Dynamic Injection of Entity Knowledge into Dense Retrievers

Ikuya Yamada[1,2] **Ryokan Ri**[3] **Takeshi Kojima**[4] **Yusuke Iwasawa**[4] **Yutaka Matsuo**[4]

[1]Studio Ousia [2]RIKEN AIP [3]SB Intuitions [4]The University of Tokyo

ikuya@ousia.jp ryou0634@gmail.com
{t.kojima,iwasawa,matsuo}@weblab.t.u-tokyo.ac.jp

Abstract

Dense retrievers often struggle with queries involving less-frequent entities due to their limited entity knowledge. We propose the Knowledgeable Passage Retriever (KPR), a BERT-based retriever enhanced with a context-entity attention layer and dynamically updatable entity embeddings. This design enables KPR to incorporate external entity knowledge without retraining. Experiments on three datasets demonstrate that KPR consistently improves retrieval accuracy, with particularly large gains on the EntityQuestions dataset. When built on the off-the-shelf bge-base retriever, KPR achieves state-of-the-art performance among similarly sized models on two datasets. Models and code are released at github.com/knowledgeable-embedding/knowledgeable-embedding.

1 Introduction

Language models (LMs) struggle to capture less-frequent or up-to-date entity knowledge (Kandpal et al., 2023; Mallen et al., 2023), often resulting in hallucinations (Shuster et al., 2021). Retrieval-augmented generation (RAG), which enhances LMs by leveraging external knowledge retrieval, is a promising approach to mitigate this issue. Dense retrievers are commonly employed for this purpose; however, because they also rely on LMs, they likewise struggle with queries involving less-frequent entities (Sciavolino et al., 2021) and often fail to retrieve such knowledge effectively.

In this paper, we address this problem by proposing a simple extension to dense retrievers that dynamically injects entity knowledge into their embeddings. Specifically, we introduce the Knowledgeable Passage Retriever (KPR), a BERT-based dense retriever (Karpukhin et al., 2020) that integrates entity embeddings adaptable at inference time (see Figure 1). KPR is intentionally designed with a simple architecture and is trained to attend to entity knowledge based on the context of the input

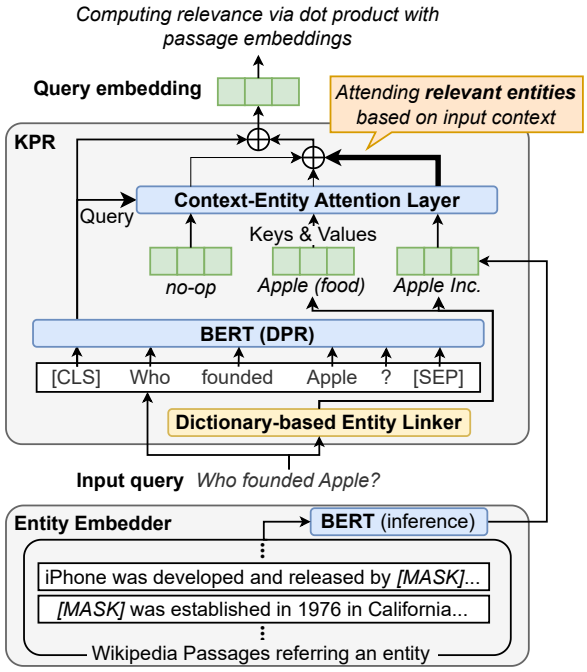

Figure 1: Architecture of KPR, a dense retriever with a context-entity attention layer that incorporates entity knowledge, given a query as input. Entity embeddings are obtained via BERT inference, and both the embeddings and the entity linker can be updated without retraining. Relevance is scored by the dot product of query and passage embeddings, both produced by KPR.

text, through a context-entity attention layer placed on top of BERT. Entity embeddings are obtained via single-pass BERT inference on texts referring to the entity and are kept frozen during training. KPR detects entities in the input text using a simple dictionary-based entity linker. Since both the entity embeddings and the linker can be modified after training, new entity knowledge can be easily added or updated without retraining.

To evaluate KPR, we use the EntityQuestions (EQ) dataset (Sciavolino et al., 2021), which includes many queries with less-frequent entities, as well as the Natural Questions (NQ) (Kwiatkowski et al., 2019) and TriviaQA (TQA) (Joshi et al.,

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 16867–16879

2017) datasets. KPR achieves a substantial 12.6% gain in top 20 accuracy on EQ over the model without KPR extensions and consistently improves performance on the other two datasets. We further build KPR on top of the off-the-shelf bge-base retriever, achieving state-of-the-art results on EQ and TQA compared to other retrievers of similar size.

2 Method

KPR is built upon DPR (Karpukhin et al., 2020), a widely used BERT-based dense retriever. We add an attention layer on top of DPR to incorporate entity knowledge from a knowledge base (KB) (see Figure 1). We use Wikipedia as the target KB.

DPR. DPR encodes a query or passage into a D-dimensional embedding, obtained from the output embedding of BERT's `[CLS]` token. Relevance between a query and a passage is scored as the dot product of their embeddings. The model is fine-tuned on datasets containing queries paired with positive and negative passages, using a cross-entropy loss with the dot product scores as logits. See Appendix A for further details.

