11711: Flexible Graph-based Retrieval for Complex Tables

Yang Yang[*] **Yansheng Cao** [*]
{yangy4, yanshenc}@andrew.cmu.edu

1 Dataset and Task

For this project, our team chose to explore the task of Ad-hoc table retrieval, as we think table is one of the powerful, versatile and interesting format for working with data through our experiences as data scientist and quantitative researchers. There are a massive number of tables in real applications such as on Wikipedia, Web and as excel spreadsheets; and to deal with a large collection of tables, a powerful best-match based search model is important especially working with tabular data in the wild. The Ad-hoc table retrieval task can be described as given a query, keyword or short phrase, return a ranked list of tables from a table corpus that are relevant to the query. During our research, we discovered that the Ad-hoc table retrieval is currently an important and yet understudied field compared to other table-to-text tasks such as semantic parsing or question answering, especially given that table retrieval is a core building block in the aforementioned other table-to-text tasks. Hence, the goal of our project is to better understanding the connection between text and semi-structured table, and better leverage the semi-structured nature of tables to improve the performance of Table retrieval.

We performed extensive research on existing literature and we found only three major data sources: WikiTables (Cafarella et al., 2009), (Venetis et al., 2011), (Zhang and Balog, 2018a), WebQueryTable (Yan et al., 2017)and NQ_TABLE (Herzig et al., 2021). Given this scarcity, we choose to work on improving Table based Retrieval using WikiTables (Cafarella et al., 2009), (Venetis et al., 2011), (Zhang and Balog, 2018a) and WebQueryTable (Yan et al., 2017), which are the benchmarks that most works in the research community experiment on, and covers are from real-world search log with diverse intents. Both dataset contains query/questions with answers embedded in the tables, as well as tables with more complex structures compared to standard relational table.

Given the challenge of table's complex structures, we aim to leverage both Linearized pretrained language model and Graph neural network with custom graphs to effectively and efficiently model the influence of each aspect of table: row, column, cell, row header, column header, and the interactions between these contents for table retrieval. Our main contribution centers around developing novel ways to construct tabular graph for different table types, which extends the flexibility of the retrieval model.

2 Literature Review

Table based retrieval system have received more attention in the information retrieval in the recent years following the advances of dense retrieval for textual data. The formal definition of our task is to develop a system that given a keyword query q and the set of tables $T_{corpus} = T_1, ..., T_N$, the system automatically returns a ranked list of tables according to how likely the query q would be satisfied by the information in each table. Prior approaches before 2020, for the Ad-hoc retrieval task, follows two approaches. 1) Information retrieval approach, consider table as the same as linear textual data and apply language model using Neural Network. 2) Feature engineering approach by either considering only lexical matching between tables and queries, or shallow modelling of semantic matching using Word2Vec. More recently, as neural architectures evolved, there is an emerging trend towards leveraging the semi-structured nature of tables to better understanding the connection between text and semi-structured table using pre-trained language models such as BERT (Devlin et al., 2018) and work that combines pre-trained LM with graph-based neural

[*]Everyone Contributed Equally – Alphabetical order

models to jointly model both the textual contents and the internal structures of tables.

In the following sections, we will discuss the limited set of open-sourced datasets and the various approaches that researchers have proposed to improve the performance of table-retrieval systems.

2.1 Dataset

WikiTables The WikiTables corpus (Bhagavatula et al., 2015) is one of the most important dataset for QA on tabular data, the dataset comprises of 1.6 million tables extracted from Wikipedia. (Zhang and Balog, 2018a) developed a table retrieval dataset using WikiTable corpus by sampling 60 test queries from two independent sources based on Web users (Cafarella et al., 2009), and query logs from Google Squared (Venetis et al., 2011). For this query subset, 3,120 candidate tables were extracted from Wikipedia and each candidate query-tables pair is labeled by annotators with one of three relevance scores with : 0 (irrelevant), 1 (relevant), and 2 (highly relevant). Each table is associated with contextual information including a caption, Wikipedia's page title and section title. Out of the 3120 query-table pairs, 377 are labeled as highly relevant, 474 as relevant, and 2269 as non-relevant, and 1800 of these tables contains some nested structure which is suitable for our goal.

