Retrieval-Based Transformer for Table Augmentation

Michael Glass[1], **Xueqing Wu**[2], **Ankita Rajaram Naik**[1], **Gaetano Rossiello**[1], **Alfio Gliozzo**[1]
[1]IBM Research AI, Yorktown Heights, NY, USA
[2]University of Illinois Urbana-Champaign

Abstract

Data preparation, also called data wrangling, is considered one of the most expensive and time-consuming steps when performing analytics or building machine learning models. Preparing data typically involves collecting and merging data from complex heterogeneous, and often large-scale data sources, such as data lakes. In this paper, we introduce a novel approach toward automatic data wrangling in an attempt to alleviate the effort of end-users, e.g. data analysts, in structuring dynamic views from data lakes in the form of tabular data. We aim to address *table augmentation* tasks, including row/column population and data imputation. Given a corpus of tables, we propose a retrieval augmented self-trained transformer model. Our self-learning strategy consists in randomly ablating tables from the corpus and training the retrieval-based model to reconstruct the original values or headers given the partial tables as input. We adopt this strategy to first train the dense neural retrieval model encoding table-parts to vectors, and then the end-to-end model trained to perform table augmentation tasks. We test on EntiTables, the standard benchmark for table augmentation, as well as introduce a new benchmark to advance further research: WebTables. Our model consistently and substantially outperforms both supervised statistical methods and the current state-of-the-art transformer-based models.

1 Introduction

The way organizations store and manage data is rapidly evolving from using strict transactional databases to data lakes that typically consist of large collections of heterogeneous data formats, such as tabular data, spreadsheets, and NoSQL databases. The absence of a unified schema in data lakes does not allow the usage of declarative query languages, e.g. SQL, making the process of data preparation[1] dramatically expensive (Terrizzano et al., 2015).

Data preparation involves several phases, such as data discovery, structuring, cleansing, enrichment and validation, with the purpose of producing views commonly organized in a tabular format used to create reports (Koehler et al., 2021) or to gather feature sets to build machine learning models (He et al., 2021). The schemaless nature of data lakes makes data discovery and structuring even more challenging since the tasks of joinability and unionability among tables become non-deterministic (Fernandez et al., 2018; Zhu et al., 2019; Bogatu et al., 2020).

In this work, we propose a novel end-to-end solution based on a retrieval augmented transformer architecture with the aim to support end-users, such as data analysts, in the process of constructing dynamic views from data lakes. To this end, we address three table augmentation tasks (Zhang and Balog, 2017, 2019): automatic row and column population and cell filling (or data imputation).

Figure 1 illustrates the three core tasks in table augmentation. All tasks proceed from a query or seed table. In the case of self-supervised training, this seed table is formed by ablating rows, columns or cell values from an existing table in the data lake. The task of column header population, also simply called column population, is to extend the table with additional possible column names or headers. This is a way of suggesting additional data that could be joined into this table. In the task of cell filling there is a specific unknown cell, for which the model predicts a specific value. The task of row population is only populating the *key column* for a row. This is the column that contains the primary entity that the remainder of the row contains data for, sometimes referred to as a row header. Typically this is the first column in a table.

Approaches to table augmentation can be purely parametric (Iida et al., 2021; Deng et al., 2022), in which case the data lake is used to train the param-

[1]Also referred as data wrangling or data munging.

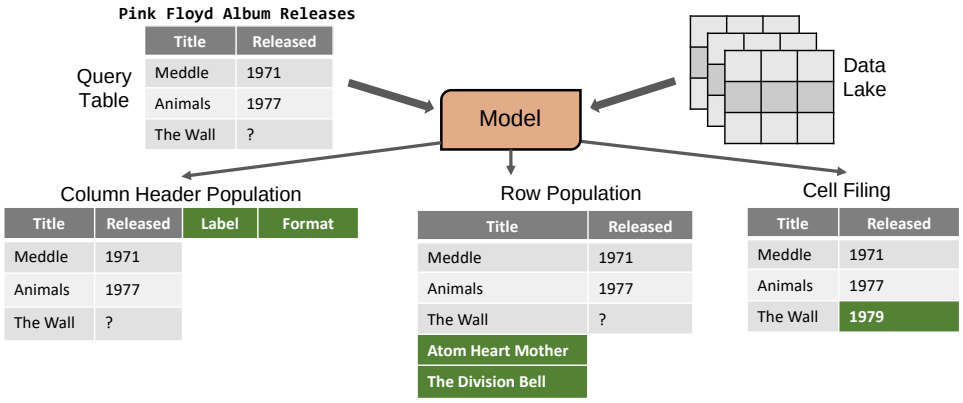


Figure 1: Given a partially completed table as a query (i.e. a few album releases from the Pink Floyd discography), the three table augmentation tasks consist of retrieving from the data lake: 1) a list of possible next column headers, such as the "Label" or "Format", 2) the missing value "1979" for the release date of the row "The Wall", 3) a list of other album releases as possible next rows, such as "Atom Heart Mother" and "The Division Bell".