Input entity representation. KPR uses the following two D-dimensional input embeddings:

- *Entity embedding*, assigned to each entity in the KB, represents the entity itself.
- *Entity position embedding*, assigned to each position in the input tokens, encodes positional information.

KPR takes the input tokens and a set of entities $E = e_1, \ldots, e_N$ detected from the text using an entity linker. The input representation of each entity is computed by summing its entity embedding and entity position embedding based on its position in the input token sequence. If an entity spans multiple tokens, its position embedding is computed as the average of the embeddings of the corresponding positions (Yamada et al., 2020b). To prevent the entity sequence from being empty, a D-dimensional *no-op* embedding is appended to the sequence of input entity representations.

Context-entity attention layer. KPR adopts a single-head key-query-value attention mechanism (Vaswani et al., 2017). The query matrix $\mathbf{Q}$ is computed based on the output embedding of BERT's `[CLS]` token (denoted by $\mathbf{H}_{\texttt{[CLS]}} \in \mathbb{R}^{1\times D}$). The key and value matrices, $\mathbf{K}$ and $\mathbf{V}$, are computed based on the input entity representations denoted by $\mathbf{U} \in \mathbb{R}^{N+1\times D}$:

$$\mathbf{Q} = \mathbf{H}_{\texttt{[CLS]}}\mathbf{X}_q, \;\; \mathbf{K} = \mathbf{U}\mathbf{X}_k, \;\; \mathbf{V} = \mathbf{U}\mathbf{X}_v,$$

where $\mathbf{X}_q \in \mathbb{R}^{D\times D}$, $\mathbf{X}_k \in \mathbb{R}^{D\times D}$, and $\mathbf{X}_v \in \mathbb{R}^{D\times D}$ are weight matrices. We aim for the attention mechanism to attend to useful entities based on the context by using entity embeddings as the key and the embedding of the `[CLS]` token, which encodes the context of the input text, as the query.

KPR computes the output embedding $\mathbf{Z} \in \mathbb{R}^{1\times D}$ as:

$$\mathbf{Y} = \text{activation}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D}}\right)\mathbf{V},$$
$$\mathbf{Z} = \ln\left(\text{dropout}\left(\mathbf{Y}\right) + \mathbf{H}_{\texttt{[CLS]}}\right),$$

where $\text{activation}(\cdot)$, $\ln(\cdot)$, and $\text{dropout}(\cdot)$ denote the activation, layer normalization, and dropout functions, respectively. We use the sigmoid function with a length-based bias, $\text{sigmoid}(x - \log N + 1)$ (Ramapuram et al., 2025), as the activation.

Entity embedder. We compute entity embeddings based on BERT inference over Wikipedia passages that refer to the corresponding entity (Ye et al., 2022). Specifically, for each passage containing an anchor link to the entity, we replace the entity name with BERT's `[MASK]` token, extract the output embedding of the `[MASK]` token, and average the resulting embeddings across passages. The final embeddings are then normalized to match the average norm of BERT's input token embeddings.

Entity linker. We use dictionary-based string matching to detect entities mentioned in the text. The dictionary comprises anchor names and their possible referent entities, obtained from Wikipedia hyperlinks. For ambiguous names (e.g., "Apple," which can refer to *Apple Inc.* and *Apple (food)*), we do not disambiguate to a single entity, but instead input all possible candidates to maintain recall.

Training. The parameters in our attention layer, entity position embeddings, and no-op embedding are initialized randomly. Entity embeddings are kept frozen during training, enabling them to be added or updated dynamically after training. The dropout probability in the attention layer is set to match that of the other layers. We follow DPR (Karpukhin et al., 2020) for the remaining training settings including the loss function. Note that

entities can be dynamically updated in KPR without retraining, since KPR relies on entity embeddings computed via BERT inference, which are kept frozen during training, and on a dictionary-based entity linker, whose entries can be modified.

Computational overhead. KPR adds entity embeddings to BERT, requiring additional storage. However, since the embeddings are used only as input features, they can be implemented as a sparse lookup table and stored not only in GPU memory but also in CPU memory or even on disk, with minimal impact on speed (Yu et al., 2025). Furthermore, KPR introduces only a small overhead in total FLOPs, as discussed in Appendix B.

3 Experiments

Entity set. We use 7.2M English Wikipedia entities to construct our entity embeddings and linker.

Datasets and metrics. We train our model on the dataset proposed by Karpukhin et al. (2020), which is based on NQ, TQA, WebQuestions (Berant et al., 2013), and CuratedTREC (Baudiš and Šedivý, 2015). We evaluate the models on the EQ, NQ, and TQA datasets. Following Sciavolino et al. (2021), we report top 20 retrieval accuracy. Top 100 accuracies are also provided in Appendix D. We use the 21M Wikipedia passages released by Karpukhin et al. (2020) as the target passages.

Baselines. We use vanilla DPR and conventional BM25, which performs robustly on queries with less frequent entities (Sciavolino et al., 2021).