WebQueryTable Chen et al(Yan et al., 2017) use search logs from a commercial search engine to get a list of queries that could be potentially answered by web tables. The dataset contains 21,113 web queries and 273,816 web tables from Wikipedia. Each query table pair is obtained from the top ranked Web page of a commercial search engine with manual evaluation, and fir each table, the table caption was also given as additional contextual information. The tables in the WebQueryTable is much smaller compared to WikTables and is dominated by classic relational tables, despite this, the large query and table collection size motivates us to experiment our system on this dataset.

2.2 Systems and Architectures

STR (Zhang and Balog, 2018b) summarized the set of lexical and semantic features that have been proposed and used in the Ad-hoc table retrieval tasks and introduced various fusion techniques to learn semantic (vectorized) representation of information in tables built from entities, category, words and entity graphs.

Table2Vec (Mikolov et al., 2013a) and (Mikolov et al., 2013b) proposed the Word2Vec model which drastically changed the direction of representation Learning on Text, and following the advance of dense distribution embedding of text using neural networks, researchers in the IR community adapted the Skipgram Word2Vec model (Mikolov et al., 2013b) as backbone to model table headings, entities, and all of the table data using heading embeddings, entity embeddings and word embeddings.

BERT and BERT4TR Since language model pre-training has successfully improved performance on many natural language processing tasks, (Devlin et al., 2018) built the Bidirectional Encoder Representations from Transformers (BERT) model. And with the advancement of large-scale pre-trained models such as BERT, research on Ad-hoc table retrieval shifted from manually crafting features/embedding based features built from Word2Vec towards learning representations for natural language sentences and semi-structured tables using semi-supervised learning.

BERT for Table Retrieval (Chen et al., 2020) applied the pre-trained language model BERT (Devlin et al., 2018)to encode flattened representation of tables for the ad-hoc table retrieval task. The authors proposed an system architecture that leverages BERT to encode table information and perform relevance matching. The pipeline is separated into three components, the first is an embedding based selection process to select the most relevant rows from tables with respect to queries, followed by using BERT to encode the query and contextual information of a table and selected tabular content is flattened as a sequence and encoded by BERT. The vectorized representation generated by BERT is then concatenated with manual a query-table features using a MLP to compute the final relevance score.

TAPAS TAPAS is a model proposed by google research (Herzig et al., 2020) which is a weakly supervised QA model on tables without generating logical forms. TAPAS extends upon BERT architecture with additional embeddings that could capture table's structure and two classification layers to select cells and predict a corresponding aggregation operator. TAPAS is trained end to end, encoding tables crawled from Wikipedia as input.

TaBERT TABERT is a pretrained language model that jointly learns representations for natural language sentences and structured tables (Yin et al., 2020). TABERT linearizes the structure of tables to be compatible with BERT model. To cope with large tables due to max length limit of BERT, the TABERT authors proposed content snapshots, a content selection method to encode the most relevant subset of table content to the query/input. This strategy is further combined with a vertical attention mechanism to share information among cell representations in different rows.

Dense Retrieval for Table QA With the advance of pre-trained language models, open-domain QA over a corpus of textual passages(free-text input data) have seen a lot of progress, specifically using a two-stage framework consisting of a retriever model that first selects a small subset of candidate passages relevant to the query, followed by a reader model that selects the "best" answer given the selected relevant contexts. Specifically on the retriever side, dense retrieval approaches targeted for retrieving passages (Lee et al., 2019) (Guu et al., 2020) and (Karpukhin et al., 2020) have shown strong performance on various QA dataset from Wikipedia to Reddit Forum data. Given the connection between text and table data, more recent researches in Ad-hoc table retrieval focuses on modify the retriever to better handle tabular contexts. (Herzig et al., 2021) designed a retriever model that contextually represent text and a table jointly using the TAPAS architecture which includes table specific embeddings that capture the table's structure, such as row and column ids.

