instruction–tuned models. # of entities: 40,943, # of test triplets: 1,299.

Model Name	Hit@1	Hit@3	Hit@10	MR	MRR
SimKGC	*0.27*	*1.07*	*3.19*	*11623*	*0.0136*
Ours-125M	0.08	0.31	0.69	12422	0.0036
Ours-350M	0.15	0.23	1.23	12919	0.0052
Ours-1.3B	0.54	6.16	17.71	1785	0.0597
Ours-iml-1.3B	0.62	7.54	**21.32**	1438	0.0718
Ours-6.7B	**1.53**	**8.31**	20.55	**1251**	**0.0787**

KGC template used for this experiment.

LAMA Agreement Experiment The test split of the transductive WN18RR dataset was adapted for LAMA by removing all triplets with multi-token tail entities. Both LAMA and DEER scored all single-token entities, with LAMA assigning the same scores to identically named entities due to disambiguation issues. The probing template was used instead of a cloze-style QA to maintain consistent bias with our method's bias. As shown in A.1, the predicted scores were ranked in descending order, and the tail entities' ranks were recorded. Pearson correlation between the rankings was computed across various model sizes, with a logarithmic scale applied to account for the reduced importance of rank differences at higher ranks.

4.2 Main Results

Relationally Inductive KGC Experiment Table 2 presents the results of the relationally inductive experiment, where the relation type "Also See" was removed from the train dataset. SimKGC, based on the 108M parameter BERT-base model, outperformed our models of comparative parameter size, DEER-125M, across all metrics. However, our models with super-billon parameter size outperformed SimKGC, despite not being fine-tuned, achieving +467% relative (δ) and +1.26% absolute (Δ) improvement for Hit@1, δ = +677%, Δ = +7.27% for Hit@3 and δ = +568% with Δ = +18% for Hit@10.

LAMA Agreement Experiment Table 3 presents the result of the LAMA agreement experiment. Log(rank) demonstrated stronger correlation than linear ranking, likely due to the increased sensitivity of rank difference at higher scores. Sub-billion models showed less correlation than super-billion models, with r = 0.373 for OPT-125M and r = 0.612 for OPT-350M, possibly due to the limitations of smaller models in summarizing entities with a single word as instructed. However, as shown in Figure 2, super-billion models exhibited high Log(rank) correlation with r = [0.710, 0.806], indicating a strong alignment between our method and LAMA.

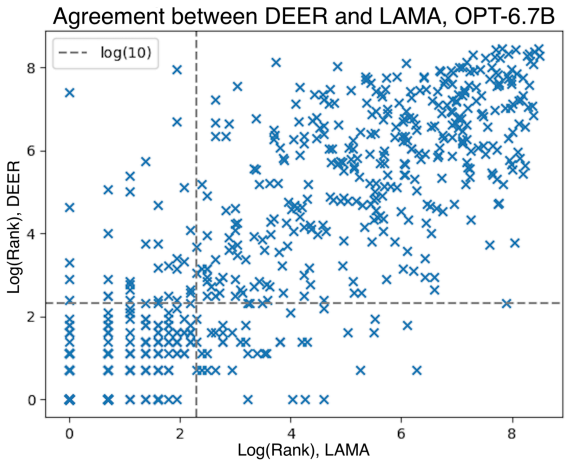

Figure 2 A scatter plot indicating a correlation between log(tail-entity rank) of LAMA and DEER.

Table 3 Pearson's correlation between DEER and LAMA Predictions of target tail-entity ranks and log(target tail-entity rank). # of entities: 4,948, # of test triplets: 16,521.

	Rank		Log(Rank)	
PLM Name	r	p-value	r	p-value
OPT-125M	0.466	1.75×10^{-33}	0.373	3.82×10^{-21}
OPT-350M	0.387	8.86×10^{-23}	0.612	1.30×10^{-62}
OPT-1.3B	0.472	1.87×10^{-34}	0.726	1.25×10^{-98}
OPT-iml-1.3B	0.551	1.268×10^{-48}	0.767	2.46×10^{-116}
OPT-6.7B	**0.596**	1.59×10^{-58}	**0.806**	1.6×10^{-137}
OPT-30B	0.487	7.56×10^{-37}	0.763	9.63×10^{-115}
OPT-iml-30B	0.462	6.75×10^{-33}	0.710	1.61×10^{-92}

5 Conclusion

We introduced DEER, a novel few-shot encoder-based KGC model that leverages a generative PLM. This approach retains the strengths of generative PLMs, such as few-shot capability and extensive parametric knowledge while addressing their drawbacks, including the need for entity linking, entity disambiguation, and limited ranking ability. Furthermore, it overcomes BEAR's limitation by enabling linear inference time complexity.