Base models. We use base-sized BERT and RetroMAE (Xiao et al., 2022), a state-of-the-art BERT-based model pretrained for retrieval. For DPR, we also adopt PELT (Ye et al., 2022), a BERT-based model that injects entity knowledge by inserting each entity embedding, placed between the token embeddings of parentheses, immediately after the corresponding entity token. While PELT introduces entity knowledge through input augmentation, KPR incorporates it via a dedicated attention layer. Unlike KPR, PELT requires each input entity name to be disambiguated to a single entity, which incurs additional computational complexity.

Entity embeddings. We use the same entity embeddings, derived from base-sized BERT inference (§2), for both PELT and all KPR models.

Model	Base Model	Entity Linker	EQ	NQ	TQA	Avg
BM25	–	–	71.2	59.1	66.9	66.0
DPR_{BERT}	BERT	–	56.8 ±0.3	79.3 ±0.2	78.9 ±0.1	71.7
KPR_{BERT}	BERT	Dictionary	69.4 ±0.4	80.7 ±0.3	79.7 ±0.1	76.6
DPR_{PELT}	BERT	ReFinED	63.8 ±0.8	79.8 ±0.2	79.0 ±0.1	74.2
KPR_{BERT}	BERT	ReFinED	69.0 ±0.4	80.4 ±0.2	79.8 ±0.1	76.4
$\text{DPR}_{\text{RetroMAE}}$	RetroMAE	–	70.9 ±0.2	81.0 ±0.2	81.2 ±0.1	77.7
$\text{KPR}_{\text{RetroMAE}}$	RetroMAE	Dictionary	**74.7** ±0.4	**83.0** ±0.3	**82.1** ±0.1	**79.9**

Table 1: Top 20 accuracies of KPR and baseline models. For clarity, models are grouped by their base pretrained models and entity linkers. We report mean accuracy and 95% confidence intervals based on Student's t-distribution over 5 training runs with different random seeds. The best overall mean scores are in bold; the best within each group are underlined. Top 100 accuracies are provided in Table 5.

Model	EQ	NQ	TQA	Avg
Entity embeddings:				
Random initialization	64.9	80.3	79.0	74.7
Wikipedia2Vec	65.9	80.4	79.2	75.2
BERT intermediate layer #3	66.2	80.3	79.4	75.3
BERT intermediate layer #6	66.8	80.3	79.4	75.5
BERT intermediate layer #9	67.8	80.4	79.6	75.9
BERT last layer (KPR_{BERT})	69.3	80.6	79.7	76.5
Activation functions:				
Softmax function	66.3	80.5	79.5	75.4
Sigmoid function with length bias	69.3	80.6	79.7	76.5

Table 2: Top 20 accuracy of KPR_{BERT} with different entity embeddings and activation functions. Unlike Table 1, the results are based on a single training run. The best score in each group is underlined.

Entity linker. By default, we use our dictionary-based linker (§2) and additionally employ the state-of-the-art ReFinED linker (Ayoola et al., 2022). We select ReFinED over other off-the-shelf systems such as BLINK (Wu et al., 2020), GENRE (Cao et al., 2021), and LUKE (Yamada et al., 2022) due to its superior performance.

We refer to KPR based on BERT as KPR_{BERT} and name the other models accordingly. Further details are provided in Appendix C.

3.1 Results and Analysis

Main results. Table 1 shows that KPR significantly outperforms all baselines across datasets and base models. KPR_{BERT} surpasses DPR_{BERT} by 12.6% on EQ and 4.9% on average. Using the same ReFinED linker, KPR_{BERT} consistently outperforms DPR_{PELT}, with a 5.2% gain on EQ,

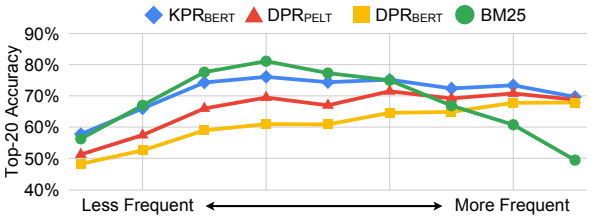

Figure 2: Top 20 retrieval accuracy of KPR$_{BERT}$, DPR$_{PELT}$, DPR$_{BERT}$, and BM25 on EQ, grouped into 10 bins based on entity frequency in Wikipedia.

Model	EQ	NQ	TQA	Avg
contriever (Izacard et al., 2022)	63.0	67.9	73.9	68.3
gte-base (Li et al., 2023)	70.8	77.0	76.6	74.7
e5-base (Wang et al., 2022)	72.4	**86.2**	81.4	80.0
bge-base (Xiao et al., 2024)	71.0	82.3	80.0	77.8
KPR$_{bge\text{-}base}$	**76.8**	82.4	**81.5**	**80.2**

Table 3: Top 20 accuracies of KPR based on bge-base and off-the-shelf retrievers on EQ, NQ, and TQA. Top 100 accuracies are provided in Table 6.

suggesting that the entity knowledge injected by PELT is partially lost within BERT and highlighting the benefit of KPR's attention layer placed on top. KPR$_{RetroMAE}$ achieves the best performance across all datasets, demonstrating its effectiveness even with a strong pretrained model. The gain is notably larger on EQ in all settings, as it includes many queries involving less-frequent entities. Further analysis of the EQ results is provided below.

