Research Report

Rethinking Table Retrieval From Data Lakes

EXISTING TABLE RETRIEVAL APPROACHES ESTIMATE EACH TABLE'S RELEVANCE FOR A PARTICULAR INFORMATION NEED AND RETURN A RANKING OF THE MOST RELEVANT TABLES. THIS APPROACH IS NOT IDEAL SINCE THE RETURNED TABLES OFTEN INCLUDE IRRELEVANT DATA AND THE REQUIRED INFORMATION MAY BE SCATTERED ACROSS MULTIPLE TABLES. TO ADDRESS THESE ISSUES, WE PROPOSE THE IDEA OF FINE-GRAINED STRUCTURED TABLE RETRIEVAL AND PRESENT OUR VISION OF R2D2, A SYSTEM WHICH SLICES TABLES INTO SMALL TILES THAT ARE LATER COMPOSED INTO A STRUCTURED RESULT THAT IS TAILORED TO THE USER-PROVIDED INFORMATION NEED. AN INITIAL EVALUATION OF OUR APPROACH DEMONSTRATES HOW OUR IDEA CAN IMPROVE TABLE RETRIEVAL AND RELEVANT DOWNSTREAM TASKS SUCH AS TABLE QUESTION ANSWERING.

Jan-Micha Bodensohn Carsten Binnig

Introduction

Table retrieval from data lakes has recently become important for many downstream tasks. For example, data engineering efforts often involve retrieving relevant tables from data lakes to provide the basis for further data analysis and exploration. Moreover, table retrieval has also gained importance beyond data engineering. For example, recent open-domain table question answering (TQA) approaches combine table retrieval with large language models (LLMs) to answer questions based on tables.

Existing table retrieval approaches often follow the same procedure: Given a data lake of tables and a user-provided information need, they estimate each table's relevance to the information need and return a ranking of the most relevant tables. While this approach helps users to search through large collections of tables, it still has severe limitations:

(1) *Irrelevant data in full tables.* A major issue of existing table retrieval approaches is that they are coarse-grained and often only return full tables. While this approach is somewhat tractable for small web tables, retrieving full tables can cause high manual filtering overheads for real-world table repositories such as enterprise data lakes, where tables often contain hundreds of columns and millions of rows.

(2) *Scattered information.* A second major issue is that relevant information is often scattered across multiple tables in the ranked result list, causing additional difficulties and overheads for users to identify and combine the relevant information.

Towards Fine-Grained Structured Table Retrieval

To address these limitations of existing table retrieval approaches, we propose the idea of fine-grained structured table retrieval, where the goal is to use data from the data lake tables to construct small relational databases that are tailored to the specific user-provided information needs. We further present our vision of R2D2 (Retrieving Relevant Data from Data Lakes), a system which tackles this goal by diverging from existing table retrieval approaches in two key aspects:

(1) Instead of retrieving full tables, R2D2 first splits the tables into smaller tiles (sub-tables), allowing the retrieval step to select which parts of the table to return to the user. In contrast to existing table segmentation approaches, we propose a novel representation-based slicing algorithm that is agnostic to the information needs but takes the table data into account to group related rows and columns in the same tiles.

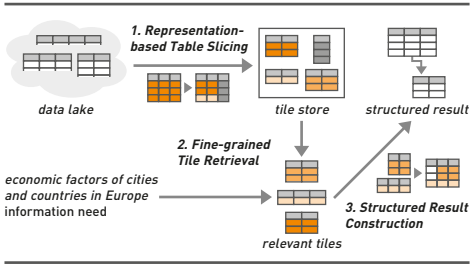

Figure 1: Prototype Overview

(2) Instead of returning ranked lists of independent tables as a result, R2D2 detects relationships between the individual tiles in the result and uses them to compose structured results in the form of small relational databases. Returning multiple tables allows us to better capture the relevant entities of the information needs. Moreover, we argue that such structured results come more naturally to users and let them get an overview of the retrieved data more easily.

Prototype Implementation

As shown in Figure 1, R2D2 works in three stages. The first stage uses our novel representation-based slicing algorithm to slice each table into a set of small tiles. The second stage then uses embeddings of the information need (i.e., the user's question) and tiles to retrieve a set of relevant tiles. Finally, the third stage derives the relationships between the retrieved tiles to compose the structured result that represents the data satisfying the user's question.