Most pre-trained language modelling based on extending BERT to tabular data leverages row-wise and col-wise attention to model the interaction between cells in the table similar to a fully connected graph fashion based on attention mechanism, however, this not only introduces high computational cost for example fine-tuning experiment for dual TAPAS encoder for the query and the table as the retriever model took 6 hours using 32 TPUs; but also does not necessarily deliver the best performance compared to a more sparse graph representation of a table, especially in the context of the ad-hoc retrieval task.

Graph Base Table Retrieval(GTR) Compared to linear language models on plain textual data (attention over row or column), table data often have different and complex structures in terms of layouts, for example, the table structure may be relational, matrix-format, nested or contains pure entites etc. Therefore, (Wang et al., 2021) proposed a Graph-based table retrieval system that constructs an multi-granular tabular graph for any arbitrary table using cell node adjacency, and uses a graph transformer based model architecture proposed by (Koncel-Kedziorski et al., 2019), similiar to graph attention network, to characterize both cell content and layout structures with cell nodes representations initialized using FastText(Mikolov et al., 2018), followed by a BERT as the query-context matching module and FastText as the encoder for the query for the query-graph matching module. Finally, the query-table matching representation is concatenated with the query context representation and fed to a MLP to calculate the relevance score between query representation and table representation.

3 Baseline Results

3.1 GTR

Motivation We select the GTR as our baseline because we find that both Tapas and TaBERT makes strong assumption of tabular structures, has the same limitation to max sequence length as BERT and are very expensive to train, as finetuning the dense table retrieval model based on TAPAS used 32 Cloud TPU. While GTR is both flexible and intuitive from a table modelling perspective, as shown in Figure 1, with Graph Representation learning, the retrieval system can handle arbitrary table layout structures, and can easily summarize multi-granular contents using global nodes in the GTR context in constrast to linearized pre-trained LM which flattens the table and ignores a lot of the potentially important connections between nodes across rows. Furthermore, the GTR retrieval systems showed promising results compared to dense retrieval using TaBERT, furthermore the computational demand is far more tractable, a single 2080Ti GPU or v100 GPU suffices for both of the datasets that we are working with.

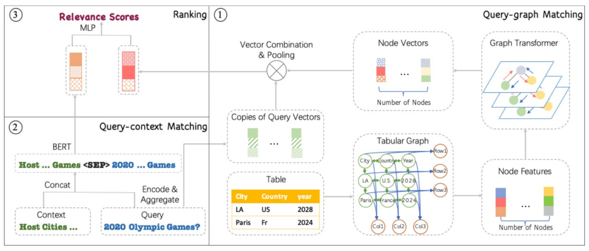

Figure 1: Overall Architecture of GTR system taken from (Wang et al., 2021)

Setting and Dataset For the baseline, we use the code-base released by the original authors of GTR To be consistent with (Wang et al., 2021). We did not use the pretrained version and instead trained our model from scratch on both WikiTables and WebQueryTable. We used a BERT_base model for the query-context matching module and we used Graph Transformer (Koncel-Kedziorski et al., 2019) with 4 layers and linear hidden dimension of 300. We used Adam Optimizer with 5.0e-5 learning rate for the BERT parameters and 1e-4 for all other parameters from the Graph Transformer and the final MLP layer. We used a warm-up step of 150 and batch size 16 for the WikiTables dataset, and we used a warm-up steps of 1000 with batch size of 4 for WebQueryTables. We trained the model for five epoches on each dataset, and we adopted Cross Validation for WikiTables given that there are only 60 queries in this dataset, while we used standard train-val-test split for WebQueryTables. The loss function used for WikiTables is Mean Squared Error, as the relevance scores are 0, 1, and 2, while we used Cross Entropy Loss for WebQueryTable, as ground truth is one relevant table. The training on WikiTables took 4 hours and is done on Google Colab with a P-100 GPU. and the training on WebQueryTable took 23 hours and is done on on a single Tesla T4 GPU.