Additional results on the MS MARCO dataset are presented in Appendix E.

Effects of entity linker. Table 1 also shows that KPR$_{BERT}$ with the dictionary-based linker outperforms its ReFinED-based variant on average. This is somewhat surprising, as the dictionary linker simply applies string matching without disambiguation (§2) and may detect incorrect or noisy entities. We attribute this to ReFinED's slightly lower recall, as it extracts entities only when confident. For example, ReFinED detects 0.93 entities per query on average in EQ, compared to 0.97 with the dictionary linker. These results also suggest that KPR is robust to noise, likely due to its attention mechanism's ability to focus on contextually relevant entities. See Appendix F for further analysis.

How do entity embeddings affect performance? We evaluate two baselines for computing entity embeddings: random initialization and Wikipedia2Vec embeddings (Yamada et al., 2020a), both based on the same Wikipedia dump as our entity embeddings. We also test embeddings extracted from an intermediate BERT layer, motivated by prior work suggesting that entity knowledge is well captured there (Meng et al., 2022).

Table 2 shows that BERT-based embeddings consistently outperform the alternatives. Notably, using the last layer yields the best performance across all datasets. This is likely because $\mathbf{H}_{\texttt{[CLS]}}$, used in our context-entity attention layer, is also taken from the last layer, which may help the model better capture relevance between context and entities.

Sigmoid vs. softmax. We evaluate the effect of replacing the sigmoid activation in our attention layer with the softmax function, which is commonly used in attention mechanisms. Table 2 shows that sigmoid consistently outperforms softmax across all datasets. We attribute this to the fact that sigmoid allows the model to assess each entity's relevance independently, without being influenced by the presence of other relevant entities.

Analysis of EQ results. To examine whether KPR's improvement stems from incorporating knowledge of less-frequent entities, we divide the EQ examples into 10 bins based on the frequency of the entity in each query and measure performance within each bin. This is feasible because every EQ query contains a single entity. We obtain entity frequencies from Wikipedia hyperlinks and create 10 log-spaced bins ranging from 1 to 10,000.

Figure 2 shows that KPR$_{BERT}$ is more consistent across entity frequencies than other models and outperforms DPR$_{BERT}$ and DPR$_{PELT}$, especially on queries with less-frequent entities. Compared to BM25, it performs comparably on less-frequent entities and substantially better on frequent ones.

Analysis of KPR's Attention Mechanism. Appendix F provides a qualitative analysis of our attention mechanism. We observe that KPR tends to assign lower weights to common entities and generally higher weights to correct entities than to incorrect ones, as shown in Figure 3.

3.2 Pushing State-of-the-Art

To evaluate the effectiveness of KPR on recent off-the-shelf retrievers, we select the bge-base model (Xiao et al., 2024) and train KPR$_{bge\text{-}base}$ using it as the base model. Since bge-base is already trained on large-scale, high-quality datasets, we freeze the base model to prevent catastrophic forgetting and train only the newly introduced parameters. Detailed settings are provided in Appendix C.

Table 3 presents results comparing our model with recent off-the-shelf retrievers. $\text{KPR}_{\text{bge-base}}$ consistently outperforms bge-base across all datasets, with a substantial 5.8% gain on EQ. It also achieves the highest average performance among all strong off-the-shelf retrievers.

4 Related Work

The performance of dense retrievers (Karpukhin et al., 2020; Guu et al., 2020) remains insufficient for queries involving less-frequent entities (Sciavolino et al., 2021), due to the limited knowledge of such entities in LMs (Kandpal et al., 2023; Mallen et al., 2023). Several studies have explored incorporating entity knowledge to enhance LMs (Zhang et al., 2019; Yamada et al., 2020b; Ye et al., 2022; Zhang et al., 2023b), among which we adopt PELT (Ye et al., 2022) as a baseline.

Entity knowledge has also been leveraged to improve retrieval tasks (Liu and Fang, 2015; Xiong et al., 2017a,b; Liu et al., 2018; Tran and Yates, 2022; Nguyen et al., 2024; Chatterjee et al., 2024). However, to our knowledge, only Tran and Yates (2022) is directly applicable to dense retrieval, integrating Wikipedia2Vec embeddings into a BERT-based retriever. Because its entity representations are computed independently of BERT, the model cannot learn to select relevant entities based on context. As a result, it fails to improve performance compared to the vanilla model when passages are represented by a single embedding. Moreover, its effectiveness on queries involving less-frequent entities remains unexamined, and updating entity embeddings requires retraining Wikipedia2Vec. In contrast, KPR integrates an attention layer into BERT, enabling it to select contextually relevant entities more effectively. This leads to improved performance across multiple retrieval benchmarks.

