Open Domain Question Answering over Tables via Dense Retrieval

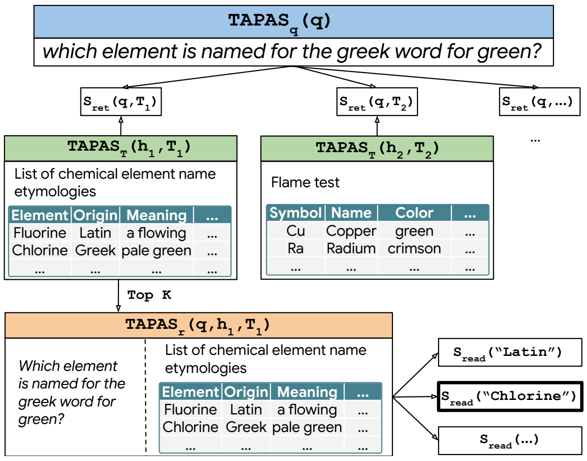
Jonathan Herzig*^,1, Thomas Müller^2, Syrine Krichene^2, Julian Martin Eisenschlos^2

[1]School of Computer Science, Tel-Aviv University
jonathan.herzig@cs.tau.ac.il

[2]Google Research
{thomasmueller,syrinekrichene,eisenjulian}@google.com

Abstract

Recent advances in open-domain QA have led to strong models based on dense retrieval, but only focused on retrieving textual passages. In this work, we tackle open-domain QA over tables for the first time, and show that retrieval can be improved by a retriever designed to handle tabular context. We present an effective pre-training procedure for our retriever and improve retrieval quality with mined hard negatives. As relevant datasets are missing, we extract a subset of NATURAL QUESTIONS (Kwiatkowski et al., 2019) into a Table QA dataset. We find that our retriever improves retrieval results from 72.0 to 81.1 recall@10 and end-to-end QA results from 33.8 to 37.7 exact match, over a BERT based retriever.


TAPAS_q (q)
which element is named for the greek word for green?
S_ret (q,T_1)
S_ret (q,T_2)
S_ret (q,...)
TAPAS_T (h_1,T_1)
List of chemical element name etymologies
TAPAS_T (h_2,T_2)
Flame test
...
Top K
TAPAS_r (q,h_1,T_1)
Which element is named for the greek word for green?
List of chemical element name etymologies
S_read ("Latin")
S_read ("Chlorine")
S_read (...)

Element	Origin	Meaning	...
Fluorine	Latin	a flowing	...
Chlorine	Greek	pale green	...
...	...	...	...

Symbol	Name	Color	...
Cu	Copper	green	...
Ra	Radium	crimson	...
...	...	...	...

Figure 1: An overview of our approach. A dense table retriever scores the question against all tables and outputs the top K tables ($K = 1$ in this example), and a reader selects the answer out of the top K tables.

1 Introduction

Models for question answering (QA) over tables usually assume that the relevant table is given during test time. This applies for semantic parsing (e.g., for models trained on SPIDER (Yu et al., 2018)) and for end-to-end QA (Neelakantan et al., 2016; Herzig et al., 2020). While this assumption simplifies the QA model, it is not realistic for many use-cases where the question is asked through some open-domain natural language interface, such as web search or a virtual assistant.

In these open-domain settings, the user has some information need, and the corresponding answer resides in some table in a large corpus of tables. The QA model then needs to utilize the corpus as an information source, efficiently search for the relevant table within, parse it, and extract the answer.

Recently, much work has explored open-domain QA over a corpus of textual passages (Chen et al., 2017; Sun et al., 2018; Yang et al., 2019; Lee et al., 2019, *inter alia*). These approaches usually follow a two-stage framework: (1) a retriever first selects a small subset of candidate passages relevant to the question, and then (2) a machine reader examines the retrieved passages and selects the correct answer. While these approaches work well on free text, it is not clear whether they can be directly applied to tables, as tables are semi-structured, and thus different than free text.