eters of the model, but not used during inference. In this setting, the table augmentation model must draw the possible augmentations for rows, columns and cells from its trained parameters. Alternatively, with retrieval-based models (Lewis et al., 2020b; Glass et al., 2021b, 2022), the data lake can also be used at inference to provide evidence for proposed augmentations. This has two key advantages: 1) the model need not memorize the data lake – or even a significant fraction of it, and 2) the model can provide justification for its predicted augmentations in the form of a provenance table or tables.

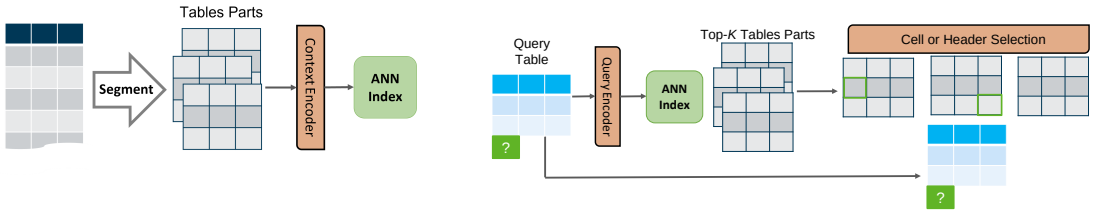
The key contributions of this paper are: (1) We introduce the first end-to-end, retrieval-based model for table augmentation. Our Retrieval Augmented Table Augmentation (RATA) model uses a biencoder retrieval model for neural indexing and searching tables from data lake, and a reader transformer to identify augmentations from retrieved tables. (2) Our model establishes a new state-of-the-art across all three tasks in table augmentation, while also providing additional value with its provenance information. (3) We create and release a new dataset for table augmentation, expanding the scope of evaluation beyond Wikipedia. This dataset, based on Cafarella et al. (2008), is also larger and more diverse than the standard Wikipedia-based dataset (Zhang and Balog, 2017).

2 Related Work

Table augmentation can be divided into three sub-tasks: row population, column population, and cell filling. For row and column population, Zhang and Balog (2017) identifies and ranks candidate values from both the table corpus and knowledge base. Table2Vec (Zhang et al., 2019a) trains header and entity embeddings from a table corpus in a skip-gram manner and uses the embeddings for the task. Although TaBERT (Yin et al., 2020) was developed as a foundational model primarily for question answering, its embeddings have also been applied for row and column population. Recent work formulates the task as multi-label classification and fine-tunes large-scale pre-trained models such as TABBIE (Iida et al., 2021) and TURL (Deng et al., 2022).

TABBIE consists of three transformers for converting cells, columns and rows to vector representations. A corrupt cell detection task is the pre-training task used to learn these embeddings on the table corpus. To fine-tune a trained TABBIE model for the column header population task, a concatenated [CLSCOL] embedding of the columns is passed through a single linear and softmax layer and trained with a multi-label classification objective. Similarly, for the row population task a multi-class classification is carried out on the first column's [CLSCOL] representation.

For cell filling, InfoGather (Yakout et al., 2012) retrieves tables from the table corpus and selects values from retrieved tables. Zhang and Balog (2019) extends the system to retrieve from both the table corpus and knowledge base. Their system that uses only the table corpus as the source is called TMatch, which we compare to in Section 6. Ahmadov et al. (2015) combines predictions both from table retrieval and from a machine learning-based value imputation system. Deng et al. (2022)



(a) Building the table-part index.

(b) Architecture of RATA.

Figure 2: Index building and inference system overviews

directly applies pre-trained TURL model to the task since cell filling is similar with its pre-training objective. Cell filling is also related to the task of value imputation, i.e., to provide an assumed value when the actual value is unknown, usually using machine learning methods (Bießmann et al., 2019). In addition to augmenting individual entities, column headers or cells, some other work aims to join tables over entire rows or columns with retrieved tables (Sarma et al., 2012; Bhagavatula et al., 2013; Lehmberg et al., 2015).

Retrieval-augmented models have been successfully applied to many tasks. For open-domain question answering (ODQA), DPR learns dense representation to retrieve evidence and trains a separate reader to select answer from retrieved evidence (Karpukhin et al., 2020). RAG uses a generator to generate outputs conditioned on retrieved evidence and jointly trains DPR with a generator on the downstream task (Lewis et al., 2020b). RAG is shown to achieve good performance on knowledge-intensive NLP tasks such as ODQA, fact verification, slot filling, etc (Lewis et al., 2020b; Petroni et al., 2021). Re^{2}G further introduces a reranker to boost performance (Glass et al., 2022). Retrieval-augmented models are also shown to be effective on zero-shot slot filling (Glass et al., 2021b), and multilingual keyphrase generation (Gao et al., 2022). Similar models have also been applied to table-related tasks such as open-domain table question answering (Herzig et al., 2021). In our work, we apply the architecture to table augmentation.