5 Conclusions

We proposed KPR, a dense retriever with a context-entity attention layer and dynamically updatable entity knowledge. KPR consistently improves performance across benchmarks, particularly on queries involving less-frequent entities. When built on bge-base, it achieves state-of-the-art results on two benchmarks among similarly sized models. Future work includes applying KPR to decoder-based retrievers and extending it to KBs beyond Wikipedia.

Limitations

This work focuses on English-language datasets and assumes the availability of a KB, namely Wikipedia. While the proposed method is modular by design, its effectiveness in languages other than English or in domains lacking a comprehensive KB remains unexplored.

References

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, C. K. Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Shunting Zhang, Michael Suo, Phil Tillet, Xu Zhao, Eikan Wang, Keren Zhou, Richard Zou, Xiaodong Wang, Ajit Mathews, William Wen, Gregory Chanan, Peng Wu, and Soumith Chintala. 2024. PyTorch 2: Faster machine learning through dynamic Python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, page 929–947.

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 209–220.

Petr Baudiš and Jan Šedivý. 2015. Modeling of the question answering task in the YodaQA system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction*, page 222–228.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Shubham Chatterjee, Iain Mackie, and Jeff Dalton. 2024. DREQ: Document re-ranking using entity-based query understanding. In *Advances in Information Retrieval*, pages 210–229.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652v1*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3929–3938.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in Python.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, pages 15696–15707.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361v1*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980v9*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Hyunji Lee, Luca Soldaini, Arman Cohan, Minjoon Seo, and Kyle Lo. 2023. Back to basics: A simple recipe for improving out-of-domain retrieval in dense encoders. *arXiv preprint arXiv:2311.09765v1*.

Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Mario Šaško, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Victor Sanh, Teven Le Scao, Kevin Canwen Xu, Nicolas Patry, Steven Liu, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Nathan Raw, Sylvain Lesage, Anton Lozhkov, Matthew Carrigan, Théo Matussière, Leandro von Werra, Lysandre Debut, Stas Bekman, and Clément Delangue. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281v1*.

Xitong Liu and Hui Fang. 2015. Latent entity space: a novel retrieval approach for entity-bearing queries. *Inf. Retr.*, 18(6):473–503.

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.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, pages 17359–17372.

Rada Mihalcea and Andras Csomai. 2007. Wikify! linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, page 233–242.

David Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, page 509–518.

Thong Nguyen, Shubham Chatterjee, Sean MacAvaney, Iain Mackie, Jeff Dalton, and Andrew Yates. 2024. DyVo: Dynamic vocabularies for learned sparse retrieval with entities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 767–783.

Jason Ramapuram, Federico Danieli, Eeshan Gunesh Dhekane, Floris Weers, Dan Busbridge, Pierre Ablin,

Tatiana Likhomanenko, Jagrit Digani, Zijin Gu, Amitis Shidani, and Russell Webb. 2025. Theory, analysis, and best practices for sigmoid self-attention. In *The Thirteenth International Conference on Learning Representations*.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 3505–3506.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803.

Daniil Sorokin and Iryna Gurevych. 2018. Mixing context granularities for improved entity linking on question answering data across entity categories. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 65–75.

Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. 2016. On generating characteristic-rich question sets for QA evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 562–572.

Hai Dang Tran and Andrew Yates. 2022. Dense retrieval with entity views. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, page 1955–1964.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533v2*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. 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.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6397–6407.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022. RetroMAE: Pre-training retrieval-oriented language models via masked auto-encoder. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 538–548.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 641–649.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017a. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 763–772.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017b. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th International Conference on World Wide Web*, page 1271–1279.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020a. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020b. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6442–6454.

Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. 2022. Global entity disambiguation with BERT. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3264–3271.

Deming Ye, Yankai Lin, Peng Li, Maosong Sun, and Zhiyuan Liu. 2022. A simple but effective pluggable entity lookup table for pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 523–529.

Da Yu, Edith Cohen, Badih Ghazi, Yangsibo Huang, Pritish Kamath, Ravi Kumar, Daogao Liu, and Chiyuan Zhang. 2025. Scaling embedding layers in language models. *arXiv preprint arXiv:2502.01637v1*.

Xinyu Zhang, Kelechi Ogueji, Xueguang Ma, and Jimmy Lin. 2023a. Toward best practices for training multilingual dense retrieval models. *ACM Trans. Inf. Syst.*, 42(2).

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Huadong Wang, Deming Ye, Chaojun Xiao, Xu Han, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2023b. Plug-and-play knowledge injection for pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10641–10658.

A Overview of DPR

DPR is a BERT-based model that encodes a query q and a passage p into D-dimensional embeddings, denoted as $\mathbf{e}_q \in \mathbb{R}^D$ and $\mathbf{e}_p \in \mathbb{R}^D$. The embedding is taken from the output embedding of BERT's `[CLS]` token. To construct passage p, the passage title and text are concatenated with BERT's `[SEP]` tokens as: `[CLS]` *passage title* `[SEP]` *passage text* `[SEP]`. Given query q, the relevance of passage p is computed as the dot product $\langle \mathbf{e}_q, \mathbf{e}_p \rangle$.

