LI-RAGE: Late Interaction Retrieval Augmented Generation with Explicit Signals for Open-Domain Table Question Answering

Weizhe Lin[*2], **Rexhina Blloshmi**[1],
Bill Byrne[1,2], **Adrià de Gispert**[1], **and Gonzalo Iglesias**[1]
[1]Amazon Alexa AI
[2]University of Cambridge
wl356@cam.ac.uk {blloshmi, willbyrn, agispert, gjii}@amazon.com

Abstract

Recent open-domain TableQA models are typically implemented as retriever-reader pipelines. The retriever component is usually a variant of the Dense Passage Retriever, which computes the similarities between questions and tables based on a single representation of each. These fixed vectors can be insufficient to capture fine-grained features of potentially very big tables with heterogeneous row/column information. We address this limitation by 1) applying late interaction models which enforce a finer-grained interaction between question and table embeddings at retrieval time. In addition, we 2) incorporate a joint training scheme of the retriever and reader with explicit table-level signals, and 3) embed a binary relevance token as a prefix to the answer generated by the reader, so we can determine at inference time whether the table used to answer the question is reliable and filter accordingly. The combined strategies set a new state-to-the-art performance on two public open-domain TableQA datasets.

1 Introduction

Tabular data is ubiquitous on the Web. Open-domain Table Question Answering (TableQA), the task of answering questions grounded in tables, is increasingly attracting attention of both public and commercial research, for its value in real-world applications. Research TableQA pipelines are typically implemented with two components: a retriever and a reader. The retriever chooses a small set from the entire pool of table candidates, while the reader generates answers processing each table candidate. State-of-the-art implementations use transformer-based models for both components. In particular, the retriever is built with variants of Dense Passage Retriever (Karpukhin et al., 2020, DPR), which computes question-table similarity by using single vector representations of the question and the table. Retriever and reader can be trained separately (Herzig et al., 2021) or jointly (Pan et al., 2022) via Retrieval Augmented Generation loss (Lewis et al., 2020b, RAG). We observe three limitations which we address in this paper.

First, a table can be very large and might contain heterogeneous information across rows/columns; encoding into a fixed size vector risks information loss, which can have an impact in QA quality. One way to alleviate this issue is to replace DPR with a Latent Interaction (LI) model, which encodes text into token-level representations. In particular, we find ColBERT (Khattab and Zaharia, 2020) to be very effective, even if not pretrained for tables.

Second, RAG uses only an implicit signal to guide the retriever. Recently, Lin and Byrne (2022) proposed RAGE loss (Retrieval Augmented Generation with Explicit Signals) for visual QA, which in our setting rewards the retriever with table-level signals from the reader model in joint training.

Third, we observe empirically that the reader does not always rank answers coming from the gold table at the top. As our reader is a sequence-to-sequence model, we propose a simple modification to the training data: we prepend binary relevance tokens ('yes/no') to the answer itself. The reader learns to generate a first token indicating whether the table is relevant to the question or not.

Using these techniques, we build an end-to-end framework, LI-RAGE, and achieve state-of-the-art results on two benchmarks for open-domain TableQA, NQ-TABLES (Herzig et al., 2021) and E2E-WTQ (Pan et al., 2021). [1]

2 Related Work

While open-domain TableQA is yet a relatively unexplored problem, with only a few applications in the past couple of years, there has been extensive work on table retrieval and TableQA separately. In table retrieval, recent advances in ma-

[*]Work done as an intern at Amazon Alexa AI.