In this paper we describe the first study to tackle open-domain QA over tables, and focus on modifying the retriever. We follow the two-step approach of a retriever model that retrieves a small set of candidate tables from a corpus, followed by a QA model (Figure 1). Specifically, we utilize dense retrieval approaches targeted for retrieving passages (Lee et al., 2019; Guu et al., 2020; Karpukhin et al., 2020), and modify the retriever to better handle tabular contexts. We present a simple and effective pre-training procedure for our retriever, and further improve its performance by mining hard negatives using the retriever model. Finally, as relevant open domain datasets are missing, we process NATURAL QUESTIONS (Kwiatkowski et al., 2019) and extract 11K examples where the answer resides in some table. Our model and data generation code

*Work completed while interning at Google.

as well as the pre-trained model are publicly available at `https://github.com/google-research/tapas`.

2 Setup

We formally define open domain extractive QA over tables as follows. We are given a training set of N examples $\mathcal{D}_{\text{train}} = \{(q_i, T_i, a_i)\}_{i=1}^{N}$, where q_i is a question, T_i is a table where the answer a_i resides, and a corpus of M tables $\mathcal{C} = \{T_i\}_{i=1}^{M}$. The answer a_i is comprised of one or more spans of tokens in T_i. Our goal is to learn a model that given a new question q and the corpus $\mathcal{C}$ returns the correct answer a.

Our task shares similarities with open domain QA over documents (Chen et al., 2017; Yang et al., 2019; Lee et al., 2019), where the corpus $\mathcal{C}$ consists of textual passages extracted from documents instead of tables, and the answer is a span that appears in some passage in the corpus. As in these works, dealing with a large corpus (of tables in our setting), requires relevant context retrieval. Naively applying a QA model, for example TAPAS (Herzig et al., 2020), over each table in the large corpus is not practical because inference is too expensive.

To this end we break our system into two independent steps. First, an efficient table retriever component selects a small set of candidate tables $\mathcal{C}_R$ from a large corpus of tables $\mathcal{C}$. Second, we apply a QA model to extract the answer a given the question q and the candidate tables $\mathcal{C}_R$.

3 Dense Table Retrieval

In this section we describe our dense table retriever (DTR), which retrieves a small set of K candidate tables $\mathcal{C}_R$ given a question q and a corpus $\mathcal{C}$. In this work we set $K = 10$ and take $\mathcal{C}$ to be the set of all tables in the dataset we experiment with (see §6).

As in recent work for open domain QA on passages (Lee et al., 2019; Guu et al., 2020; Karpukhin et al., 2020; Chen et al., 2021; Oguz et al., 2020), we also follow a dense retrieval architecture. As tables that contain the answer to q do not necessarily include tokens from q, a dense encoding can better capture similarities between table contents and a question.

For training DTR, we leverage both in-domain training data $\mathcal{D}_{\text{train}}$, and automatically constructed pre-training data $\mathcal{D}_{\text{pt}}$ of text-table pairs (see below).

Retrieval Model In this work we focus on learning a retriever that can represent tables in a meaningful way, by capturing their specific structure. Traditional information retrieval methods such as BM25 are targeted to capture token overlaps between a query and a textual document, and other dense encoders are pre-trained language models (such as BERT) targeted for text representations.

Recently, Herzig et al. (2020) proposed TAPAS, an encoder based on BERT, designed to contextually represent text and a table jointly. TAPAS includes table specific embeddings that capture its structure, such as row and column ids. In DTR, we use TAPAS to represent both the query q and the table T. For efficient retrieval during inference we use two different TAPAS instances (for q and for T), and learn a similarity metric between them as Lee et al. (2019); Karpukhin et al. (2020).