Training. Let $\mathcal{D} = \{\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,N}^- \rangle\}_{i=1}^M$ be a set of M training instances, where each instance consists of a query q_i, a positive passage p_i^+ relevant to the query, and N negative passages $p_{i,j}^-$ irrelevant to the query. The model is trained by minimizing the negative log-likelihood of the positive passage:

$$\mathcal{L} = -\log \frac{\exp(\langle \mathbf{e}_{q_i}, \mathbf{e}_{p_i^+} \rangle)}{\exp(\langle \mathbf{e}_{q_i}, \mathbf{e}_{p_i^+} \rangle) + \sum_{j=1}^N \exp(\langle \mathbf{e}_{q_i}, \mathbf{e}_{p_{i,j}^-} \rangle)}.$$

During training, *in-batch negatives* are used, where each positive and negative passage in the batch serves as a negative for all other queries.

Inference. DPR constructs a passage index by encoding all target passages. At runtime, it retrieves the top-ranked passages using maximum inner product search, with the query embedding as input.

B Notes on Computational Overhead

In this section, we analyze the impact of KPR's attention layer on the total number of floating point operations (FLOPs), following the estimation method of Kaplan et al. (2020). The non-embedding FLOPs for a single forward pass of BERT can be approximated as:

$$\text{FLOPs}_{\text{BERT}} \approx 2LDM(12D + M),$$

where L, D, and M denote the number of hidden layers, hidden size, and input token length, respectively.

The FLOPs for a forward pass of KPR's attention layer can be approximated as:

$$\text{FLOPs}_{\text{KPR-att}} \approx 2D^2(2N + 1) + 2DN,$$

where N denotes the number of input entities. The two terms correspond to the computation of the key-query-value matrices and the dot product, respectively.

The total FLOPs for a single forward pass of KPR is then given by:

$$\text{FLOPs}_{\text{KPR}} \approx \text{FLOPs}_{\text{BERT}} + \text{FLOPs}_{\text{KPR-att}}.$$

For example, with $M = 128$ input tokens and $N = 16$ entities, a forward pass of base-sized BERT ($L = 12$, $D = 768$) requires approximately 22 GFLOPs. The additional cost from KPR's attention layer is about 39 MFLOPs, accounting for only 0.18% of the total FLOPs.

Note that while FLOPs provide a hardware-agnostic estimate of computational cost, actual runtime may vary due to factors such as implementation optimizations and hardware constraints. Consequently, the computational latency of KPR's attention layer may not precisely correspond to its 0.18% FLOPs share.

C Detailed Experimental Setup

Entity vocabulary. The entity vocabulary of KPR consists of 7.2M English Wikipedia entities that appear as hyperlinks at least once in the April 2024 dump. These hyperlinks are extracted using the `mwparserfromhell` library.[1]

Entity embeddings. As described in Section 2, entity embeddings are obtained by running BERT inference on Wikipedia passages that refer to the corresponding entities. For each entity, we randomly select up to 128 such passages and compute the embedding using the method outlined in Section 2.

For the Wikipedia2Vec entity embeddings used in Section 3.1, we train the model with the default hyperparameters, except that the embedding dimension is set to 768 to match BERT's input embeddings. The same Wikipedia dump is used as for our entity embeddings.

Entity linker. The dictionary used in our entity linker is constructed directly from entity hyperlinks in Wikipedia. For example, if a hyperlink with the anchor text "New York" refers to the entity *New York City*, we register "New York" as an entity name and *New York City* as its possible referent.

To build the dictionary, we collect two statistics commonly used in the entity linking literature: *link probability*, the probability that a name appears as a hyperlink in Wikipedia, and *commonness*, the probability that a name refers to a specific entity

[1] https://github.com/earwig/mwparserfromhell

Dataset	License	#Train	#Dev	#Test
Natural Questions	Apache-2.0	58,880	8,757	3,610
TriviaQA	Apache-2.0	60,413	–	11,313
WebQuestions	CC BY 4.0	2,474	–	–
CuratedTREC	–	1,125	–	–
EntityQuestions	MIT	–	–	22,075
MS MARCO	Non-commercial	502,939	6,980	–
GraphQuestions	CC BY 4.0	1,672	417	2,075

Table 4: Licenses and the number of training, development, and test examples (if available and used) for each dataset used in this paper.

Model	Base Model	Entity Linker	EQ	NQ	TQA	Avg
BM25	–	–	79.8	73.7	76.7	76.7
DPR_{BERT}	BERT	–	70.0	86.0	84.7	80.2
KPR_{BERT}	BERT	Dictionary	78.9	87.0	85.0	83.6
DPR_{PELT}	BERT	ReFinED	75.3	86.3	84.7	82.1
KPR_{BERT}	BERT	ReFinED	78.7	86.7	85.0	83.5
$\text{DPR}_{\text{RetroMAE}}$	RetroMAE	–	79.8	87.7	86.0	84.5
$\text{KPR}_{\text{RetroMAE}}$	RetroMAE	Dictionary	**82.3**	**88.2**	**86.7**	**85.7**

Table 5: Top 100 accuracies of KPR and baseline models. For clarity, models are grouped by their base pretrained models and entity linkers. The best overall scores are shown in bold, and the best scores within each group are underlined. The corresponding top 20 accuracies are available in Table 1.

