Sub-Table Rescorer for Table Question Answering

Atsushi Kojima
Money Forward, Inc.
21F Tamachi Station Tower S, 3-1-21 Shibaura, Minato-ku, Tokyo 108-0023, Japan
kojima.atsushi@moneyforward.co.jp

Abstract

We propose a sub-table rescorer (STR) to improve the performance of an inner table retriever (ITR)-based inference for the table question answering. Tabular language model (TLM) truncates the sequence of a long table due to their input token limits. It leads to accuracy degradation. To solve this problem, ITR extracts sub-table candidates, which correspond to a part of an entire greater original table on the basis of relevance scores to the question for each of the columns and rows. Then, the top$N$ longest sub-tables are selected. Our proposed STR estimates the relevance score between a question and each sub-table. In this work, we explored two different methods to integrate STR to the ITR-based inference. In the first method, STR rescores sub-table candidates, and the top$N$ sub-tables are chosen. Then, TLM outputs the most confident answer. In the second method, the score calculated by STR is interpolated with the score calculated by TLM. Then, the most confident answer is chosen. In the experiment, we evaluate the performance on the WikiTableQuestions dataset. By applying STR to the ITR-based inference, we observed 4.4% and 6.3% relative reductions in error rate in the rescoring- and score-fusion-based methods, respectively.

Keywords: Tabular language model, Retriever model, Question answering

1. Introduction

Recently, a Transformer (Vaswani et al., 2017)-based tabular language model (TLM) has been widely used for table question answering (Liu et al., 2022; Jiang et al., 2022; Herzig et al., 2020), because TLMs can be efficiently trained in a self-supervised fashion (Lewis et al., 2019; Devlin et al., 2019). For an inference, a fine-tuned TLM gets a token sequence of a question and a table, and outputs the answer from the table content.

For many TLMs, handling the sequence of a long table is a challenging task. Many TLMs have a maximum limitation of the input token length for efficient training and inference. For instance, the maximum token lengths of TaPEx (Liu et al., 2022) and OmniTab (Jiang et al., 2022) are both limited to 1024 tokens. Therefore, TLMs simply truncate the sequence of a table if the length of the input sequence exceeds the maximum limitation of TLMs, which leads to accuracy degradation (Lin et al., 2023).

To solve this problem, the inner table retriever (ITR) (Lin et al., 2023) that extracts N sub-tables, which correspond to a part of a greater original table, has been proposed. To preserve the information related to a question in sub-tables, ITR uses the retriever model, which calculates relevance scores to the question per table item (i.e., each of the columns and rows in a table). Sub-table candidates are obtained on the basis of relevance scores, and the top$N$ longest sub-tables that do not exceed the maximum limitation of TLM are selected. To generate an answer, TLM gets token sequences of N sub-tables and outputs N answer candidates. Then, the most confident answer is chosen from among the N answer candidates.

In this work, we propose a sub-table rescorer (STR), which calculates the relevance scores for a question per sub-table candidate to improve the performance of an ITR-based inference. For integrating STR to the ITR-based inference, we explored two different methods. In the first method, STR is used for rescoring. In this method, STR rescores sub-table candidates obtained by ITR and outputs the top$N$ sub-tables by sorting sub-table candidates by relevance score in descending order. In the second method, STR is used for score fusion. In this method, the score calculated by STR is interpolated with the score calculated by TLM for each N answer candidate to obtain the most confident answer from answer candidates. We will here report the results of our experiments conducted using the public WikiTableQuestions dataset (Pasupat and Liang, 2015).

2. Related Work

There are three steps to obtain N sub-tables from the original table using ITR (Lin et al., 2023). In the first step, the scores of relevance between question and table items are calculated using a retriever model (Karpukhin et al., 2020) that consists of question and table item encoders. In the second step, sub-table candidates, which consist of the cells at the intersection of the selected columns and rows, are obtained using table items sorted in

LREC-COLING 2024, pages 15422–15427
20-25 May, 2024.

the descending order of relevance scores. In the third step, the top$N$ longest sub-tables, which do not exceed the maximum limitation of TLM, are selected.