[1]We make our code available at: https://github.com/amazon-science/robust-tableqa

chine learning have enabled extracting deep features for tables with Transformers (Vaswani et al., 2017), by designing models to parse complex tabular structure (Herzig et al., 2021; Wang et al., 2021), or by simply linearizing tables with interleaving tokens to preserve its structure (Pan et al., 2022; Wang et al., 2022). In TableQA, until recently researchers assumed gold tables were given and focused on developing models that understood and answered questions over tables, i.e. the readers. Earlier models generated commands in logical forms (e.g. SQL queries) that were executable over tables (Yu et al., 2018; Lin et al., 2019; Xu et al., 2018), while recent state-of-the-art models directly predict the answers from the input question and table by either classification (Herzig et al., 2020; Yang et al., 2022, TaPas) or autoregressive generation (Liu et al., 2022, TaPEx). Following these advances, in open-domain TableQA the best performing systems are based on a retriever-reader pipeline (Herzig et al., 2021; Pan et al., 2022). Herzig et al. (2021, DTR) leverages TaPas (Herzig et al., 2020) to both initialize a DPR-like retriever and the reader. T-RAG (Pan et al., 2022) uses DPR as retriever of rows/columns by decomposing the table and generates the answer via a sequence-to-sequence reader (Lewis et al., 2020a), applying the RAG loss to refine the retriever with implicit signals during end-to-end TableQA fine-tuning. Unlike DTR and T-RAG, CLTR (Pan et al., 2021) employs only retrieval of rows and columns and obtains the answer cell by intersecting the top-scored ones. In this work we focus mainly on the retriever, and unlike previous work that relies on single vector embeddings, we leverage late interaction retrievers (Khattab and Zaharia, 2020) to achieve a finer-grained interaction between questions and tables. In contrast to T-RAG and CLTR, we do not need to decompose the table into rows and columns, but retrieve a whole table from the corpus, ensuring that the reader is given all the relevant information. In addition, we explore different techniques for *explicitly* refining the retriever during end-to-end TableQA achieving superior performance.

3 Methodology

Given a question q, the tasks are to find the *gold* table t^* from a table corpus $\mathcal{T}$, i.e. table retrieval (§ 3.1), and to derive the answer denotations $\mathcal{S}$ (1 or more cells from the table), i.e. question answering over the retrieved tables (§ 3.2). We assume that labeled datasets consisting of triples $\{(q, \mathcal{S}, t^*)\}$ are available to us. We flatten the tables into sequences with interleaving special tokens that encode its structure (see Appendix A).

3.1 Table Retrieval

In order to exploit question-table similarity at a finer-grained level than when using DPR models, we leverage LI models to encode and retrieve tables for a question. We use ColBERT, which consists of a question encoder $\mathcal{F}_q$ and a table encoder $\mathcal{F}_t$, to encode questions and tables at the *token level*:

$$\mathbf{Q} = \mathcal{F}_q(q) \in \mathcal{R}^{l_q \times d}; \mathbf{T} = \mathcal{F}_t(t) \in \mathcal{R}^{l_t \times d}, \quad (1)$$

where l_q and l_t are input token lengths of q and t. The relevance score accounts for the interactions between all question and table token embeddings:

$$r(q,t) = \sum_{i=1}^{l_q} \max_{j=1}^{l_t} \mathbf{Q}_i \mathbf{T}_j^\top \quad (2)$$

LI models extract multi-dimensional question/table embeddings and token-level similarity, as opposed to finding the similarity of single embeddings for the whole question/table in DPR, thus capturing a finer-grained interaction between them.

To train the model we exploit the gold (positive) table t^* for each question q, i.e. explicitly considering the table-level ground truth. We use in-batch negative sampling for training, per Karpukhin et al. (2020). All documents in a training batch other than t^* are considered negative for q, and denoted as $\mathcal{N}(q)$. We train with the contrastive loss $\mathcal{L}_{CL}$:

$$-\sum_{(q,t^*)} \log \frac{\exp\left(r(q,t^*)\right)}{\exp\left(r(q,t^*)\right) + \sum_{z\in\mathcal{N}(q)} \exp\left(r(q,z)\right)} \quad (3)$$