More concretely, the TAPAS encoder $\texttt{TAPAS}(x_1, [x_2])$ takes one or two inputs as arguments, where x_1 is a string and x_2 is a flattened table. We then define the retrieval score as the inner product of dense vector representations of the question q and the table T:

$$\begin{aligned} h_q &= \mathbf{W_q}\texttt{TAPAS}_\texttt{q}(q)\texttt{[CLS]} \\ h_T &= \mathbf{W_T}\texttt{TAPAS}_\texttt{T}(\texttt{title}(T), T)\texttt{[CLS]} \\ \text{S}_{\text{ret}}(q, T) &= h_q^T h_T, \end{aligned}$$

where $\texttt{TAPAS}(\cdot)\texttt{[CLS]}$ returns the hidden state for the $\texttt{CLS}$ token, $\mathbf{W_q}$ and $\mathbf{W_T}$ are matrices that project the TAPAS output into $d = 256$ dimensional vectors, and $\texttt{title}(T)$ is the page title for table T. We found the table's page title to assist in retrieving relevant tables, which is also useful for Wikipedia passage retrieval (Lee et al., 2019).

Training The goal of the retriever is to create a vector space such that relevant pairs of questions and tables will have smaller distance (which results in a large dot product) than the irrelevant pairs, by learning an embedding. To increase the likelihood of gold (q, T) pairs, we train the retriever with in-batch negatives (Gillick et al., 2019; Henderson et al., 2017; Karpukhin et al., 2020). Let $\{(q_i, T_i)\}_{i=1}^{B}$ be a batch of B examples from $\mathcal{D}_{\text{train}}$, where for each q_i, T_i is the gold table to retrieve, and for each $j \neq i$ we treat T_j as a negative. We now define the likelihood of the gold table T_i as:

$$p(T_i|q_i) = \frac{\exp[\text{S}_{\text{ret}}(q_i, T_i)]}{\sum_{j=1}^{B} \exp[\text{S}_{\text{ret}}(q_i, T_j)]}.$$

To train the model efficiently, we define $\mathbf{Q}$ and $\mathbf{T}$ to be a $B \times d$ matrix that hold the representations for questions and tables respectively. Then, $\mathbf{S} = \mathbf{Q}\mathbf{T}^T$ gives an $B \times B$ matrix where the logits from the gold table are on the diagonal. We then train using a row-wise cross entropy loss where the labels are a $B \times B$ identity matrix.

Pre-training One could train our retriever from scratch, solely relying on a sufficiently large in-domain training dataset $\mathcal{D}_{\text{train}}$. However, we find performance to improve after using a simple pre-training method for our retriever. Lee et al. (2019) suggest to pre-train a textual dense retriever using an Inverse Cloze Task (ICT). In ICT, the goal is to predict a context given a sentence s. The context is a passage that originally contains s, but with s masked. The motivation is that the relevant context should be semantically similar to s, and should contain information missing from s.

Similarly, we posit that a table T that appears in close proximity to some text span s is more relevant to s than a random table. To construct a set $\mathcal{D}_{\text{pt}} = \{(s_i, T_i)\}_{i=1}^{M}$ that consists of M pre-training pairs (s, T), we use the pre-training data from Herzig et al. (2020). They extracted text-table pairs from 6.2M Wikipedia tables, where text spans were sampled from the table caption, page title, page description, segment title and text of the segment the table occurs in. This resulted in a total of 21.3M text-table (s, T) pairs. While Herzig et al. (2020) uses extracted (s, T) pairs for pre-training TAPAS with a masked language modeling objective, we pre-train DTR from these pairs, with the same objective used for in-domain data.

Hard Negatives Following similar work (Gillick et al., 2019; Karpukhin et al., 2020; Xiong et al., 2021), we use an initial retrieval model to extract the most similar tables from $\mathcal{C}$ for each question in the training set. From this list we discard each table that does contain the reference answer to remove false negatives. We use the highest scoring remaining table as a particular hard negative.