To obtain answers, all N sub-tables are used. TLM gets N sub-tables and outputs the most confident answer.

3. STR

We propose STR that calculates the relevance scores of sub-tables to improve the performance of the ITR-based inference. Compared with the retriever model used in ITR, which calculates the relevance scores of table items, our proposed STR can capture a long-range context, because a relevance score is calculated per sub-table. STR consists of question and sub-table encoders, which convert questions and sub-tables to hidden vectors, respectively.

$$\boldsymbol{h}_{\text{question}} = \text{QuestionEncoder}(X_{\text{question}}; \theta_{\text{question}}), \quad (1)$$

$$\boldsymbol{h}_{\text{sub-t}} = \text{Sub-TableEncoder}(X_{\text{sub-t}}; \theta_{\text{sub-t}}), \quad (2)$$

where X_{question} and $X_{\text{sub-t}}$ are the question and sub-table, $\text{QuestionEncoder}(\cdot)$ and $\text{Sub-TableEncoder}(\cdot)$ are the question and sub-table encoders, and θ_{question} and $\theta_{\text{sub-t}}$ are the parameters of the question and sub-table encoders, respectively. The relevance score is obtained by calculating the cosine similarity between the hidden vectors $\boldsymbol{h}_{\text{question}}$ and $\boldsymbol{h}_{\text{sub-t}}$. As the architecture of the encoders, BERT (Devlin et al., 2019) is used.

In this work, we explored two different methods to integrate STR to the ITR-based inference. In the first method, STR is used for rescoring. Figure 1 shows the rescoring-based method. In this method, STR gets sub-table candidates obtained by ITR and rescores each sub-table candidate. Then, the top$N$ sub-tables are chosen by sorting sub-table candidates by relevance score in descending order. TLM gets N sub-tables and outputs the most confident answer. In the second method, STR is used for score-fusion. Figure 2 shows the score-fusion-based method. In this method, the relevance score calculated by STR is interpolated with the score calculated by TLM for each N answer candidate obtained by ITR, and the most confident answer is chosen from the N answer candidates on the basis of the score interpolated by TLM and STR. The interpolated score for ith answer candidate y_i is calculated as

$$\begin{aligned}\text{score}_{y_i} = &\log P_{\text{tlm}}(y_i|\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}}) \\ &+\alpha \log s(\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}}),\end{aligned} \quad (3)$$

where sub-t$_i$ is the ith sub-table. $\log P_{\text{tlm}}$ is the score calculated by TLM. $\boldsymbol{h}_{\text{sub-t}_i}$ is the hidden vector of the ith sub-table. $s(\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}})$ is the normalized score of relevance between the question and the ith sub-table. α is a tunable parameter. The normalized score of relevance is calculated as

$$s(\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}}) = \frac{\text{sim}(\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}}) + 1}{2}, \quad (4)$$

where $\text{sim}(\boldsymbol{h}_{\text{sub-t}_i}, \boldsymbol{h}_{\text{question}})$ means the operation for calculating the cosine similarity between the hidden vectors $\boldsymbol{h}_{\text{sub-t}_i}$ and $\boldsymbol{h}_{\text{question}}$.

STR is trained by minimizing contrastive loss (Karpukhin et al., 2020). Loss λ can be written as

$$\lambda = -\log \frac{\text{sim}(\boldsymbol{h}^{\text{pos}}_{\text{sub-t}}, \boldsymbol{h}_{\text{question}})}{\text{sim}(\boldsymbol{h}^{\text{pos}}_{\text{sub-t}}, \boldsymbol{h}_{\text{question}}) + \sum_{n=1}^{K} \text{sim}(\boldsymbol{h}^{\text{neg}}_{\text{sub-t}_n}, \boldsymbol{h}_{\text{question}})}, \quad (5)$$