Evaluation Metric and Result During inference, the system retrieves tables based on both the table cells and contextual information as shown in Figure 1. For each query, we generate a ranked list of table ids by sort all candidate tables directly using the predicted relevance scores on each dataset. The set of metrics used for WikiTables is NDCG@k, where $k = 5, 10, 15, 20$ and Mean Average Precision, as for each query, there are multiple tables annotated with high relevance score from the dataset. For WebQueryTable, we used Top 1 Precision(P@1) and Mean Average Precision. Both the NDCG scores at different cutoff points and Mean Average Precision score are computed using the TREC evaluation tool.

Result We report our reproduction result of the GTR system along with the original authors' results 1. We had trouble reproducing the results in (Wang et al., 2021) for WikiTables (we reported the average result from 5 rounds of experiments, this may potentially be due to randomness in cross validation and small data size. On the other hand, we sucessfully reproduced the results of GTR on WebQueryTable. We borrow results from previous baselines BERT4TR (Chen et al., 2020), and TaBERT reported from the GTR paper(Wang et al., 2021) for performance comparison.

	P@1	MAP
BERT4TR	-	0.7104
TaBERT	0.5067	0.6338
Our reproduction results	0.6239	0.7339
(Wang et al., 2021) result	0.6257	0.7369

Table 2: The performance (accuracy) of baselines on WebQueryTable

Error Analysis for Baseline

- There are certain cases that the model over-evaluate the text similarity between query and table title but under-evaluate the actual content of the table. For example, for query ”fast car”, the model gives the table ’Fast Cars and Superstars: The Gillette Young Guns Celebrity Race’ highest rank. Although the title explicitly contains the text ”fast car”, the table is all about racer's performance rather than the cars. Similarly, for query ”infections treatment”, the tables it ranks the highest contains information about the diseases and bacteria without information about treatment. We believe these errors are due to weak tabular content modelling given a query that is a high-level description so that the table content has very few direct connections to the query.

- We notice that the model tends to perform better in larger sized tables and it tends to give lower correlation scores for smaller tables. First, among the top ranked tables for each query, the correctly predicted rate is higher for larger tables as shown in the figure. Additionally, for small sized table (bottom 25%),

	NDCG@5	NDCG@10	NDCG@15	NDCG@20	MAP
BERT4TR	0.6052	0.6171	0.6386	0.6689	0.6191
TaBERT	0.5926	0.6108	0.6451	0.6668	0.6326
Our GTR reproduction results	**0.6347**	**0.6519**	**0.6821**	**0.6977**	**0.6524**
GTR result from (Wang et al., 2021)	**0.6554**	**0.6747**	**0.6978**	**0.7211**	**0.6665**

Table 1: GTR Baseline Reproduction Results on Wikitable dataset

only about 64.28% of the high correlated table query pairs are predicted correctly, while for large size tables (top 25%), the number rise to 76.71 %.

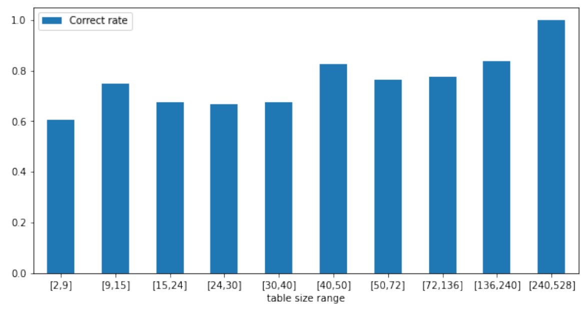

Figure 2: Correct rate of different table sizes

4 Approach

For the GTR model, we find two sources of error: (1) model over-evaluating to the text similarity between query and table context but under-evaluate the actual content of the table on small tables (2) Insufficient language capabilities for abbreviation and domain words. Thus, for our project, we proposed a generic content snapshot based approach to customize tabular graph construction based on each query to improve the query-graph matching module.