Representation-based slicing. By slicing the

tables into tiles, we allow the retrieval step to filter which parts of the table to return to the user. Naively partitioning the tables into row-wise or column-wise chunks may leave related information, that is relevant to the same information needs, scattered across multiple tiles. To tackle this issue, we propose a novel slicing algorithm that works based on representations of the table data. In each step, we partition the table either horizontally or vertically to incrementally increase the focus of the resulting tiles whilst keeping related rows and columns in the same tile.

The slicing algorithm starts with a full table and then recursively splits it into multiple tiles. In each step, we first determine which tile to split. Since we want to keep related information in the same tile, we compute a measure of semantic heterogeneity d^i for each tile and split the tile with the highest heterogeneity. To operationalize this, we compute representations for each tile's rows and columns using a pre-trained language model (Reimers and Gurevych, 2019). Next, we determine the maximum pairwise cosine distances d^i_{row} and d^i_{col} between the row and column representations of tile i and consider the greater of the two distances as the tile's semantic heterogeneity d^i. We also use d^i_{row} and d^i_{col} to determine whether to split the tile row-wise or column-wise based on which of the two values is greater. To then actually split the tile, we cluster the tile into k clusters of rows/columns based on the tile's row/column representations. We then continue to recursively split the resulting tiles until a specified number of tiles is reached.

Fine-grained tile retrieval. Given a user-provided information need, R2D2 performs a cosine similarity-based vector search to retrieve a set of relevant tiles. To compute the representation for the information need, we simply embed the natural language string. To compute the tile representations, we compute embeddings for all rows and columns of the source table. The tile representations are then computed by averaging the embeddings of all corresponding rows and columns.

Structured result construction. Finally, R2D2 uses the retrieved tiles to compose the structured result that is returned to the user. The main idea is that instead of returning a ranked list of individual tiles, we aim to construct a structured set of tables that better represents the relationships between the data. Our prototype of R2D2 merges all tiles that come from the same source table and thus have explicit structural relationships. For each source table, we determine the smallest sets of rows and columns so that their overlap contains all retrieved tiles and construct one result table based on this overlap.

Table Question Answering

To demonstrate the viability of the ideas behind our approach, we combine R2D2 with a LLM to evaluate how our approach can benefit TQA as a first downstream task.

Data sets. We experiment on two data sets: Spider (Yu et al., 2018) is a text-to-SQL data set containing relational databases with natural language questions and matching SQL queries. Open-WikiTable (Kweon et al., 2023) is an open-domain TQA data set containing web tables with natural language questions and matching SQL queries.

Experimental setup. We combine our table retriever (R2D2) with a LLM (OpenAI's GPT-4 Turbo) in a retriever-reader framework, i.e., the LLM receives the retrieved tables and the question and generates the answer. To analyze the effectiveness of the retrieval approach, we constrain the input size to 1,000 tokens for Open-WikiTable and 200 tokens for Spider. We compute the exact match (EM) accuracy to evaluate the TQA and measure the recall (R) based on which cells are included in the model input.

Results. Table 1 shows that R2D2 vastly improves the accuracy and recall on both data sets compared to retrieving full tables (no slicing). Furthermore, we see that our representation-based slicing algorithm outperforms naïve row-wise and column-wise slicing on Open-WikiTable. On Spider, our approach outperforms row-wise slicing and performs on par with column-wise slicing. Finally, we see that omitting the result construction and instead using individual tiles as input to the LLM leads to sharp declines in accuracy and recall, indicating the importance of the result construction step.

	OWT		Spider	
	EM	R	EM	R
No Slicing	.08	.07	.08	.44
Row-Wise Slicing	.29	.38	.09	.64
Column-Wise Slicing	.25	.48	**.13**	**.84**
R2D2 W/O Result Construction	.12	.21	.11	.65
R2D2	**.32**	**.51**	**.13**	.82

Table 1: TQA EM Accuracy and Recall

References

Kweon, S.; Kwon, Y.; Cho, S.; Jo, Y.; Choi, E.:
Open-WikiTable: Dataset for Open Domain Question Answering with Complex Reasoning over Table.
Working Paper, 2023.

Reimers, N.; Gurevych, I.:
Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.
In: Proceedings of the 14th Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP); Hong Kong, China, 2019.

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; et al.:
Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task.
In: Proceedings of the 23rd Conference on Empirical Methods in Natural Language Processing (EMNLP); Brussels, Belgium, 2018.