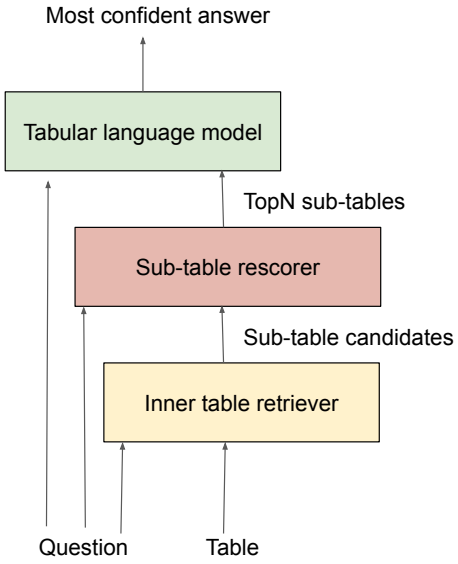

Figure 1: Rescoring-based method.

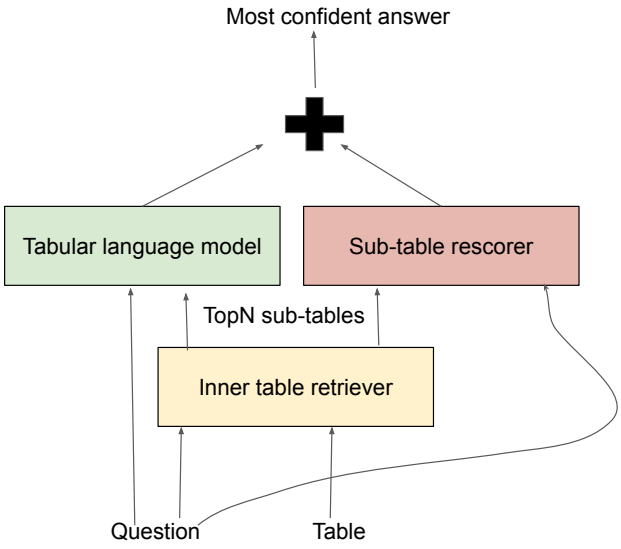

Figure 2: Score-fusion-based method.

where $\boldsymbol{h}^{\text{pos}}_{\text{sub-t}}$ is the hidden vector of the sub-table that includes gold answers. $\boldsymbol{h}^{\text{neg}}_{\text{sub-t}_n}$ is the nth hidden vector of the sub-table that does not include gold answers. K is the number of negative samples. To get positive samples, sub-tables that include gold answers are trimmed from the main table randomly. As for negative samples, sub-tables that do not include gold answers are also trimmed from the table randomly. Both of the positive and negative samples are sampled from the same table.

4. Experiments and Results

4.1. Experimental Setup

As the experimental dataset, we used the WikiTableQuestions dataset (Pasupat and Liang, 2015), which consists of 22033 questions. To train STR, the WikiTableQuestions training dataset was used. We evaluate the performance of STR using the WikiTableQuestions test and development sets. To split data into training and test datasets, we followed the data split method in the official WikiTableQuestions so that the tables in the test dataset were not included in the training dataset. STR was applied only to tables that exceeded the maximum limitation of the input token length of TLMs similar to ITR (Lin et al., 2023). For the evaluation metric, we used the denotation accuracy, because TLMs might generate answers with the same meaning but different notation.

STR was trained by fine-tuning BERT encoders (Karpukhin et al., 2020) trained using the Natural Questions dataset (Kwiatkowski et al., 2019; Lee et al., 2019). Table 1 shows the training conditions of STR. Ten positive samples were obtained from the main table, and four negative samples were obtained per positive sample.

Parameter	Value
Number of positive samples per table	10
Number of negative samples per positive sample	4
Learning rate	10^{-5}
Optimizer	Adam
Weight decay	0.01
Number of epochs	30
Batch size	8

Table 1: Training conditions for STR.