Given the new triplets of question, reference table and mined negative table, we train a new model using a modified version of the in-batch negative training discussed above. Given $\mathbf{Q}$ and $\mathbf{S}$ as defined above and a new matrix $\mathbf{N}$ $(B \times d)$ that holds the representations of the negative tables, $\mathbf{S}' = \mathbf{Q}\mathbf{N}^T$ gives another $B \times B$ matrix that we want to be small in value (possibly negative). If we concatenate $\mathbf{S}$ and $\mathbf{S}'$ row-wise we get a new matrix for which we can perform the same cross entropy training as before. The label matrix is now obtained by concatenating an identity matrix row-wise with a zero matrix.

Inference During inference time, we apply the table encoder $\texttt{TAPAS}_{\text{T}}$ to all the tables $T \in \mathcal{C}$ offline. Given a test question q, we derive its representation h_q and retrieve the top K tables with representations closest to h_q.

In our experiments, we use exhaustive search to find the top K tables, but to scale to large corpora, fast maximum inner product search using existing tools such as FAISS (Johnson et al., 2019) and SCANN (Guo et al., 2020) could be used, instead.

4 Question Answering over Tables

A reader model is used to extract the answer a given the question q and K candidate tables. The model scores each candidate and at the same time extracts a suitable answer span from the table. Each table and question are jointly encoded using a TAPAS model. The score is a simple logistic loss based on the CLS token, as in Eisenschlos et al. (2020).

The answer span extraction is modeled as a softmax over all possible spans up to a certain length. Spans that are located outside of a table cell or that cross a cell are masked. Following Lee et al. (2017, 2019), the span representation is the concatenation of the contextual representation of the first and last token in the span s:

$$h_{start} = \texttt{TAPAS}_r(q, \texttt{title}(T), T)[\texttt{START}(s)]$$
$$h_{end} = \texttt{TAPAS}_r(q, \texttt{title}(T), T)[\texttt{END}(s)]$$
$$\texttt{S}_{\text{read}}(q, T) = \texttt{MLP}([h_{start}, h_{end}]).$$

The training and test data are created by running a retrieval model. We extract the $K = 10$ highest scoring candidate tables for each question. At training time we add the reference table if it is missing from the candidates.

At inference time all table candidates are processed and the answer of the candidate with the highest score is returned as the predicted answer.

5 Dataset

We create a new English dataset called **NQ-TABLES** from NATURAL QUESTIONS (Kwiatkowski et al., 2019) (NQ). Concurrently with this work, Zayats et al. (2021) study a similar subset of NQ but without the retrieval aspect.

NQ was designed for question answering over Wikipedia articles. The $320K$ questions are mined

Model	R@1	R@10	R@50
BM25	16.77	40.06	58.39
DTR-Text	32.90	72.00	86.86
DTR-Schema	34.36	74.24	88.37
DTR	36.24	76.02	90.25
DTR +hnbm25	42.17	80.51	92.31
DTR +hn	**42.42**	**81.13**	**92.56**
DTR -pt	16.64	47.80	68.68

Table 1: Table retrieval results on NQ-TABLES test set. hn: hard negatives from DTR, hnbm25: hard negatives from BM25 baseline, pt: pre-training. DTR numbers are means over 5 random runs.

from real Google search queries and the answers are spans in Wikipedia articles identified by annotators. Although the answers for most questions appear in textual passages, we identified 12K examples where the answer resides in a table, and can be used as a QA over tables example. To this end, we form NQ-TABLES that consists of (q, T, a) triplets from these examples. Tables are extracted from the article's HTML, and are normalized by transposing *infobox* tables.

We randomly split the original NQ train set into train and dev (based on a hash of the page title) and use all questions from the original NQ dev set as our test set. To construct the corpus $\mathcal{C}$, we extract all tables that appear in articles in all NQ sets.

