BERT-kNN: Adding a kNN Search Component to Pretrained Language Models for Better QA

Nora Kassner, Hinrich Schütze
Center for Information and Language Processing (CIS)
LMU Munich, Germany
`kassner@cis.lmu.de`

Abstract

Khandelwal et al. (2020) use a k-nearest-neighbor (kNN) component to improve language model performance. We show that this idea is beneficial for open-domain question answering (QA). To improve the recall of facts encountered during training, we combine BERT (Devlin et al., 2019) with a traditional information retrieval step (IR) and a kNN search over a large datastore of an embedded text collection. Our contributions are as follows: i) BERT-kNN outperforms BERT on cloze-style QA by large margins without any further training. ii) We show that BERT often identifies the correct response category (e.g., US city), but only kNN recovers the factually correct answer (e.g., "Miami"). iii) Compared to BERT, BERT-kNN excels for rare facts. iv) BERT-kNN can easily handle facts not covered by BERT's training set, e.g., recent events.

1 Introduction

Pretrained language models (PLMs) like BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019) and RoBERTa (Liu et al., 2019) have emerged as universal tools that not only capture a diverse range of linguistic, but also (as recent evidence seems to suggest) factual knowledge.

Petroni et al. (2019) introduced LAMA (LAnguage Model Analysis) to test BERT's performance on open-domain QA and therefore investigate PLMs' capacity to recall factual knowledge without the use of finetuning. Since the PLM training objective is to predict masked tokens, question answering tasks can be reformulated as cloze questions; e.g., "Who wrote 'Ulysses'?" is reformulated as "[MASK] wrote 'Ulysses'." In this setup, Petroni et al. (2019) show that, on QA, PLMs outperform baselines trained on automatically extracted knowledge bases (KBs).

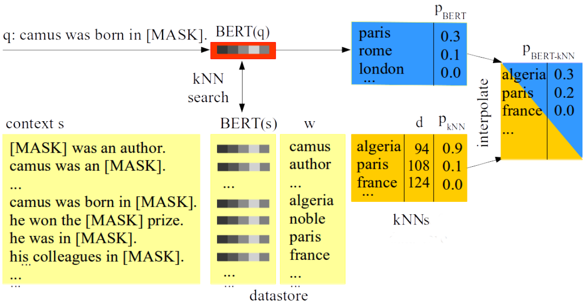

Figure 1: BERT-kNN interpolates BERT's prediction for question q with a kNN-search. The kNN search runs in BERT's embedding space, comparing the embedding of q with the embeddings of a retrieved subset of a large text collection: Pairs of a word w in the text collection and the BERT embedding of w's context ($BERT(s)$) are stored in a key-value datastore. An IR step is used to define a relevant subset of the full datastore (yellow). $BERT(q)$ (red) is BERT's embedding of the question. The kNN search runs between $BERT(q)$ and $BERT(s)$ and the corresponding distance d and word w is returned (orange). Finally, BERT's predictions (blue) are interpolated with this kNN search result.

Still, given that PLMs have seen more text than humans read in a lifetime, their performance on open-domain QA seems poor. Also, many LAMA facts that PLMs do get right are not "recalled" from training, but are guesses instead (Poerner et al., 2019). To address PLMs' poor performance on facts and choosing BERT as our PLM, we introduce BERT-kNN.

BERT-kNN combines BERT's predictions with a kNN search. The kNN search runs in BERT's embedding space, comparing the embedding of the question with the embeddings of a retrieved subset of a large text collection. The text collection can be BERT's training set or any other suitable text corpus. Due to its kNN component and its resulting ability to directly access facts stated in the searched text, BERT-kNN outperforms BERT on cloze-style

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3424–3430
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

Dataset	BERT-base	BERT-large	ERNIE	Know-BERT	E-BERT	BERT-kNN
LAMA	27.7	30.6	30.4	31.7	36.2	**39.4**
LAMA-UHN	20.6	23.0	24.7	24.6	31.1	**34.8**

Table 1: Mean P@1 on LAMA and LAMA-UHN on the TREx and GoogleRE subsets for BERT-base, BERT-large, ERNIE (Zhang et al., 2019), KnowBert (Peters et al., 2019), E-BERT (Poerner et al., 2019) and BERT-kNN. BERT-kNN performs best.