4.1 Analysis of GTR Tabular Graph Construction

Given that tables rich semi-structured data, we believe that a rich representation of the tabular content must include the following semantic relations.

- **Table Header - Cell Relation :** The headers defines name entities/class-instance the cells under the header belongs to, from a relational standpoint, it is a is-instance relation.

- **Cell-Cell Relation** Cells in each row and columns are semantically related.

- **Entity-Attribute Relation across headers:** The headers defines name entities, and each cell node is an attribute of the entity given the header node.

The graph construction in original GTR paper is simplified to two types of connections, 1) connection between only adjacent table cells 2) connection between unit(row, column) and its subunit(table cell), as shown in Figure 3.

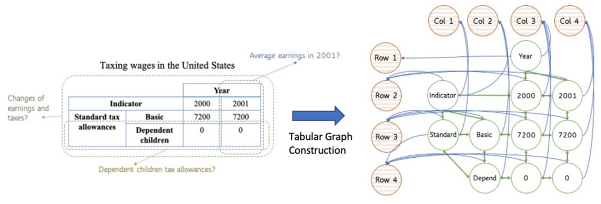

Figure 3: GTR Tabular Graph taken from (Wang et al., 2021)

This formulation is essentially a pair-wise MRF representation similar to image representation. We believe that it satisfies the first relation, and partially satisfies the second relation, as normally performing row switch and column switch should not change the information within the table. However, in the current graph construction, performing row/column switch would change the graph representation, though we do not believe this will significantly hurt result. And finally the third relation is mostly ignored. Therefore, we first performed experiment with fully connected row and column in the tabular graph construction module. Furthermore, we designed a process to find the most relevant columns, Core Columns, in a table with the query based on content salience measure, then connect the cells in the core column with every other cell in the same row, and fully connect every cell in the core column. We believe that this process best matches the process behind how a human finds content in a table for a given query.

Salience Measure We use cosine similarity to computed the final similarity score between a column and the query, and we experimented with two schemas for computing the embedding of the columns:

- **Similarity of Average Embedding** Use average embedding of the header and the cell contents to represent each column:

$$SAL_(c) = cosine(\frac{\sum_{cell \in c} v_{cell}}{len(c)}, \frac{\sum_{w \in q} v_w}{len(q)}),$$

- **Max similarity** Use the max cosine similarity between query and item in each column.

$$SAL_(c) = max_{cell \in c, w \in q} cosine(v_{cell}, v_w)$$

Tabular Graph Construction Our core graph construction would results in a tabular graph for a query: ”CMU student count”, as shown in Figure 4.

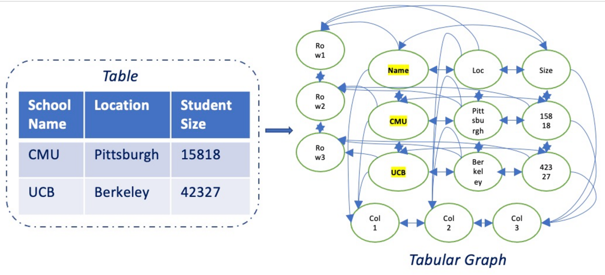

Figure 4: Our Tabular Graph Construction

Based on the query, the core column selected by using the highest salience measure should be the ”Name” column, hence for each cell in the ”Name” columns, we will add pairwise connection for the cell and every other cell in the same row.