To this end, for each q, the retriever outputs K top-scoring tables $\{t_k\}_{k=1}^K$. Finally, following RAG, we obtain their (approximate[2]) conditional probability $p_\theta(\cdot|q)$ with the retriever parameters θ:

$$p_\theta(t_k|q) = \frac{\exp(r(q,t_k))}{\sum_{j=1}^K \exp(r(q,t_j))} \quad (4)$$

3.2 Retrieval-based TableQA

For the TableQA task we make use of a sequence-to-sequence Transformer-based model that directly

[2]because we sum over the top-K tables instead of all tables, assuming their probabilities are small and irrelevant.

produces an answer for a given question and table. The TableQA model p_ϕ takes as input a sequence composed of the question q and each of the retrieved tables t_k as described in §3.1, and generates an answer y_k for each input table t_k:

$$y_k = \underset{y}{\operatorname{argmax}}\, p_\phi(y|q, t_k) \tag{5}$$

Finally, the model returns the answer associated with the highest probability/confidence:

$$\widehat{y}, \widehat{t} = \underset{y, t_k}{\operatorname{argmax}}\, p_\phi(y|q, t_k) \tag{6}$$

3.3 Joint Training of Retrieval and TableQA

We train both modules jointly using a compositional loss (Lin and Byrne, 2022, RAGE), which considers signals from table relevance and answer prediction, as follows:

$$-\sum_{(q,\mathcal{S})} \Big(\sum_{k=1}^{K} \log p_\phi(s_k^*|q, t_k) + \sum_{k \in \mathcal{P}^+(q,\mathcal{S})} \log p_\theta(t_k|q) \Big) \tag{7}$$

where s_k^* is a concatenation of all comma-separated answers in $\mathcal{S}$ and $\mathcal{P}^+(q, \mathcal{S}) = \{k : y_k = s_k^* \wedge t_k = t^*\}$ is a subset of the retrieved K tables, which contains those tables that satisfy (1) being a gold table relevant to answering the question; (2) the answer generator successfully produces the correct answer from that table. The core idea is to leverage the signal from model prediction to decide which tables are beneficial to producing the correct answer. Their scores are dynamically adjusted during training, which tailors the retriever to better serve the answer generation.

3.4 Learned Table Relevance

The answer generator is trained to produce s_k^* for each input (q, t_k) pair. Ideally, we would assume that the answer generated from the gold table t^* is also associated with the highest probability from the answer generator. However, it might happen that an answer derived from a non-gold retrieved table may achieve higher confidence than the answer derived from a gold retrieved table. We propose a simple yet effective approach to improve this process: we add a *binary relevance token* preceding s_k^* as 'yes' if $t_k = t^*$, 'no' otherwise. This design aims at guiding the model to prioritize reliable answer sources at training time. At generation time, if the leading generation of a (q, t_k) pair is 'yes', we consider t_k to be a more reliable answer source and prioritize it over other input tables—that generate 'no' instead—when selecting the final prediction. We rely on the confidence scores if the leading token of all the candidates is 'no'.

4 Experimental Setup

Datasets and metrics. We evaluate our system on two benchmarks, i.e. NQ-TABLES (Herzig et al., 2021) and E2E-WTQ (Pan et al., 2021).[3]
NQ-TABLES contains generally hard questions extracted from the NaturalQuestions (Kwiatkowski et al., 2019) dataset, comprising the questions that can be answered from tables rather than plain text. For this benchmark, we evaluate the models using: Token F1, i.e. token-wise F1 score; and exact match (EM) or accuracy, i.e. whether predictions match the annotations.
E2E-WTQ contains look-up questions that require cell selection operation and is a subset of WikiTableQuestions (Pasupat and Liang, 2015). In E2E-WTQ train/valid/test splits are the same as in WikiTableQuestions, with questions limited to those that do not aggregations across multiple table cells. We evaluate models via accuracy[4].