QA by large margins.

A schematic depiction of the model is shown in Figure 1. Specifically, we use BERT to embed each token's masked context s in the text collection ($BERT(s)$). Each pair of context embedding and token is stored as a key-value pair in a datastore. Testing for a cloze question q, the embedding of q ($BERT(q)$) serves as query to find the k context-target pairs in the subset of the datastore that are closest. The final prediction is an interpolation of the kNN search and the PLM predictions.

We find that the kNN search over the full datstore alone does not obtain good results. Therefore, we first query a separate information retrieval (IR) index with the original question q and only search over the most relevant subset of the full datastore when finding the k-nearest-neighbors of $BERT(q)$ in embedding space.

We find that the PLM often correctly predicts the answer category and therefore the correct answer is often among the top k-nearest-neighbors. A typical example is "Albert Einstein was born in [MASK]": the PLM knows that a city is likely to follow and maybe even that it is a German city, but it fails to pick the correct city. On the other hand, the top-ranked answer in the kNN search is "Ulm" and so the correct filler for the mask can be identified.

BERT-kNN sets a new state-of-the-art on the LAMA cloze-style QA dataset without any further training. Even though BERT-kNN is based on BERT-base, it also outperforms BERT-large. The performance gap between BERT and BERT-kNN is most pronounced on hard-to-guess facts. Our method can also make recent events available to BERT without any need of retraining: we can simply add embedded text collections covering recent events to BERT-kNN's datastore.

The source code of our experiments is available under: https://github.com/norakassner/BERT-kNN.

2 Data

The LAMA dataset is a cloze-style QA dataset that allows to query PLMs for facts in a way analogous to KB queries. A cloze question is generated using a subject-relation-object triple from a KB and a templatic statement for the relation that contains variables X and Y for subject and object; e.g, "X was born in Y". The subject is substituted for X and [MASK] for Y. In all LAMA triples, Y is a single-token answer.

LAMA covers different sources: The GoogleRE[1] set covers the relations "place of birth", "date of birth" and "place of death". TREx (ElSahar et al., 2018) consists of a subset of Wikidata triples covering 41 relations. ConceptNet (Li et al., 2016) combines 16 commonsense relations among words and phrases. The underlying Open Mind Common Sense corpus provides matching statements to query the language model. SQuAD (Rajpurkar et al., 2016) is a standard question answering dataset. LAMA contains a subset of 305 context-insensitive questions. Unlike KB queries, SQuAD uses manually reformulated cloze-style questions which are not based on a template.

We use SQuAD and an additional 305 ConceptNet queries for hyperparamter search.

Poerner et al. (2019) introduce LAMA-UHN, a subset of LAMA's TREx and GoogleRE questions from which easy-to-guess facts have been removed.

To test BERT-kNN's performance on unseen facts, we collect Wikidata triples containing TREx relations from Wikipedia pages created January–May 2020 and add them to the datastore.

3 Method

BERT-kNN combines BERT with a kNN search component. Our method is generally applicable to PLMs. Here, we use BERT-base-uncased (Devlin et al., 2019). BERT is pretrained on the BookCorpus (Zhu et al., 2015) and the English Wikipedia.

Datastore. Our text collection C is the 2016-12-21 English Wikipedia.[2] For each single-token word occurrence w in a sentence in C, we com-

[1] https://code.google.com/archive/p/relation-extraction-corpus/

[2] dumps.wikimedia.org/enwiki

Dataset	Statistics		Model		
	Facts	Rel	BERT	kNN	BERT -kNN
GoogleRE	5527	3	9.8	**51.1**	48.6
TREx	34039	42	29.1	34.4	**38.7**
ConceptNet	11153	16	**15.6**	4.7	11.6
SQuAD	305	-	14.1	**25.5**	24.9
unseen	34637	32	18.8	21.5	**27.1**

Table 2: Mean P@1 for BERT-base, kNN and their interpolation (BERT-kNN) for LAMA subsets and unseen facts. BERT results differ from Petroni et al. (2019) where a smaller vocabulary is used.