Other Approach We also tried dual encoder based approach to encode the table contents(Caption and Table Graph Representation) and the query separately, and project each representation to same dimension and concatenated together with their element-wise subtraction and Hadamard product, as this has shown to be a way to perform embedding interactions. However, the result is 5-8% worse on validation set for WikiTables across metrics and appears to be over-fitting significantly to training data, in the future, if we have additional resources, we hope to spend more time focusing on different strategies of hard negative mining, and improve dual encoder dense retrieval model training, as it is evident that dual-encoders are harder to train compared to single encoder, but offers great benefits for efficient inference.

Experiment Setting The hyper-parameter settings are kept the same as the GTR baseline for consistency and we used the same machines to do training. For WikiTables, we experimented with both the fully connected graph approach, and content snapshot for core column selection with number of columns k = 1, 3, 10. And we only had enough budget to experiment with the original GTR result and core column selection with number of columns k = 10 on WebQueryTable, as the query table preprocessing step becomes a major bottleneck in computation since the original GTR method of pairwise MRF connection or the fully connected schema means query and tabular graph is separate, so only 273816 tabular graphs to construct; while in our case we have to generate tabular graph for each table given each query, which is 5*10e9 tabular graphs, and would take approximately 4 days on a single machine. This approach would be more plausible for either smaller data size or parallelize the graph constructions to more workers.

Results We compare our results with proposed methods for tabular graph construction with the results from our reproduction of GTR system on both datasets in 3 and 4. The result using core column selection outperforms the simple pairwise heuristics used of pairwise connection cell nodes on both datasets, and from our experiment on WikiTables dataset, we find that Core Columns Selection with k = 3 produces the best results, and this is likely due to better handling of long queries with multiple entities in the query. We will discuss errors of the current model through our ablation study.

	P@1	MAP
Our reproduction results	0.6239	0.7339
Core Col Selection with k = 10	**0.6305**	**0.7431**

Table 4: Comparative Results on WebQueryTable

Ablation Study and Analysis

- We first want to measure if query length affect the precision of the prediction. We use number of words to represent query length and the rank of the correlated table to measure the precision shown in5.

	NDCG@5	NDCG@10	NDCG@15	NDCG@20	MAP
Our GTR reproduction results	0.6347	0.6519	0.6821	0.6977	0.6524
Fully-Connected Graph Tabular Graph	0.6357	0.6582	**0.6871**	0.6989	0.6642
Core Columns Selection with k = 10	0.6342	0.6564	0.6801	0.7004	0.6568
Core Columns Selection with k = 3	**0.6452**	**0.6452**	0.6857	**0.7110**	**0.6653**

Table 3: Comparative Results on Wikitable dataset

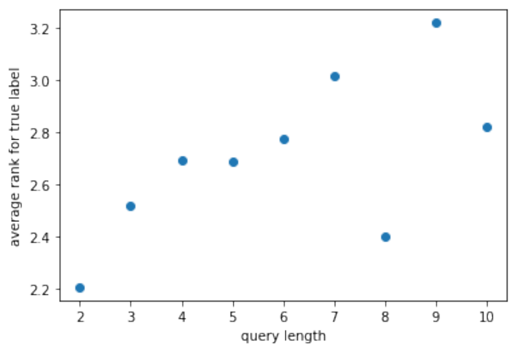

Figure 5: Average true label rank for different query length

College	SAT Math	SAT Read
Indiana University Bloomington	510-620	540-600
Indiana University Pennsylvania	450-540	440-530
Iowa State University	530-680	460-620
Ithaca College	530-640	520-630

Figure 6: WebQuery web15198: Colleges Average SAT amp; ACT Test Scores - PowerScore

The correlation between the query length and the average rank is 0.64. Given the large sample size, the number gives sufficient reason to believe that long query performs poorer than short queries. Intuitively, this make sense because long queries often contains more information with more logic, which makes it harder to find the related table and entry.