Our experiments show that DEER outperforms a fine-tuned SimKGC model in relationally inductive settings and aligns closely with LAMA. Its few-shot capabilities make it effective for completing KGs where obtaining an ideal training dataset is difficult. Moreover, its alignment with LAMA offers a promising avenue for future knowledge probing research, potentially offering deeper insights into knowledge recall abilities of generative PLMs.

References

[1] Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**, pp. 4281–4294, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[2] Xin Xie, Ningyu Zhang, Zhoubo Li, Shumin Deng, Hui Chen, Feiyu Xiong, Mosha Chen, and Huajun Chen. From discrimination to generation: Knowledge graph completion with generative transformer. In **Companion Proceedings of the Web Conference 2022**, WWW '22, p. 162–165, New York, NY, USA, 2022. Association for Computing Machinery.

[3] Xin Xie, Zhoubo Li, Xiaohan Wang, ZeKun Xi, and Ningyu Zhang. LambdaKG: A library for pre-trained language model-based knowledge graph embeddings. In Sriparna Saha and Herry Sujaini, editors, **Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: System Demonstrations**, pp. 25–33, Bali, Indonesia, November 2023. Association for Computational Linguistics.

[4] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, pp. 5418–5426, Online, November 2020. Association for Computational Linguistics.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[6] Dora Jambor, Komal Teru, Joelle Pineau, and William L. Hamilton. Exploring the limits of few-shot link prediction in knowledge graphs. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, **Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume**, pp. 2816–2822, Online, April 2021. Association for Computational Linguistics.

[7] Xin Zhao, Naoki Yoshinaga, and Daisuke Oba. What matters in memorizing and recalling facts? multifaceted benchmarks for knowledge probing in language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Findings of the Association for Computational Linguistics: EMNLP 2024**, pp. 13186–13214, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[8] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, **Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)**, pp. 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics.

[9] Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. Give me the facts! a survey on factual knowledge probing in pre-trained language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 15588–15605, Singapore, December 2023. Association for Computational Linguistics.

[10] Jacek Wiland, Max Ploner, and Alan Akbik. BEAR: A unified framework for evaluating relational knowledge in causal and masked language models. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, **Findings of the Association for Computational Linguistics: NAACL 2024**, pp. 2393–2411, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[11] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, **Findings of the Association for Computational Linguistics: EMNLP 2024**, pp. 3182–3196, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[12] Jan-Christoph Kalo and Leandra Fichtel. Kamel: Knowledge analysis with multitoken entities in language models. In **AKBC**, 2022.

[13] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In **Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence**, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

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

[15] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[16] Komal Teru, Etienne Denis, and Will Hamilton. Inductive relation prediction by subgraph reasoning. In Hal Daumé III and Aarti Singh, editors, **Proceedings of the 37th International Conference on Machine Learning**, Vol. 119 of **Proceedings of Machine Learning Research**, pp. 9448–9457. PMLR, 13–18 Jul 2020.

A Appendix

A.1 LAMA Agreement Experiment

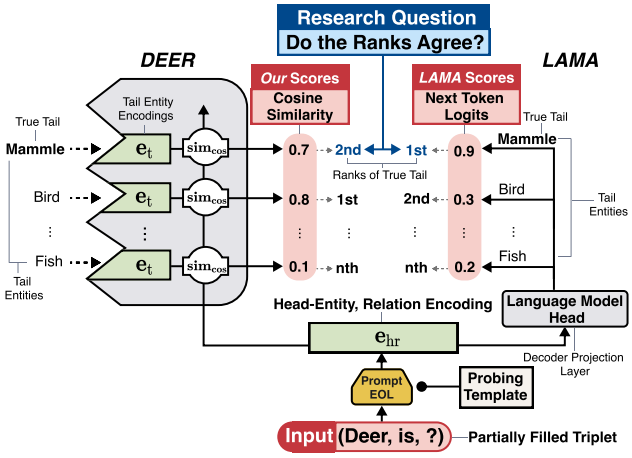

Figure 3 A diagram illustrating data-flow in the LAMA Agreement Experiment. Note, an identical vector $\mathbf{e}_{\text{hr}}$, is used for the cosine similarity in our method and token prediction in LAMA.

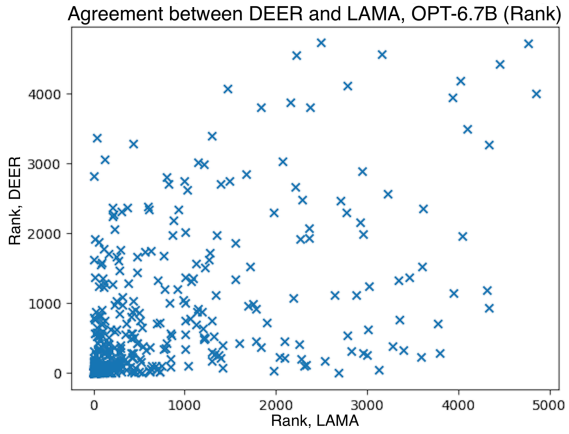

Figure 4 A scatter plot showing correlation between tail-entity tank of DEER and LAMA.