In addition, we report Recall@K for the retrieval performance in both, which measures whether the gold table is among the top-K retrieved tables.[5]

System configurations. For the table retrieval component, we conduct contrastive experiments using both DPR and LI. We first fine-tune the official pretrained DPR or ColBERTv2 model on each dataset before using them in the joint retriever-reader training. We do not train the TableQA model from scratch, instead we warm-start the training with TaPEx, a state-of-the-art pre-trained model for tabular data understanding based on BART (Lewis et al., 2020a). Since the E2E-WTQ is very small and not enough for learning a robust TableQA model, we additionally fine-tune TaPEx on its superset, i.e. WikiTableQuestions. Note that no test samples are leaked due to this as the dataset splits of E2E-WTQ are the same as WikiTableQuestions. We select the best checkpoints based on the validation set. We set K=5 since it shows the best balance between performance and latency by both RAG and RAGE. Training details, computational cost

[3] Dataset statistics are shown in Appendix B.
[4] Also named as Hit@1 in Pan et al. (2021, 2022)
[5] We do not report metrics such as P@K, N@K, MAP used by T-RAG and CLTR, which decompose tables, being incompatible with our setting (see Appendix C).

Models	NQ-TABLES			E2E-WTQ	
	Token F1	EM	Recall@K	Accuracy	Recall@K
DTR+hn (Herzig et al., 2021)	47.70	37.69	81.13@10	-	-
CLTR (Pan et al., 2021)	-	-	-	46.75	-
T-RAG (Pan et al., 2022)	50.92	43.06	85.40@10	50.65	-
RAG	39.67	38.33	69.16@5	38.05	61.29@5
DPR-RAGE	49.68	43.02	84.35@5	48.79	59.68@5
LI-RAGE	**54.17**	**46.15**	**87.90**@5	**62.10**	**81.85**@5
(w/o joint training)	53.53	45.52	85.21@5	59.27	81.45@5
(w/o relevance tokens)	50.56	42.53	86.90@5	53.69	81.75@5
(w/o joint training & relevance tokens)	49.83	42.19	85.21@5	50.16	81.45@5

Table 1: End-to-end TableQA performance on NQ-TABLES and E2E-WTQ. Best performances are in **bold**.

Models	NQ-TABLES				E2E-WTQ			
	K=1	K=5	K=10	K=50	K=1	K=5	K=10	K=50
BM25	17.62	35.97	43.80	61.00	58.09	74.27	79.67	87.55
DPR-RAGE	58.29	84.35	90.72	97.08	33.61	59.68	66.80	88.38
(w/o joint training)	53.07	84.25	90.62	**97.81**	32.78	58.47	66.39	88.38
LI-RAGE	**59.12**	**87.90**	**92.81**	97.60	**68.46**	**81.85**	**85.89**	93.36
(w/o joint training)	53.75	85.21	90.10	97.71	66.13	81.45	84.27	**93.55**

Table 2: Retrieval performance on NQ-TABLES and E2E-WTQ. Best performances are in **bold**.

and software solution are provided in Appendix D.

Comparison systems. We compare with models from the literature, i.e. **DTR**, **CLTR**, **T-RAG** (see §2), and **BM25**—sparse retrieval baseline. Moreover, we build the following model variants:

LI-RAGE: our main system that leverages ColBERT as retriever, TaPEx as answer generator, RAGE loss for joint training and the binary relevance token in output. We also ablate the system showing the effectiveness of each feature. When disabling joint training, i.e., for ablating the model, the retriever is not updated.

DPR-RAGE: similar to LI-RAGE, except for the retriever being a DPR model.

RAG: we train the RAG (Lewis et al., 2020b) in TableQA data, initializing the retriever and answer generator with our fine-tuned DPR and TaPEx, respectively. Different from DPR-RAGE, RAG does not produce the binary relevance token and updates the retriever only with the RAG loss, which is an implicit signal from the reader.