For the training of STR, we used the Transformer learning schedule (Vaswani et al., 2017). We also used the Adam optimizer (Kingma and Ba, 2014), setting $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-7}$. As TLMs, TaPEx (Liu et al., 2022)[1] and OmniTab (Jiang et al., 2022)[2] were used. For inference, we used greedy decoding. In addition, we set the number of sub-tables N to 10. We set the normalized score of relevance weight α to 0.7 for both TaPEx and OmniTab.

4.2. Results

The results of STR are shown in Table 2. B0 is the result of TaPEx. C0 is the baseline result obtained by the ITR-based inference using TaPEx. By applying STR to the ITR-based inference using TaPEx (C0), we observed 3.7% and 5.6% relative error reductions in the rescoring- and score-fusion-based methods (D3 and E2) on the test set, respectively. By comparing the two methods, we found that the score-fusion-based method (E2) was better than the rescoring-based method (D3). As for OmniTab, we also observed an improvement achieved by applying STR to the ITR-based inference. F0 is the result of OmniTab. G0 is the baseline result obtained by the ITR-based inference using OmniTab. By applying STR to the ITR-based inference (G0), we observed 4.4% and 6.3% relative error reductions in the rescoring- and score-fusion-based methods (H0 and J0) on the test set, respectively.

Exp	Method	Dev	Test
B0	TaPEx (Liu et al., 2022)	57.2	55.5
C0	+ ITR (Lin et al., 2023)	58.4	56.9
D3	++ Rescoring w/ STR	59.4	58.5
E2	++ Score fusion w/ STR	**62.0**	**59.3**
F0	OmniTab (Jiang et al., 2022)	61.0	62.1
G0	+ ITR (Lin et al., 2023)	62.1	63.4
H0	++ Rescoring w/ STR	63.1	65.0
J0	++ Score fusion w/ STR	**63.7**	**65.7**

Table 2: Results of STR. Results are given as accuracy.

Table 3 shows the effect of α on the accuracy of the score-fusion-based method on the test set. TaPEx was used as the TLM. The result for $\alpha = 0$ (C0) was obtained by the ITR-based inference without STR. By applying the score-fusion-based method, we found that the accuracy was improved at all settings of α (E0, E1, E2, E3, E4, and E5). When we set α to 0.7, we obtained the highest accuracy (E2). When we set α to be larger than 1 (E3, E4, and E5), the accuracy was lower than that for E2.

Table 4 shows the effect of N on the rescoring-based method on the test set. TaPEx was used as the TLM. In this table, C0 was

[1] microsoft/tapex-large-finetuned-wtq
[2] neulab/omnitab-large-finetuned-wtq

Exp	α	Accuracy
C0	0	56.9
E0	0.2	58.8
E1	0.5	59.2
E2	0.7	**59.3**
E3	1	59.2
E4	1.2	59.2
E5	1.4	59.2

Table 3: Effect of α on the accuracy of score-fusion-based method. As for TLM, we used TaPEx.

Exp	Rescoring	N	Accuracy
C0		10	56.9
D0	✓	1	56.5
D1	✓	3	56.5
D2	✓	5	57.5
D3	✓	10	**58.5**
D4	✓	15	**58.5**

Table 4: Effect of N on the accuracy of rescoring-based method. As for TLM, we used TaPEx.

obtained by the ITR-based inference without STR. The rescoring-based method at $N = 5$ (D2) outperforms ITR-based inference without STR at $N = 10$ (C0). Therefore, the rescoring-based method helps TLM perform efficient inference with a smaller N of sub-tables.

4.3. Analysis of Scores Calculated by STR

Sub-table	Score by STR	Prediction by TaPEx
[HEAD] Nation | Bronze		
[ROW] 1: Italy | 6		
[ROW] 2: Yugoslavia | 1		
[ROW] 3: Sweden | 0		Yugoslavia
[ROW] 4: Panama | 1	-6.2	(incorrect)
[HEAD] Nation | Bronze		
[ROW] 1: Yugoslavia | 1		Romania
[ROW] 2: **Romania** | 2	-4.4	(correct)

Table 5: An example of inference results obtained by STR. The question and gold answer are "which country won more bronze medals, Romania or Yugoslavia?" and "Romania", respectively. The sub-table in the first row of the table shows the top1 result obtained by ITR before rescoring by STR. The sub-table in the second row of the table is the top1 result after rescoring by STR.