The logit value of true-tail entity in LAMA and the Softmaxed Cosine Similarity value of the entity in DEER was additionally compared. Table 5 shows the Pearson correlation between the two values, and Figure 5 shows a scatter plot between the values with OPT-6.7B.

Table 4 Correlation of LAMA's logits and softmaxed cos sim.

Model	Pearson Correlation	p-value
OPT-125M	0.405	6.66×10^{-25}
OPT-350M	0.498	1.33×10^{-38}
OPT-1.3B	0.724	5.48×10^{-98}
OPT-iml-1.3B	0.795	5.00×10^{-131}
OPT-6.7B	0.816	2.34×10^{-143}
OPT-30B	0.717	3.31×10^{-95}
OPT-iml-30B	0.672	1.468×10^{-79}

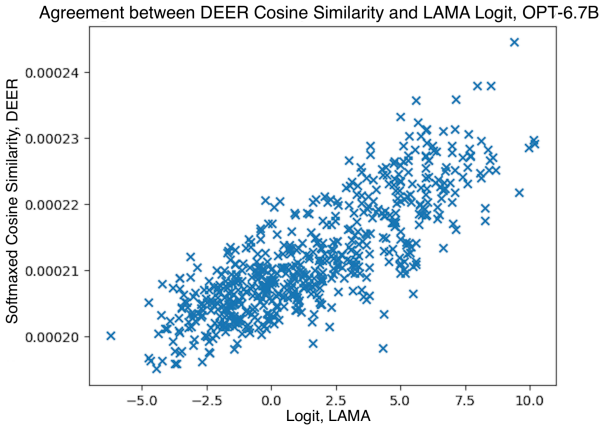

Figure 5 A scatter plot comparing softmaxed cosine similarity of $\mathbf{e}_{\text{hr}}$, $\mathbf{e}_{\text{t}}$ in DEER and logits of tail-entity token in LAMA, indicating a correlation between the two values.

Table 5 A table illustrating the difference between Hit@K and MRR between LAMA and our method across different parameters size. The setup is identical to the LAMA agreement experiment.

	Hit@1	Hit@10	Hit@500	MRR
LAMA-125M	0.168	3.86	35.1	0.0154
Ours-125M	0.168	2.68	39.4	0.0124
Relative Error, %	**0.0**	31	12	19
LAMA-350M	3.02	12.4	38.4	0.0573
Ours-350M	0.336	2.85	36.2	0.0157
Relative Error, %	89	77	5.7	73
LAMA-1.3B	4.19	21.8	56.4	0.0997
Ours-1.3B	2.52	15.9	52.2	0.0699
Relative Error, %	40	27	7.4	30
LAMA-iml-1.3B	5.03	24.5	62.9	0.112
Ours-iml-1.3B	9.23	33.9	74.7	0.176
Relative Error, %	83	38	19	57
LAMA-6.8B	7.38	33.2	55.5	0.156
Ours-6.8B	8.39	33.9	53.0	0.165
Relative Error, %	14	**2.0**	**3.6**	**5.2**
LAMA-30B	5.87	32.6	77.7	0.143
Ours-30B	10.2	39.8	78.4	0.199
Relative Error, %	73	22	0.90	39
LAMA-iml-30B	8.05	35.4	78.9	0.163
Ours-iml-30B	9.40	32.9	73.2	0.175
Relative Error, %	17	10	7.2	7.4

A.2 Inductive KGC Experiment

Table 6 Performance of DEER on inductive split of WN18RR [16]. KGC Template was used.

	Hit@1	Hit@10	Hit@1000	MRR
OPT-125M	0.6 ± 0.7	6 ± 6	40 ± 10	0.02 ± 0.02
OPT-350M	0.6 ± 0.8	5 ± 6	30 ± 10	0.02 ± 0.02
OPT-1.3B	2.3 ± 0.9	25 ± 7	70 ± 10	0.10 ± 0.03
OPT-iml-1.3B	2.3 ± 0.8	27 ± 9	77 ± 9	0.10 ± 0.03
OPT-6.7B	1.6 ± 0.7	28 ± 3	75 ± 6	0.10 ± 0.01
OPT-30B	1.7 ± 0.3	$\mathbf{35 \pm 3}$	$\mathbf{82 \pm 3}$	$\mathbf{0.12 \pm 0.01}$
OPT-iml-30B	$\mathbf{2.9 \pm 0.8}$	26 ± 5	73 ± 3	0.10 ± 0.02