(Mihalcea and Csomai, 2007; Milne and Witten, 2008). To filter out names unlikely to denote entities (e.g., function words), we exclude a name if its link probability is below 5%. We also exclude an entity if its commonness with respect to the name is below 30%.

We tokenize text using the default English tokenizer provided by the `SpaCy` library (Honnibal et al., 2020) and extract entity names by matching all possible n-grams against the dictionary. Dictionary matching is implemented using a trie from the `marisa-trie` library[2] and is performed in a case-insensitive manner.

In Section 3, we also use the ReFinED entity linker, employing the model trained on Wikipedia with default parameters.

Model. We use the base-sized BERT[3] with 110M parameters. RetroMAE[4] and bge-base[5] are also based on this BERT model. KPR with the entity embeddings contains 5.6B parameters.

Datasets. The licenses and the number of training, development, and test examples for the datasets used in this paper are provided in Table 4. All datasets are publicly and freely available for research purposes. Access to CuratedTREC is governed by the TREC organizers. MS MARCO is freely available for non-commercial research purposes.[6] We also use Wikipedia, which is licensed under CC BY-SA 4.0.

For the target passages used in retrieval, we use a preprocessed version of English Wikipedia containing 21M passages, originally released by Karpukhin et al. (2020). The corpus is based on the December 2018 Wikipedia dump, with semi-structured content such as tables, infoboxes, lists, and disambiguation pages removed. The remaining article text is segmented into non-overlapping chunks, each containing approximately 100 words.

Training. Our training settings follow Karpukhin et al. (2020). For each query, we use one positive and one hard negative passage, and construct a mini-batch of 128 queries. We share parameters between the query and passage encoders to reduce computational cost, as using separate en-

[2] https://github.com/pytries/marisa-trie

[3] https://huggingface.co/google-bert/bert-base-uncased

[4] https://huggingface.co/Shitao/RetroMAE

[5] https://huggingface.co/BAAI/bge-base-en

[6] https://microsoft.github.io/msmarco/

Model	EQ	NQ	TQA	Avg
contriever (Izacard et al., 2022)	75.2	80.6	82.9	80.0
gte-base (Li et al., 2023)	79.8	86.4	83.9	83.4
e5-base (Wang et al., 2022)	82.5	**90.8**	86.2	86.5
bge-base (Xiao et al., 2024)	80.8	88.5	85.9	85.1
$\text{KPR}_{\text{bge-base}}$	**84.2**	89.0	**86.6**	**86.6**

Table 6: Top 100 accuracies of KPR based on bge-base and off-the-shelf retrievers on EQ, NQ, and TQA. The corresponding top 20 accuracies are available in Table 3.

Model	MRR@10	MRR@100	R@10	R@100	R@1000
DPR_{BERT} (Xiao et al., 2022)	31.7	32.8	58.0	85.7	96.0
KPR_{BERT}	**33.1**	**34.2**	**60.1**	**86.5**	**96.1**
$\text{DPR}_{\text{RetroMAE}}$ (Xiao et al., 2022)	35.5	36.7	63.6	89.2	97.6
$\text{KPR}_{\text{RetroMAE}}$	**37.3**	**38.4**	**66.4**	**91.2**	**98.3**

Table 7: The experimental results of KPR and DPR on the MS MARCO dataset.

coders yields nearly identical results (within 0.2% difference in top 20 accuracy across datasets) in our experiments, consistent with prior work (Lee et al., 2023; Zhang et al., 2023a). We optimize the model using Adam (Kingma and Ba, 2014) with a learning rate of 2e-5 for 40 epochs. We apply in-batch negatives (see Appendix A) during training. We do not perform hyperparameter tuning and instead use the same settings provided in the GitHub repository of Karpukhin et al. (2020).[7]

To construct KPR with the bge-base model presented in Section 3.2, we make slight modifications to align with the original settings of bge-base. Specifically, we use cosine similarity instead of the dot product for the similarity function and set the temperature of the softmax function to 0.02 before computing the cross-entropy loss during training. We also prepend the instruction "Represent this sentence for searching relevant passages:" to each query. We tune the hyperparameters over 12 trials using the development set of NQ. In particular, we select the number of epochs from $\{10, 20, 40\}$ and the learning rate from $\{2\text{e}{-5}, 3\text{e}{-5}, 5\text{e}{-5}, 1\text{e}{-4}\}$, and choose 20 epochs and a learning rate of $5\text{e}{-5}$. Note that training is highly stable, and all hyperparameter settings yield similar results.

Our training is implemented using the `PyTorch` library (Ansel et al., 2024), the `Transformers` library (Wolf et al., 2020), the `Datasets` library (Lhoest et al., 2021), and the `DeepSpeed` library (Rasley et al., 2020). Experiments are conducted on servers equipped with two Intel Xeon E5-2698 v4 CPUs and eight NVIDIA V100 GPUs, each with 32GB of memory. Training KPR takes approximately eight hours.