3 Approach

While the row, column, and cell predictions of purely parametric table augmentation methods may be useful on their own, they can be much more effective for a human-in-the-loop use case if they are supported by provenance. A user of a data preparation application may be unwilling to simply accept the prediction of a model, but when paired with evidence from the data lake, that prediction can be better assessed. Furthermore, the retrieval model itself may be useful for exploration and general search in a data lake. In this view, table augmentation can be seen as self-supervised pretraining for table retrieval.

Fortunately, there is now considerable work on *retrieval augmented* transformer models (Glass et al., 2022; Lewis et al., 2020b). These models augment the parametric knowledge of the transformer, with non-parametric knowledge in the form of an indexed corpus. To do so, they use a neural retrieval model based on DPR (Dense Passage Retrieval) (Karpukhin et al., 2020) that is trained end-to-end to assist in generation.

We build on this line of research to introduce a general model for all table augmentation tasks: row population, column header population and cell filling. Our model, Retrieval Augmented Table Augmentation (RATA), comprises of an index of tables, a retrieval component, and a reader or selection component. The table index is built from the tables in the training set, which are first decomposed into table-parts, then transformed into sequences for use with standard retrieval approaches. The retrieval component is a biencoder architecture similar to DPR (Karpukhin et al., 2020), but trained without ground truth on correct provenance. We call this *Dense Table Retrieval* or DTR. The reader component is an extractive approach. An extractive rather than generative approach ensures that the model's predictions are always grounded in actual data, rather than speculative guesses. The extractive approach is also a more natural fit for row and column population tasks, where there is no required order to the answers. Finally, the extractive approach permits an initial training phase for the retrieval component where the *answer-bearing* tables are considered as a bag of positives.

Figure 1 illustrates the tasks of table augmentation by example. Formally, the input I is a table

with r rows and c columns comprising a caption $\mathcal{C}$, headers $\mathbf{H}$, and matrix of cell values, $\mathbf{V}$. One of the columns, usually the first, is indicated as the key column key.

$$I = \langle \mathcal{C}, \mathbf{H}, \mathbf{V}, key \rangle, 1 \leq key \leq c$$
$$\mathbf{H} = [h_1, h_2, ..., h_c]$$
$$\mathbf{V} = \begin{bmatrix} v_{1,1}, v_{1,2}, ..., v_{1,c} \\ ... \\ v_{r,1}, v_{r,2}, ..., v_{r,c} \end{bmatrix}$$

The input table is ablated in a task specific way to produce a query table and gold answers, $\langle Q, \mathbf{G} \rangle$, described as follows:

$$Q_{rp} = \langle \mathcal{C}, \mathbf{H}, \mathbf{V}_{\mathbf{1..n_{seed}}}, key \rangle$$
$$\mathbf{G}_{rp} = \{\mathbf{V}_{i,key} : i > n_{seed}\}$$
$$Q_{cp} = \langle \mathcal{C}, \mathbf{H}_{\mathbf{1..n_{seed}}}, \mathbf{V}_{..,\mathbf{1..n_{seed}}}, key \rangle$$
$$\mathbf{G}_{cp} = \{\mathbf{H}_i : i > n_{seed}\}$$
$$Q_{cf} = \langle \mathcal{C}, \mathbf{H}, \mathbf{V} \setminus \mathbf{v_{i,j}}, key \rangle$$
$$\mathbf{G}_{cf} = \{v_{i,j}\}$$

where *rp*, *cp* and *cf* refer to the row population, column header population and cell filling tasks, respectively.

3.1 End-to-End Model

Figure 2a shows how tables in a data lake are first indexed to provide a non-parametric knowledge store. Each table is first split into chunks of up to three rows plus the header, which we refer to as *table-parts*. We form sequence representations of these table-parts following work in other transformer-based approaches to tables (Glass et al., 2021a). The table-part sequence representations (S^t) are formed from the row sequence representations (S^r_i) and the table caption:

$$S^r_i = \bigoplus_{j=1}^{c} h_j \oplus \text{':'} \oplus v_{i,j} \oplus \text{'*'}$$

$$S^t = \mathcal{C} \oplus \text{[SEP]} \oplus \bigoplus_{i=start}^{end} S^r_i \oplus \text{'|'}$$

Here $\oplus$ indicates concatenation and the strings ':', '*', and '|' delimit the header, cell value contents, and each row respectively. Any distinctive tokens can work as delimiters since the transformer will learn an appropriate embedding representation.