Configuration	P@1
hidden layer 12	36.8
hidden layer 11	**39.4**
hidden layer 10	34.7
hidden layer 11 (without IR)	26.9

Table 3: Mean P@1 on LAMA (TREx, GoogleRE subsets) for different context embedding strategies. Top: The context embedding is represented by the embedding of the masked token in different hidden layers. Best performance is obtained using BERT's hidden layer 11. Bottom: We show that BERT-kNN's performance without the additional IR step drops significantly. We therefore conclude that the IR step is an essential part of BERT-kNN.

pute the pair (c, w) where c is a context embedding computed by BERT. To be specific, we mask the occurrence of w in the sentence and use the embedding of the masked token. We store all pairs (c, w) in a key-value datastore D where c serves as key and w as value.

Information Retrieval. We find that just using the datastore D does not give good results (see result section). We therefore use Chen et al. (2017)'s IR system to first select a small subset of D using a keyword search. The IR index contains all Wikipedia articles. An article is represented as a bag of words and word bigrams. We find the top 3 relevant Wikipedia articles using TF-IDF search. For KB queries, we use the subject to query the IR index. If the subject has its dedicated Wikipedia page, we simply use this. For non-knowledge base queries, we use the cloze-style question q ([MASK] is removed).

Inference. During testing, we first run the IR search to identify the subset D' of D that corresponds to the relevant Wikipedia articles. For the kNN search, q is embedded in the same way as the context representations c in D: we set $BERT(q)$ to the embedding computed by BERT for [MASK]. We then retrieve the $k = 128$ nearest-neighbors of $BERT(q)$ in D'. We convert the distances (Euclidean) between $BERT(q)$ and the kNNs to a probability distribution using softmax. Since a word w can occur several times in kNN, we compute its final output probability as the sum over all occurrences.

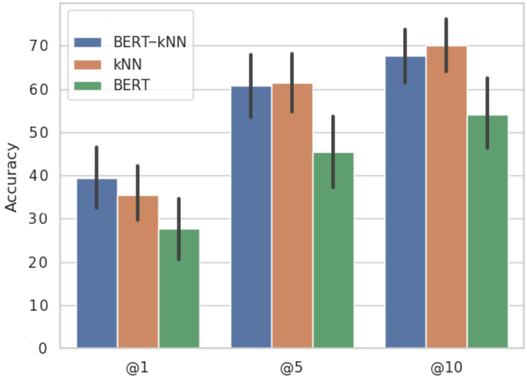

Figure 2: Mean P@1, P@5, P@10 on LAMA for original BERT and BERT-kNN.

In the final step, we interpolate kNN's (weight 0.3) and BERT's original predictions (weight 0.7). We optimize hyperparameters on dev. See supplementary for details.

Evaluation. Following Petroni et al. (2019) we report mean precision at rank r (P@r). P@r is 1 if the top r predictions contain the correct answer, otherwise it returns 0. To compute mean precision, we first average within each relation and then across relations.

4 Results and Discussion

Table 1 shows that BERT-kNN outperforms BERT on LAMA. It has about 10 precision point gain over BERT, base and large. Recall that BERT-kNN uses BERT-base. The performance gap between original BERT and BERT-kNN becomes even larger when evaluating on LAMA-UHN, a subset of LAMA with hard-to-guess facts.

It also outperforms entity-enhanced versions of BERT (see related work) – ERNIE (Zhang et al., 2019), KnowBert (Peters et al., 2019) and E-BERT (Poerner et al., 2019) – on LAMA.

Table 2 shows that BERT-kNN outperforms BERT on 3 out of 4 LAMA subsets. BERT prevails on ConceptNet; see discussion below. Huge gains are obtained on the GoogleRE dataset. Figure 2 shows precision at 1, 5 and 10. BERT-kNN performs better than BERT in all three categories.

