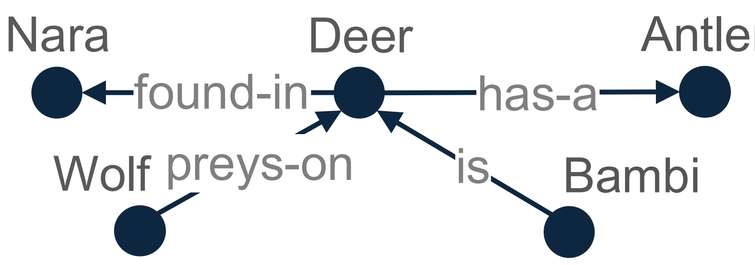
[Q10-11] Cosine Similarity as Logits?: Few-shot Knowledge Graph Completion with Embedding Vectors of a Generative PLM and its Application in Knowledge Probing

Tomoyuki Jinno[1] Kazuki Hayashi[1] Yusuke Sakai[1] Hidetaka Kamigaito[1] Taro Watanabe[1]
(1. Nara Institute of Science and Technology)

Encoder-based Knowledge Graph Completion with Few-Shot Capability

- Problems: Encoder-Based KGCs Require **Fine-Tuning** and Decoder-Based KGCs are **Slow**
- Solution: Propose an Encoder-Based KGC using **Embedding Vectors of a Decoder Model**
- Findings: **Outperforms SimKGC** in a Relationally Inductive Setting and **Agrees with LAMA**

Background: What is a KG and KGC?



What is a **Knowledge Graph** (KG)?

- Stores Relations Between Entities
- Relations Expressed as a Triplet:
  - (head entity, relation type, tail entity)
  - E.g., (BERT, *used-for*, classification)

What is **KG Completion** (KGC)?

- Predicts Missing Relations in a KG
  - Input: (*head entity*, *relation type*, ?)
  - Output: *tail entity*

Query — KGC Model — 【In Rank Format】 Tail-Entites

(Deer, *has-a*, ?) — In → — Out → Antler

Text-Based KGC Falls into:

1. **Encoder-Based**: Dense Retrieval Like
   Consists of:
   1. Query Encoder, f_q (Deer, *has-a*, ?) ⇨ ↗
   2. Tail Entity Encoder, f_t Antler ⇨ →
   3. Similarity Function, ϕ (↗, →) ⇨
2. **Decoder-Based**: Text-Generation Like

KGCs are also used for **Knowledge Probing**

- Aim: Evaluate Factual Knowledge Retrieval Capability of LLMs

Problems: Prior Methods

KG Completion Issues

	Pros 👍	Cons 👎
Enc-Based KGC	Fast	Requires **Training Data**
Dec-Based KGC	Few-Shot	**Slow** Inference

No KGC with **Encoder-Based Efficiency** and **In-Context Learning Capability**

Knowledge Probing Issues

	Scores	Pros 👍	Cons 👎
LAMA	Single Token Logits	Hit@k	**Single-Token**
KAMEL	String Match	Multi-Token	**Hit@1**
BEAR	Multi Token Logits	Hit@k and Multi-Token	$\mathcal{O}(Q \times T)$