NQ can contain the same Wikipedia page in different versions which leads to many almost identical tables. We merge close duplicates using the following procedure. For all tables that occur on the same Wikipedia page we flatten the entire table content, tokenize it and compute l_2 normalized unigram vectors of the token counts of each table. We then compute the pair-wise cosine similarity of all tables. We iterate over the table pairs in decreasing order of similarity and attempt to merge them into clusters. This is essentially a version of single link clustering. In particular, we will merge two tables if the similarity is > 0.91, they do not occur on the same version of the page, their difference is rows is at most 2 and they have the same number of columns.

Dataset sizes are given in the following table:

train	dev	test	corpus $\mathcal{C}$
9,594	1,068	966	169,898

Retriever	Reader	EM	F1	Oracle EM	Oracle F1
BM25	TAPAS	21.46	28.24	29.51	40.79
DTR-Text	BERT	29.58	37.38	39.39	51.48
DTR-Text	TAPAS	33.78	43.49	42.83	56.46
DTR-Schema	TAPAS	32.75	42.19	42.63	55.05
DTR	TAPAS	35.50	45.44	46.09	59.01
DTR +hnbm25	TAPAS	36.61	46.74	47.46	60.72
DTR +hn	TAPAS	**37.69**	**47.70**	**48.20**	**61.50**

Table 2: QA results on NQ-TABLES test set. Numbers are means over 5 random runs.

6 Experiments

Details about the experimental setup are given Appendix A.

Retrieval Baselines We consider the following baselines as alternatives to DTR. We use the **BM25** (Robertson and Zaragoza, 2009) implementation of Gensim (Řehůřek and Sojka, 2010)[1]. To measure if a table-specific encoder is necessary, we implement **DTR-TEXT**, where the retriever is initialized from BERT (Devlin et al., 2019) instead of TAPAS. To test whether the content of the table is relevant, we experiment with **DTR-SCHEMA**, where only the headers and title are used to represent tables.

Retrieval Results Table 1 shows the test results for table retrieval (dev results are in Appendix B). We report recall at K (R@K) metrics as the fraction of questions for which the highest scoring K tables contain the reference table.

We find that all dense models that have been pretrained out-peform the BM25 baseline by a large margin. The model that uses the TAPAS table embeddings (DTR) out-performs the dense baselines by more than 1 point in R@10. The addition of mined negatives (DTR +hn) yields an additional improvement of more than 5 points. Mining negatives from DTR works better than mining negatives from BM25 (DTR +hnbm25, +0.6 R@10).

End-to-End QA Results for end-to-end QA experiments are shown in Table 2 (dev results are in Appendix B). We use the exact match (EM) and token F1 metrics as implemented in SQUAD (Rajpurkar et al., 2016).[2] We additionally report oracle

[1]We find that recall improves if the document title and table header tokens are counted multiple times. In all experiments we use a count of 15.

[2]`https://worksheets.codalab.org/rest/bundles/0x6b567e1cf2e041ec80d7098f031c5c9e/contents/blob/`

metrics which are computed on the best answer returned for any of the candidates.

We again find that all dense models out-perform the BM25 baseline. A TAPAS-based reader out-performs a BERT reader by more than 3 points in EM. The simple DTR model out-performs the baselines by more than 1 point in EM. Hard negatives from BM25 (+hnbm25) improve DTR's performance by 1 point, while hard negatives from DTR (+hn) improve performance by 2 points. We additionally perform a McNemar's significance test for our proposed model, DTR+hn, and find that it performs significantly better ($p<0.05$) than all baselines.

Analysis Analyzing the best model in Table 2 (DTR +hn) on the dev set, we find that 29% of the questions are answered correctly, 14% require a list answer (which is out of scope for this paper), 12% do not have any table candidate that contains the answer, for 11% the model does not select a table that contains the answer, and for 34% the reader fails to extract the correct span.