Table 3 compares different context embedding strategies. BERT's masked token embedding of

	Query and True Answer	Generation
Google RE	hans gefors was born in [MASK]. True: stockholm	BERT-kNN: stockholm (0.36), oslo (0.15), copenhagen (0.13) BERT: oslo (0.22), copenhagen (0.18), bergen (0.09) kNN: stockholm (1.0), lund (0.00), hans (0.00)
TREx	regiomontanus works in the field of [MASK]. True: mathematics	BERT-kNN: astronomy (0.20), mathematics (0.13), medicine (0.06) BERT: medicine (0.09), law (0.05), physics (0.03) kNN: astronomy (0.63), mathematics (0.36), astronomical (0.00)
Concept Net	ears can [MASK] sound. True: hear	BERT-kNN: hear (0.27), detect (0.23), produce (0.06) BERT: hear (0.28), detect (0.06), produce (0.04) kNN: detect (0.77), hear (0.14), produce (0.10)
Squad	tesla was in favour of the [MASK] current type. True: ac	BERT-kNN: alternating (0.39), electric (0.18), direct (0.11) BERT: electric (0.28), alternating (0.18), direct (0.11) kNN: alternating (0.87), direct (0.12), ac (0.00)

Table 4: Examples of generation for BERT-base, kNN, BERT-kNN. The last column reports the top three tokens generated together with the associated probability (in parentheses).

hidden layer 11 performs best. We also show the necessity of the IR step by running a kNN search over all Wikipedia contexts, which results in precision lower than original BERT. To run an efficient kNN search over all contexts instead of the relevant subset identified by the IR step, we use the FAISS libary (Johnson et al., 2017).

Table 2 also shows that neither BERT nor kNN alone are sufficient for top performance, while the interpolation of the two yields optimal results. In many cases, BERT and kNN are complementary. kNN is worse than BERT on ConceptNet, presumably because commonsense knowledge like “birds can fly” is less well-represented in Wikipedia than entity triples and also because relevant articles are harder to find by IR search. We keep the interpolation parameter constant over all datasets. Table 4 shows that kNN often has high confidence for correct answers – in such cases it is likely to dominate less confident predictions by BERT. The converse is also true (not shown). Further optimization could be obtained by tuning interpolation per dataset.

BERT-kNN answers facts unseen during pretraining better than BERT, see Table 2. BERT was not trained on 2020 events, so it must resort to guessing. Generally, we see that BERT’s knowledge is mainly based on guessing as it has seen Wikipedia during training but is not able to recall the knowledge recovered by kNN.

Table 4 gives examples for BERT and BERT-kNN predictions. We see that BERT predicts the answer category correctly, but it often needs help from kNN to recover the correct entity within that category.

5 Related work

PLMs are top performers for many tasks, including QA (Kwiatkowski et al., 2019; Alberti et al., 2019; Bosselut et al., 2019). Petroni et al. (2019) introduced the LAMA QA task to probe PLMs’ knowledge of facts typically modeled by KBs.

The basic idea of BERT-kNN is similar to Khandelwal et al. (2020)’s interpolation of a PLM and kNN for language modeling. In contrast, we address QA. We introduce an IR step into the model that is essential for good performance. Also, our context representations differ as we use embeddings of the masked token.

Grave et al. (2016) and Merity et al. (2017), inter alia, also make use of memory to store hidden states. They focus on recent history, making it easier to copy rare vocabulary items.

DRQA (Chen et al., 2017) is an open-domain QA model that combines an IR step with a neural reading comprehension model. We use the same IR module, but our model differs significantly. DRQA does not predict masked tokens, but extracts answers from text. It does not use PLMs nor a kNN module. Most importantly, BERT-kNN is fully unsupervised and does not require any extra training.

Some work on knowledge in PLMs focuses on injecting knowledge into BERT’s encoder. ERNIE (Zhang et al., 2019) and KnowBert (Peters et al., 2019) are entity-enhanced versions of BERT. They introduce additional encoder layers that are integrated into BERT’s original encoder by expensive additional pretraining. Poerner et al. (2019) injects factual entity knowledge into BERT’s embeddings without pretraining by aligning Wikipedia2Vec entity vectors (Yamada et al., 2016) with BERT’s wordpiece vocabulary. This approach is also limited to labeled entities. Our approach on the other hand is not limited to labeled entities nor does it require any pretraining. Our approach is conceptually different from entity-enhanced versions of BERT and could potentially be combined with them for even better performance. Also, these models address language modeling, not QA.