5 Results and Discussions

5.1 Main Results

As shown in Table 1, LI-RAGE achieves the best performance across the board on both datasets, with more than 3 points improvements in Token F1 and EM in NQ-TABLES, and 11.45 points in E2E-WTQ with respect to previously best reported results in the literature. We attribute these results to the high performance of the LI retriever. On NQ-TABLES it obtains the best recall rate (87.90%) when only 5 tables are retrieved, as opposed to the previous models that achieve a lower recall rate with $K = 10$ tables, and also performs better when compared with RAG and DPR-RAGE, by a large margin.

Effects of Joint Training. Similar to the observation of Lin and Byrne (2022), joint training with RAGE improves over the frozen system on both retrieval and TableQA performance. As shown in Table 1, joint training improves the end-to-end TableQA performance on both datasets by ~0.6-2.83%, and shows a superior retrieval ability especially on NQ-TABLES (85.21 to 87.90).

Effects of Binary Relevance Tokens. As shown in Table 1, removing the binary relevance tokens greatly reduces system performance, by around 3.6% Token F1 and EM in NQ-TABLES and 8.4% in E2E-WTQ accuracy.

Effects of LI. We report the retrieval performance in Table 2. LI-RAGE achieves the highest recall, outperforming BM25 in both datasets, and DPR by ~3% on NQ-TABLES and by over 20-30% Recall@5/1 on E2E-WTQ. The large margin on E2E-WTQ is because it contains generally long tables with diverse information, and LI models prove beneficial in learning richer table representations.

5.2 Remarks of Design Rationale

We tailor our solution for TableQA, with the specific design of two main components, i.e., adding a relevance token and modifying the RAGE loss.

Relevance token. In open-domain QA, open-ended questions may have multiple correct answers and can be answered by different passages. As a result, increasing the number of retrieved passages (K) often improves the retrieval performance by enlarging the coverage of search. However, this is not the case for tables; in open-domain TableQA, the question often has only one gold table and most of the questions focus on a particular cell in the gold table. In our experiments, increasing K decreased the performance when $K > 5$ since presenting more tables to the answer generator only increases confusion and chance of mistakes (overconfident on some wrongly retrieved tables). When using relevance tokens as per our design, increasing K does not adversely impact the performance since irrelevant tables are dropped. In addition, we also explored alternative strategies that leverage retrieval scores to determine document reliability. The first strategy predicts the final answer from the table with the highest retrieval score. This setting achieves 41.04 EM on NQ-TABLES, which is even lower than our ablated LI-RAGE *w/o joint training & relevance tokens* attaining 42.19 EM (see Table 1). A second strategy weights predictions from different tables with the corresponding retrieval score, i.e., by multiplying the retrieval score (from the retriever) with the answer confidence (from the answer generator) when using K=5. This again performs poorer than our ablated LI-RAGE *w/o joint training & relevance tokens* that uses only answer generator confidence, achieving 40.91 EM on NQ-TABLES and 42.19 EM, respectively. In summary, relevance tokens work better than document retrieval scores or combination of retriever and reader scores.

RAGE loss. We modify the original RAGE loss (Lin and Byrne, 2022) to adapt it to the domain of tables. In particular, we dropped the third term in the equation, which penalizes documents when they do not contain gold answers and also do not contribute to successful question-answering. Enabling this term in the loss, penalizes $K - 1$ documents in most cases, which leads to collapsed performance of the retriever in joint training for TableQA. This is motivated by the same fact that gold tables are relatively sparse in TableQA and penalizing wrong documents leads to instability of training and quick retriever overfitting. Disabling this term instead, softens the RAGE loss by only awarding "good" tables and distinguishing good tables from bad ones, which improved the performance by around 1% EM on NQ-TABLES.

6 Conclusion

We introduce a novel open-domain TableQA framework, LI-RAGE, that leverages late interaction retrievers to enable finer-grained interaction between questions and tables. Additionally, LI-RAGE incorporates the RAGE loss and binary relevance tokens which enable significant improvements over the state-of-the-art in two challenging TableQA tasks.