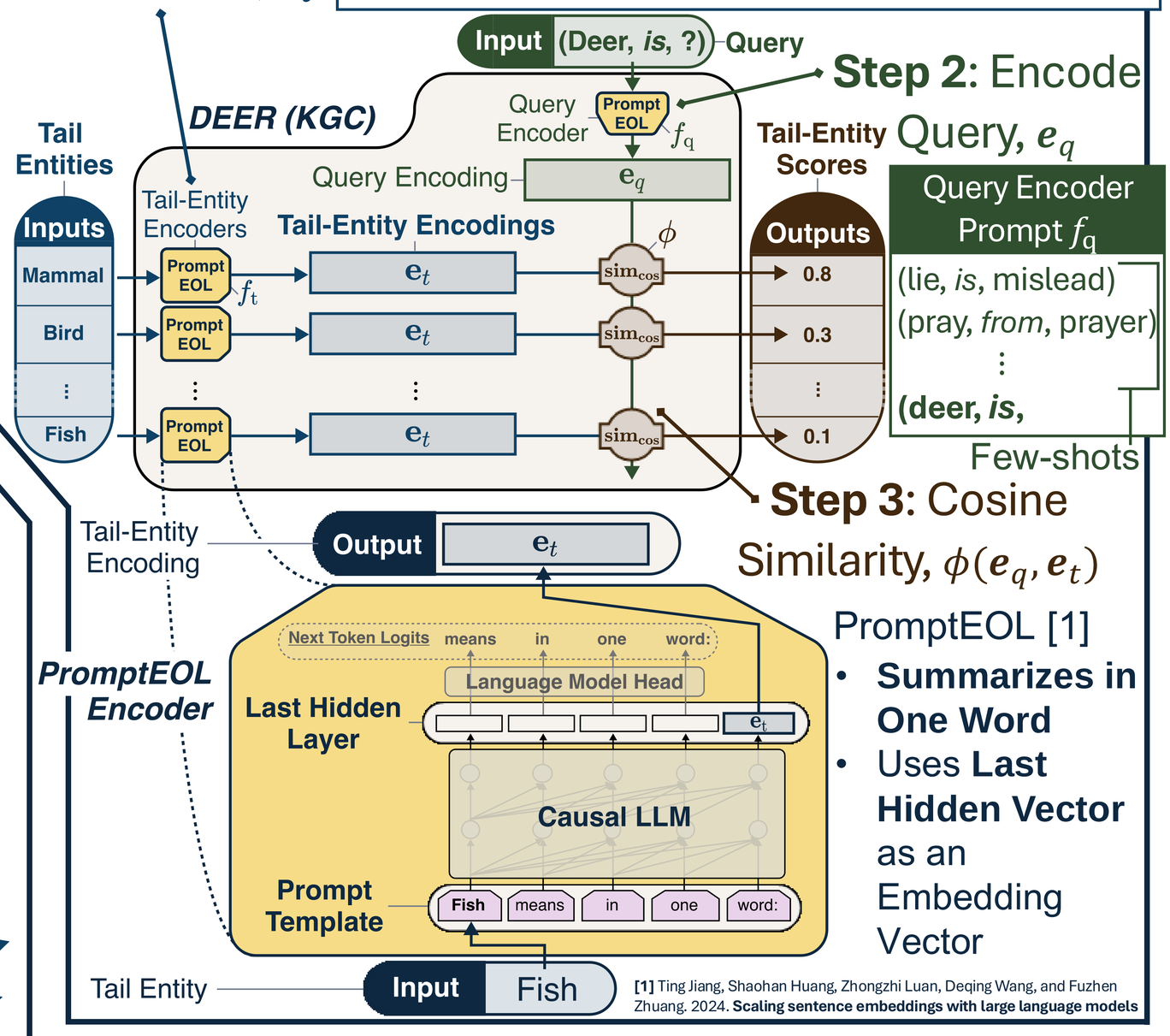
Our Prober: **Hit@k + Multi-Token + $\mathcal{O}(Q + T)$**

Method: DEcoder Embedding-Based Relational KGC

Use PromptEOL [1] as the Encoders f_q and f_t

Step 1: Encode Tail Entities, e_t

Mammal – Warm-blooded animal — Tail-Entity Prompt f_t
This sentence: "{word}" means in one word: "{one word}"
This sentence: "**Mammal**" means in one word:



PromptEOL [1]
- **Summarizes in One Word**
- Uses **Last Hidden Vector** as an Embedding Vector

[1] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. **Scaling sentence embeddings with large language models**

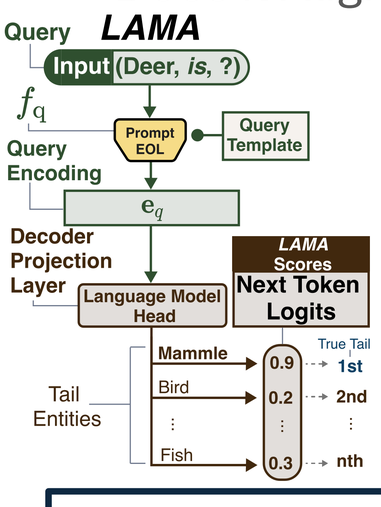
Results and Conclusion

Exp 1: KG Completion

WN18RR Results	Transductive Hit@1	Transductive Hit@10	Relationally Inductive Hit@1	Relationally Inductive Hit@10
SimKGC (Fine-Tuned Baseline)	**59%**	**80%**	0.77%	12.8%
DEER-30B (Ours)	1.5%	29%	**1.31%**	**24.2%**

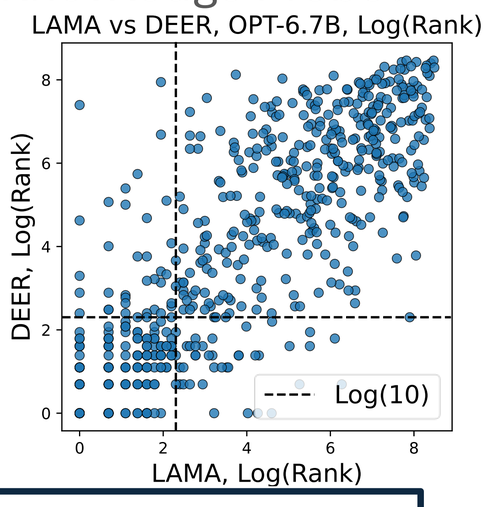
Exp 2: Knowledge Probing

Does it Align with an Existing Knowledge Probe?



Correlation between LAMA and DEER (Log Scale)

PLM	r
OPT-125M	0.373
OPT-350M	0.612
OPT-1.3B	0.726
OPT-6.7B	**0.806**
OPT-30B	0.763



Conclusion

- Exp 1: DEER outperforms a Fine-Tuned KGC Baseline in Relationally Inductive Setting of WN18RR
- Exp 2: DEER can be used for Knowledge Probing