We further analyzed the last category by manually annotating 100 random examples. We find that for 23 examples the answer is partially correct (usually caused by inconsistent span annotations in NQ). For 11 examples the answer is ambiguous (e.g., the release date of a movie released in different regions). For 22 examples the table is missing context or does only contain the answer accidentally. Finally, 44 examples are wrong, usually because they require some kind of table reasoning, like computing the maximum over a column, or using common sense knowledge.

7 Conclusion

In this paper we demonstrated that a retriever designed to handle tabular context can outperform other textual retrievers for open-domain QA on tables. We additionally showed that our retriever can be effectively pre-trained and improved by hard negatives. In future work we aim to tackle multi-modal open-domain QA, combining passages and tables as context.

Acknowledgments

We would like to thank William Cohen, Sewon Min, Yasemin Altun and the anonymous reviewers for their constructive feedback, useful comments and suggestions. This work was completed in partial fulfillment for the PhD degree of the first author, which was also supported by a Google PhD fellowship.

References

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.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. 2021. Open question answering over tables and text. In *International Conference on Learning Representations*.

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.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, Hong Kong, China. Association for Computational Linguistics.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley, editors. 2017. *Google Vizier: A Service for Black-Box Optimization*.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896. PMLR.

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 Ma-*

chine Learning, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *ArXiv*, abs/1705.00652.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, pages 1–1.

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 (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

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

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2017. Learning recurrent span representations for extractive question answering. In *arXiv 1611.01436*.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unified open-domain question answering with structured and unstructured knowledge. *arXiv preprint arXiv:2012.14610*.

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.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. 2021. Representations for question answering from documents with tables and text. In *Proceedings of the 2021 Conference of the European Chapter of the Association for Computational Linguistics*, Minneapolis, Minnesota. Association for Computational Linguistics.

A Experimental Setup

The DTR model uses a batch size of 256. We pre-trained the question and table encoders for 1M steps, and fine-tuned them for a maximum of 200,000 steps, with a learning rate of 1.25e-5 using Adam and linear scheduling with warm-up and dropout rate 0.2. The hyper-parameter values were selected based on the values used by Herzig et al. (2020) on the SQA dataset. We evaluate DTR performance using recall@k, and do early stopping according to recall@10 on the dev set. We use only the tables that appear in the dev set as the corpus for the early stopping for efficiency.

For the QA reader, we initialize the model from the public TAPAS checkpoint. We use a batch size of 512, train for 50,000 steps with a learning rate of 1e-6, and dropout rate 0.2. In this setup we do not use early stopping but always train the model for the full number of steps. We limit the maximal answer length to 10 word pieces. The hyper-parameters of the QA model were optimized using a black box Bayesian optimizer similar to Google Vizier (Golovin et al., 2017). We used the hyper-parameter bounds given in Table 3.

parameter	min	max
learning rate	$1e{-6}$	$1e{-2}$
warm up ratio	0.0	0.2
dropout	0.0	0.2

Table 3: Hyper-parameter ranges for tuning the QA model.

We trained all models on 32 Cloud TPU v3. Pre-Training a retrieval model takes approx. 6 days. Training a retrieval model takes approx. 4-5h. Training a QA model takes approx. 10h.

The number of parameters is the same as for a BERT large model: 340M.

B Results

Dev and test results for the retrieval experiments are given in Table 4. Dev and test results for end-to-end QA are given in the appendix in Table 5.