Table 5 shows an example of inference results obtained by STR. The score calculated by STR for the sub-table in the second row of the table is higher than that calculated for the sub-table in the first row of the table, because the sub-table in the second row of the table includes the gold answer. As a result, TLM can output the correct answer by choosing the sub-table that includes the gold answer.

We compared the scores calculated by TaPEx and STR on sub-tables that included the gold answer and sub-tables that did not include the gold answer. Figure 3 shows the distributions of scores calculated by TaPEx and STR on a sample in the WikiTableQuestions dataset. In this figure, the win (blue bar) and lose (orange bar) sub-tables represent the sub-tables that include and do not include the gold answer, respectively. Regarding the distribution of scores calculated by TLM, the scores in some sub-tables that include the gold answer are lower than those in some sub-tables that do not include the gold answer. Regarding the distribution of scores calculated by STR, we can see that the scores in the sub-tables that include the gold answer tend to be higher than those in sub-tables that do not include the gold answer.

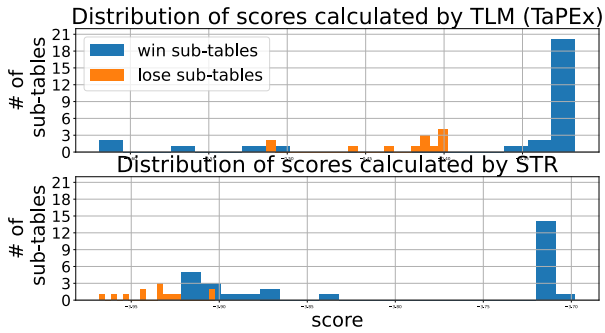

Figure 3: Comparison distributions of scores calculated by TaPEx and STR.

5. Conclusion

In this study, we proposed STR, which calculates relevance scores per sub-table to improve the performance of the ITR-based inference. In this work, we explored two different methods to integrate STR to the ITR-based inference. In the first method, STR is used for rescoring. In this method, STR gets sub-table candidates obtained by ITR and rescores each sub-table candidate. Then, the top$N$ sub-tables are chosen by sorting sub-table candidates by relevance score in descending order. TLM gets N sub-tables and outputs the most confident answer. In the second method, STR is used for score fusion. In this method, the score calculated by STR is interpolated with the score calculated by TLM. Then, the most confident answer is chosen from among the answer candidates. In

the experiment, we observed 4.4% and 6.3% relative reductions in error rate in the rescoring- and score-fusion-based methods by applying STR to the ITR-based inference, respectively. In the future, we will investigate ensemble training using multiple TLMs.

6. Bibliographical References

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 5998–6008. Curran Associates, Inc.

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

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.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin 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.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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*, pages 1470–1480, Beijing, China. 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. *arXiv preprint arXiv:2107.07653v3*.

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.

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

Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adrià de Gispert, and Gonzalo Iglesias. 2023. An inner table retriever for robust table question answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 9909–9926, Toronto, Canada. Association for Computational Linguistics.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.

7. Language Resource References

Karpukhin, Vladimir and Oguz, Barlas and Min, Sewon and Lewis, Patrick and Wu, Ledell and Edunov, Sergey and Chen, Danqi and Yih, Wen-tau. 2020. *Dense passage retrieval for open-domain question answering*. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. *Compositional semantic parsing on semi-structured tables*. Association for Computational Linguistics.

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

Zhengbao Jiang and Yi Mao and Pengcheng He and Graham Neubig and Weizhu Chen. 2022. *OmniTab: Pretraining with natural and synthetic data for few-shot table-based question answering*. Association for Computational Linguistics.