7 Limitations

Our proposed system was tested on two open-domain TableQA datasets, with one of them (E2E-WTQ) being relatively small compared to the other. Also, the current open-domain TableQA datasets are limited to look-up questions. They do not cover more complicated questions that involve multiple cells and complex table operations, such as SUM/MAX/MIN/SUBTRACT in some questions of WikiSQL and WikiTableQuestion. Therefore, the effectiveness of our system should be further evaluated on more complicated datasets of larger scale in the future. Another limitation lies in the token length limit of modern Transformer models. The best-achieving models typically accept up to 1024 tokens (e.g. BART, the base model of TaPEx). This limitation becomes more obvious when tables grow longer and the information being sought go beyond the limit. We believe that, with better approaches addressing this limitation, our system can achieve better performance. The solution can be either applying sampling strategies to pick the rows and columns that are most relevant to answering the question, or increasing the capacity of future Transformer models.

References

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages

512–519, Online. Association for Computational Linguistics.

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*, 7(3):535–547.

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.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Weizhe Lin and Bill Byrne. 2022. Retrieval augmented visual question answering with outside knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Feifei Pan, Mustafa Canim, Michael Glass, Alfio Gliozzo, and Peter Fox. 2021. CLTR: An end-to-end, transformer-based system for cell-level table retrieval and table question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 202–209, Online. Association for Computational Linguistics.

Feifei Pan, Mustafa Canim, Michael Glass, Alfio Gliozzo, and James Hendler. 2022. End-to-end table question answering via retrieval-augmented generation. *arXiv preprint arXiv:2203.16714*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. Plaid: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information Knowledge Management*, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. 2021. Retrieving complex tables with multi-granular graph representation learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information*

Retrieval, SIGIR '21, page 1472–1482, New York, NY, USA. Association for Computing Machinery.

Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. 2022. Table retrieval may not necessitate table-specific model design. In *Proceedings of the Workshop on Structured and Unstructured Knowledge Integration (SUKI)*, pages 36–46, Seattle, USA. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2018. SQLNet: Generating structured queries from natural language without reinforcement learning.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland. 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.

A Table Linearization

In the retriever component, the input table is linearized into a sequence with separation tokens interleaving the table elements to make the input structure-aware, e.g. "*<SOT> [table title] <EOT> <BOC> mountain peak <SOC> elevation <EOC> <BOR> red slate mountain <SOR> 13,162 ft <EOR> <BOR> ...*".

In the reader component, the TaPEx tokenizer linearizes the table with structure-aware separation, for example, "*[HEAD] mountain peak | elevation [ROW] 1 : red slate mountain | 13 , 162 ft [ROW] 2 ...*".

B Dataset Statistics

Dataset	Train	Dev	Test	#Tables
NQ-TABLES	9,594	1,068	966	169,898
E2E-WTQ	851	124	241	2,108

Table 3: Dataset statistics.

Parameter	Value
Negative samples	4 (per positive sample)
Total GPUs	8
Learning rate	0.0001
Optimizer	Adam
Batch size (per device)	8 (DPR) / 6 (LI)
Grad. accum. steps	4
Training steps	6000 (NQ-TABLES) 600 (E2E-WTQ)

Table 4: Hyperparameters for DPR and LI training.

Parameter	Value (NQ-TABLES)	Value (E2E-WTQ)
Warmup steps	0	
Epochs	20	15
Reader LR	0.00002	0.000015
Retriever LR	0.00001	
LR decay	Linear	None
Optimizer	AdamW	
Total GPUs	8	
Batch size	1 (per device)	
Grad. accum. steps	4	
Weight decay	0.01	
Label smoothing	0.1	

Table 5: Hyperparameters for LI-RAGE training.

C CLTR and T-RAG Evaluation

In these open-domain TableQA datasets, each question is associated with only one gold table. As a result, Precision@K in retrieval has a certain upper bound at $\frac{1}{K}$. Therefore, evaluating the retriever with Recall@K is more reasonable in this case.