- Furthermore, we take a closer look at the queries that performs poorly in the current model and we attribute the errors to three main reason. The first reason is insufficient language understanding. The first example is abbreviations. For example, in the WebQueryTable dataset web18319, the query is ”state bank of india savings account interest rate”, while in the caption of the corresponding table ”state bank of india” is abbreviated as ”SBI”. The model cannot match SBI with ”state bank of india” and consequently gives higher relevance score to tables with the word ”India” in it.

  Another typical insufficient understanding example is professional words. For example in WebQueryTable web3569, the query contains the phrase ”hydrogen chloride”, while the correlated table uses the word ”Hydrochloric acid”. In WebQueryTable web7842, the query asks ’100 g butter to cups’, while the correlated table gives answer for all recipes.

- The second major problem is under-evaluate the correlation between table entries. For example, in WebQueryTable web13598, the query is ”stat holiday in march”. Only the correct table contains March holidays, however, the model gives it a low relevance score probably some irrelevant information in the caption. This was a big issue for the original GTR model, the problem is slightly alleviated with our tabular graph construction method

- The last major problem is an unsufficient understanding of table structure, especially in an ”A of B” relation. For example, in WebQueryTable web15198, the query asks ’ithaca college sat scores’. If we take a look at the correlated table 6, we can see the word ”SAT” is in column title, while ”Ithaca” is in the table entry. In this table, the ”College” column act more as a row title role since the entries in the ”SAT Math” column represent ”SAT Math of some college”. However, in our graph construction, we only connect the entries with its own column node and own row node, which does not really include such relation.

- Another concern we had for the GTR architecture with our emphasis on the query-table matching module compared to linearized pre-trained language models is that that there will be a weaker table-text association due to semantic shifts between the feature space of the BERT language model and the graph encoder. However, we did not find this to be a major issue on the experiments that we performed and studied for the two datasets.

5 Conclusion and Future work

For our project, we extended upon the GTR framework by proposing a novel tabular graph construction heuristic to better handle for complex table retrieval. The experimental results showed that this is a promising direction, as the input graph is critical towards the effective graph representation learning through graph neural network. Furthermore, through our ablation study, we found that the tabular graph construction module can be further improved by connecting core column cells to other header cells to hadnle "A of B" relations and a stronger textual encoder would also potentially improve results for case where the BERT base LM struggled with. Also, we discovered that new graph neural network architecture can handle different edge types, we hope to extend our tabular graph construction module by explicit create different edges to model the three types of semantic relations and supplement FastText embedding with RDF embeddings (Vandewiele et al., 2020), as tabular data contains mostly different entities types. There are also great opportunities in extending the improved architecture to other table-text tasks such as table summarizatio, and OCR tasks for table and cell structure

References

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *The Semantic Web - ISWC 2015*, pages 425–441, Cham. Springer International Publishing.

Michael J Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proceedings of the VLDB Endowment*, 2(1):1090–1101.

Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. 2020. Table search using a deep contextualized language model. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training.

Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open domain question answering over tables via dense retrieval.

Jonathan Herzig, Pawel Krzysztof Nowak, Francesco Piccinno Thomas Müller, 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. Association for Computational Linguistics.

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

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality.

Gilles Vandewiele, Bram Steenwinckel, Terencio Agozzino, Michael Weyns, Pieter Bonte, Femke Ongenae, and Filip De Turck. 2020. pyRDF2Vec: Python Implementation and Extension of RDF2Vec. IDLab.

Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. In *37th International Conference on Very Large Data Bases (VLDB)*. Stanford InfoLab.

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

Zhao Yan, Duyu Tang, Nan Duan, Junwei Bao, Yuanhua Lv, Ming Zhou, and Zhoujun Li. 2017. Content-based table retrieval for web queries.

Pengcheng Yin, Graham Neubig, Wen tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data.

Shuo Zhang and Krisztian Balog. 2018a. Ad hoc table retrieval using semantic similarity. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*.

Shuo Zhang and Krisztian Balog. 2018b. Ad hoc table retrieval using semantic similarity. *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*.