D Results Based on Top 100 Accuracy

Table 5 presents the top 100 accuracies of KPR and baseline models. Furthermore, Table 6 reports the top 100 accuracies of KPR based on bge-base and recent off-the-shelf retrievers.

E Additional Experiments on MS MARCO

We conduct additional experiments on the MS MARCO dataset to assess the effectiveness of KPR on a widely used retrieval benchmark. The experimental setup follows Xiao et al. (2022). We train the model on the public training set, using hard negatives mined with BM25, and employ the official corpus of 8.8 million passages as the retrieval target. Evaluation is conducted on the development set. We report Mean Reciprocal Rank (MRR@10 and MRR@100) and recall (R@10, R@100, and R@1000).

We use the hyperparameters provided in the GitHub repository of Xiao et al. (2022).[8] For each query, we select one positive and 15 negative passages to form a mini-batch of 16 queries. Negative passages are sampled from the top 200 ranked by BM25. We apply in-batch negatives during training. The model is trained for 4 epochs with a learn-

[7] https://github.com/facebookresearch/DPR/blob/main/conf/train/biencoder_nq.yaml

[8] https://github.com/staoxiao/RetroMAE/tree/master/examples/retriever/msmarco

ing rate of 2e-5 using the Adam optimizer. For RetroMAE, we use the variant trained on the MS MARCO corpus[9], following Xiao et al. (2022).

As shown in Table 7, KPR consistently outperforms DPR built on both BERT and RetroMAE across all metrics. These results clearly demonstrate the effectiveness of KPR on this benchmark.

F Qualitative Analysis of KPR's Attention Mechanism

To examine the behavior of KPR's attention mechanism, we conduct an experiment using the GraphQuestions dataset (Su et al., 2016; Sorokin and Gurevych, 2018), which contains questions annotated with entity names linked to Wikidata entities that can be resolved to Wikipedia entities. We randomly select 500 questions that include entity names associated with more than two possible referent entities in the dictionary used by our entity linker. Each question and its entities are input to KPR, and we manually inspect the attention weights (i.e., the normalized outputs of the sigmoid activation) assigned to each entity.

We present 10 example questions with their entities and corresponding attention weights in Figure 3. This analysis reveals two general trends: (1) KPR tends to assign lower attention weights to common entities, such as widely known geographic names, racial group names, and religious terms (Questions #1 and #2). We hypothesize that this occurs because such entities are already well-represented in the underlying BERT model, and KPR does not require additional knowledge to handle them. (2) KPR generally assigns higher attention weights to correct entities than to incorrect ones (Questions #1–#8), suggesting that it implicitly learns to disambiguate entities. However, it occasionally assigns higher or comparable weights to incorrect entities, particularly when those entities are rarer than the correct ones (Questions #8–#10). Moreover, in some cases where the correct and incorrect entities are semantically similar (e.g., Questions #1 and #2), both may still contribute positively to retrieval performance.

These findings partly explain why KPR appears robust to noise (see Section 3.1), even though it sometimes assigns non-negligible weights to irrelevant entities. Another possible explanation is that, as suggested by a recent study (Elhage et al., 2022), LMs can represent numerous features simultaneously in an almost orthogonal manner, which may allow the incorporation of noisy entities without significantly affecting similarity scores.

[9] https://huggingface.co/Shitao/RetroMAE_MSMARCO

Question #1: in which month does the average rainfall of *new york* exceed 86 mm?
✔ New York City (**0.031**)
✖ New York (state) (0.030)

Question #2: what is the number of tv directors that are *jewish*?
✔ Jews (**0.041**)
✖ Judaism (0.030)

Question #3: how does one drink *margarita*?
✔ Margarita (**0.099**)
✖ Margarita Island (0.061)

Question #4: the largest unit of area in the *si* system is called as?
✔ International System of Units (**0.098**)
✖ Silicon (0.077)

Question #5: *thatcher* was the inspiration for which for your eyes only character?
✔ Margaret Thatcher (**0.127**)
✖ Thatcher, Arizona (0.071)

Question #6: what movies were edited by *spielberg*?
✔ Steven Spielberg (**0.161**)
✖ Spielberg, Styria (0.091)

Question #7: who made the sensor of *d300*?
✔ Nikon D300 (**0.121**)
✖ State road D.300 (Turkey) (0.088)

Question #8: which position was *glen johnson* playing in 2010 world cup?
✔ Glen Johnson (**0.122**)
✖ Glen Johnson (boxer) (0.120)

Question #9: *michael tyson* uses which stance?
✔ Mike Tyson (0.107)
✖ Michael Tyson (antiquary) (**0.110**)

Question #10: *o* was discovered by how many people?
✔ Oxygen (0.099)
✖ Big O notation (**0.142**)

Figure 3: Qualitative analysis of KPR's attention mechanism on the GraphQuestions dataset. Entity names are underlined; correct and incorrect referent entities are marked with ✔and ✖, respectively. The referent entity receiving the higher attention weight is shown in bold.