	Dev			Test		
Model	R@1	R@10	R@50	R@1	R@10	R@50
BM25	17.13	42.13	57.21	16.77	40.06	58.39
DTR-Text	28.68 ± 0.37	67.76 ± 0.56	85.75 ± 0.84	32.90 ± 0.57	72.00 ± 2.14	86.86 ± 1.25
DTR-Schema	29.38 ± 2.11	67.67 ± 0.28	85.29 ± 0.19	34.36 ± 1.09	74.24 ± 1.46	88.37 ± 0.47
DTR-Text +hnbm25	35.30 ± 2.05	74.10 ± 1.50	87.85 ± 0.23	40.88 ± 0.66	78.10 ± 0.74	91.48 ± 0.21
DTR	31.58 ± 0.09	71.79 ± 0.00	88.38 ± 0.37	36.24 ± 0.57	76.02 ± 0.10	90.25 ± 0.89
DTR +hnbm25	37.86 ± 1.77	75.65 ± 0.72	89.58 ± 0.39	42.17 ± 2.91	80.51 ± 0.66	92.31 ± 0.70
DTR +hn	39.14 ± 0.98	76.13 ± 1.30	89.91 ± 0.35	42.42 ± 2.94	81.13 ± 1.61	92.56 ± 0.96
DTR -pt	9.05 ± 1.08	35.73 ± 2.21	60.34 ± 3.20	16.64 ± 1.49	47.80 ± 1.42	68.68 ± 0.91

Table 4: Table retrieval results on NQ-TABLES dev and test sets. hn: hard negatives, hnbm25: hard negatives from BM25 baseline, pt: pre-training.

		Dev				Test			
Retriever	Reader	EM	F1	Oracle EM	Oracle F1	EM	F1	Oracle EM	Oracle F1
BM25	TAPAS	18.76 ± 0.80	26.32 ± 1.23	25.57 ± 0.77	37.45 ± 0.81	21.46 ± 0.71	28.24 ± 0.78	29.51 ± 0.49	40.79 ± 0.43
DTR-Text	Bert	21.11 ± 1.09	29.17 ± 1.11	30.74 ± 1.07	42.82 ± 0.80	29.58 ± 0.85	37.38 ± 0.87	39.39 ± 0.31	51.48 ± 0.30
DTR-Text	TAPAS	27.67 ± 1.30	37.13 ± 1.74	36.44 ± 1.35	49.17 ± 1.66	33.78 ± 1.12	43.49 ± 1.23	42.83 ± 0.74	56.46 ± 0.55
DTR-Text +hnbm25	TAPAS	27.84 ± 0.95	38.00 ± 1.14	38.97 ± 0.81	52.38 ± 0.97	36.89 ± 0.77	46.67 ± 0.98	46.30 ± 0.65	59.22 ± 0.84
DTR-Schema	TAPAS	27.12 ± 1.04	36.14 ± 1.17	36.19 ± 0.93	48.81 ± 1.29	32.75 ± 0.36	42.19 ± 0.21	42.63 ± 0.68	55.05 ± 0.59
DTR	TAPAS	27.84 ± 1.62	37.77 ± 1.86	38.43 ± 0.75	51.26 ± 0.69	35.50 ± 0.45	45.44 ± 0.53	46.09 ± 0.47	59.01 ± 0.30
DTR +hn	TAPAS	28.67 ± 0.57	39.14 ± 0.46	39.38 ± 0.69	53.08 ± 0.43	37.69 ± 0.87	47.70 ± 1.05	48.20 ± 0.53	61.50 ± 0.34
DTR +hn c=1	TAPAS	23.50 ± 0.12	33.44 ± 0.30	23.50 ± 0.12	33.44 ± 0.30	31.12 ± 0.31	39.44 ± 0.20	31.12 ± 0.31	39.44 ± 0.20
DTR +hn c=50	TAPAS	23.97 ± 2.22	33.72 ± 2.81	42.11 ± 1.95	58.02 ± 2.08	30.73 ± 2.79	40.77 ± 3.26	47.26 ± 1.33	63.04 ± 1.51
DTR +hnbm25	TAPAS	27.67 ± 0.63	37.77 ± 0.93	40.26 ± 0.94	53.77 ± 0.80	36.61 ± 0.76	46.74 ± 0.82	47.46 ± 0.83	60.72 ± 0.91

Table 5: QA results on NQ-TABLES dev and test set. c: Number of candidates (default is 10), hn: With hard negatives, hnbm25: with hard negatives from BM25.