We confirmed with the authors of CLTR and T-RAG that they decomposed tables into single rows and columns to form the table database. In evaluating their systems on the E2E-WTQ dataset, the authors reported some retrieval metrics including Precision@K (P@K) which goes beyond the $\frac{1}{K}$ limit (e.g. T-RAG achieved 0.7806 P@5). This is because they reported a hit for a retrieved row/column as long as it belongs to the gold table. With different setups for table corpus, the retrieval metrics of their systems are not directly comparable. Therefore, we compare Recall@K with BM25 and DPR only, and compare the end-to-end TableQA accuracy with CLTR and T-RAG (which is called Hit@1 in their papers).

Models	Training Speed (iter/sec)	Training Batch Size	Training Time (mins)	Inference Speed ↑ (sec/iter)	Inference Batch Size
DPR	1.10	8	60 (NQ)/ 10 (WTQ)	-	-
LI	1.75	6	60 (NQ)/ 10 (WTQ)	-	-
DPR-RAGE	2.1	1	300 (NQ)/ 35 (WTQ)	1.22	4
LI-RAGE	0.74	1	450 (NQ)/ 50 (WTQ)	1.40	4

Table 6: Computational cost for DPR/LI retriever models and LI-RAGE and DPR-RAGE.

Parameter	Value
Warmup steps	1000
Epochs	40
Learning Rate	0.00002
LR decay	Linear
Optimizer	AdamW
Total GPUs	8
Batch size	1 (per device)
Grad. accum. steps	4
Weight decay	0.01
Label smoothing	0.1

Table 7: Hyperparameters for `tapex-large` fine-tuning on WikiTableQuestions for E2E-WTQ.

D Technical Details

D.1 Hyperparameters

The training hyperparameters are shown in Table 4, 5, and 7. The tuning of hyperparameters was performed on validation performance.

DPR: The dimension of the extracted table/question embeddings is $d = 768$.

LI: The dimension of the extracted table embeddings is $l_t \times d = l_t \times 128$, where l_t depends on the length of input tables. Following Santhanam et al. (2022b), the dimension of the extracted question embeddings is fixed to $l_q \times d = 32 \times 128$. We pad the questions with less tokens than l_q.

D.2 Indexing and Dynamic Retrieval

DPR. Following Lewis et al. (2020b), one-dimensional table embeddings are pre-extracted with the DPR model that has been finetuned on the retrieval task. The FAISS system (Johnson et al., 2019) is used to index all table embeddings which enables fast nearest neighbour search with sub-linear time complexity. In training LI-RAGE, question embeddings are dynamically extracted from the retriever, and tables with highest scores are retrieved using the precomputed index.

LI. Khattab and Zaharia (2020) proposed the first version of ColBERT, and Santhanam et al. (2022b) introduced ColBERTv2, which is an enhanced version of ColBERT. Santhanam et al. (2022a) developed an efficient search engine, PLAID, for ColBERTv2, which significantly improved the retrieval latency. We redirect readers to the aforementioned papers for more details. We started from the official ColBERTv2 implementation [6]and refactored the code base. We integrated ColBERTv2 into our training framework, so that fast and dynamic retrieval can be done during end-to-end joint training.

D.3 Computational Cost

In Table 6 we report computational cost of the proposed models. It is clear that time spent on the training of LI is not significantly increased compared to DPR training. This is because both models use contrastive learning in training. But we note that the index building time of LI is around 5 mins while that of DPR only takes 40 seconds.

In terms of joint training, the end-to-end training time of LI-RAGE is longer. This is due to (1) slightly slower dynamic retrieval during end-to-end training; (2) refining the retriever via larger multi-dimensional embeddings in comparison to one-dimensional embeddings used in DPR-RAGE. However, the inference speed is not affected much (from 1.22 sec/iteration to 1.40). This suggests that when deployed as real applications, LI-RAGE does not bring significant increase in computation.

[6] `https://github.com/stanford-futuredata/ColBERT`