The combination of PLMs with an IR step/kNN search has attracted a lot of recent research interest. The following paragraph lists concurrent work:

Petroni et al. (2020) also combine BERT with an IR step to improve cloze-style QA. They do not use a kNN search nor an interpolation step but feed the retrieved contexts into BERT's encoder. Guu et al. (2020) augment PLMs with a latent knowledge retriever. In contrast to our work they continue the pretraining stage. They jointly optimize the masked language modeling objective and backpropagate through the retrieval step. Lewis et al. (2020); Izacard and Grave (2020) leverage retrieved contexts for better QA using finetuned generative models. They differ in that the latter fuse evidence of multiple contexts in the decoder. Joshi et al. (2020) integrate retrieved contexts into PMLs for better reading comprehension.

6 Conclusion

This work introduced BERT-kNN, an interpolation of BERT predictions with a kNN search for unsupervised cloze-style QA. BERT-kNN sets a state-of-the-art on LAMA without any further training. BERT-kNN can be easily enhanced with knowledge about new events that are not covered in the training text used for pretraining BERT.

In future work, we want to exploit the utility of the kNN component for explainability: kNN predictions are based on retrieved contexts, which can be shown to users to justify an answer.

Acknowledgements

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibility for its content.

References

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A BERT baseline for the natural questions. *ArXiv*, abs/1901.08634.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hady ElSahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon S. Hare, Frédérique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018.*

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2016. Improving neural language models with a continuous cache. *ICLR*, abs/1612.04426.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Mandar Joshi, Kenton Lee, Yi Luan, and Kristina Toutanova. 2020. Contextualized representations using textual encyclopedic knowledge. *arXiv preprint arXiv:2004.12006*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*.

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*, 7:453–466.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kttler, Mike Lewis, Wen tau Yih, Tim Rocktschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2019. Bert is not a knowledge base (yet): Factual knowledge vs. name-based reasoning in unsupervised qa. *ArXiv*, abs/1911.03681.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, Berlin, Germany. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A Data

LAMA and LAMA-UHN can be downloaded from: https://dl.fbaipublicfiles.com/LAMA/

For TREx unseen, we downloaded the latest Wikidata and Wikipedia dump from: https://dumps.wikimedia.org/wikidatawiki/entities/wikipedia_en/latest-all.json.bz2
and
https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2.

We filter for TREx relations and only consider facts which have a Wikipedia page created after January 1st 2020. We only consider relations with 5 questions or more. We add the additional embedded Wikipedia articles to the datastore.

B Inference

The probability of the kNN search for word w is given by:
$p_{kNN}(w \mid q) \sim \sum_{(c_w,w)\in kNN} e^{-d(BERT(q),c_w)/l}$.

The final probability of BERT-kNN is the interpolation of the predictions of BERT and the kNN search:
$p_{BERT-kNN}(q) = \lambda p_{kNN}(q) + (1-\lambda) p_{BERT}(q)$,

with
q question,
$BERT(q)$ embedding q,
w target word,
s_w context of w,
$c_w = BERT(s)$ embedded context,
d distance,
l distance scaling,
λ interpolation parameter.

C Hyperparameters

Hyperparameter optimization is done with the 305 SQuAD questions and additional randomly sampled 305 ConceptNet questions. We remove the 305 ConceptNet questions from the test set.
We run the hyperparameter search once.
We run a grid search for the following hyperparameters:
Number of documents N = [1, 2, 3, 4, 5],
Interpolation λ = [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8],
Number of NN k = [64, 128, 512],
Distance scaling l = [5, 6, 7, 8, 9, 10, 11, 12].

The optimal P@1 was found for:
Number of documents N = 3,
Interpolation parameter λ = 0.3,
Number of NN k= 128,
Distance scaling l = 6.

D kNN without IR

To enable a kNN search over the full datastore we use FAISS index (Johnson et al., 2017). We train the index using 1M randomly sampled keys and 40960 number of clusters. Embeddings are quantized to 64 bytes. During inference the index looks up 64 clusters.

E Computational Infrastructure

The creation of the datastore is computationally expensive but only a single forward pass is needed. The datastore creation is run on a server with 128 GB memory, Intel(R) Xeon(R) CPU E5-2630 v4, CPU rate 2.2GHz, number of cores 40(20), 8x GeForce GTX 1080Ti. One GPU embedds 300 contexts/s. The datastore includes 900M contexts.

Evaluation is run on a server with 128 GB memory, Intel(R) Xeon(R) CPU E5-2630 v4, CPU rate 2.2GHz, number of cores 40(20). Evaluation time for one query is 2 s but code can be optimized for better performance.