These sequences are then projected to vectors using the context encoder by taking the [CLS]. We index the dense representations for all table-parts in the data lake using FAISS (Johnson et al., 2017) with Hierarchical Navigable Small World (Malkov and Yashunin, 2018).

Figure 2b shows the architecture of our approach, Retrieval Augmented Table Augmentation (RATA). The input query is encoded to a vector for retrieving related table-parts from the indexed data lake. Similar to table-part representation, we form sequence representation for the query, use a query encoder to encode it, and take the [CLS] vector as query representation. Both the context encoder and the query encoder use the $\text{BERT}_{\text{BASE}}$ architecture. We use unnormalized dot product to score a pair of query q and table-part d. Top-k table-parts with highest scores will be retrieved.

$$score(q, d) = \text{BERT}_{qe}(q)_{[CLS]} \cdot \text{BERT}_{ce}(d)_{[CLS]}$$

After the top-k most relevant table-parts are retrieved, the reader component selects the most likely augmentations for the query table. In the case of column population, the candidate augmentations are all headers from retrieved table-parts; for cell filling it is all cells; and for row population it is only those cell values that are entities.

The sequence representation of the query table is paired with each table-part representation, using the standard [CLS] and [SEP] token to demarcate the bounds of each sequence. In the table-part representation, the candidates are marked by special begin and end tokens: '⟨' and '⟩'. This combined sequence is then the input to a transformer encoder (initialized from $\text{BERT}_{\text{LARGE}}$ (Devlin et al., 2019)). For each pair of candidate answer marks ('⟨' and '⟩'), the final token embeddings are concatenated to produce a single vector. Then a linear layer is applied to predict the likelihood that the candidate is a correct answer to the query.

$$\alpha = [i : t_i = \text{“⟨”}]$$
$$\omega = [i : t_i = \text{“⟩”}]$$
$$ans_n = t_{\alpha_n+1}, t_{\alpha_n+2}, ..., t_{\omega_n-1}$$
$$C = \begin{bmatrix} E_{\alpha_0} \oplus E_{\omega_0} \\ E_{\alpha_1} \oplus E_{\omega_1} \\ E_{\alpha_2} \oplus E_{\omega_2} \\ ... \end{bmatrix}$$
$$\rho = softmax(C \cdot \mathbf{w_{candidate}})$$

Formally, the input is a sequence of tokens $T = [t_0, t_1, ...]$. The transformer encoder produces a sequence of embeddings $BERT_{reader}(T) =$

$E = [e_0, e_1, ...]$. The candidate representation vectors, C, are then multiplied by the learned parameter vector $\mathbf{w_{candidate}}$ and a softmax is applied to produce the reader scores, ρ, for the retrieved table-part.

Note that the likelihood for a given answer occurrence ans_n is ρ_n. The candidate likelihood vectors for each of the top-k retrieved table-parts, $\rho^1, \rho^2, ..., \rho^k$, are then combined with the softmax normalized retrieval scores, $\mathbf{r} = [r_1, r_2, ..., r_k]$, to provide a probability distribution over all candidates in all retrieved table-parts. Since these scores are for each occurrence of a candidate string, we aggregate over each distinct normalized candidate string by summing the likelihoods for all occurrences. This produces the final score, $s(a)$ for each answer string a. The loss is the negative log-likelihood of all gold answer strings, $\mathbf{G}$. Because of this formulation, during training any instance with no correct candidates in any retrieved table-part is skipped.

$$\mathbf{p}^j = softmax(\mathbf{r})_j \cdot \rho^j$$
$$s(a) = \sum_{j=1}^{k} \sum_{n:ans_n^j = a} \mathbf{p}_n^j$$
$$loss = -\sum_{a \in \mathbf{G}} log\left(s(a)\right)$$

We use answer normalization to determine if a candidate matches a gold answer, as described in Appendix B. For row population and cell filling in EntiTables, the cell values are already linked to entities so normalization is not necessary.

For RATA training, we iterate through the tables in the training set. To construct input query from a table, we ablate either all rows after the first n_{seed} (row population), or all columns after the first n_{seed} (column population), or a particular cell (cell filling). We ensure that table-parts from the query table are not retrieved by filtering the retrieved results. Like most previous approaches to end-to-end training of neural retrieval, we train only the query encoder in the end-to-end training phase. This avoids expensive re-indexing of the entire data lake either each time the context encoder is updated, or periodically as in ANCE (Xiong et al., 2020).

3.2 Retrieval Training

While it is possible in theory to train neural retrieval entirely through impact in the end-to-end table augmentation tasks, a good initialization is important for learning. Without an initial effective retrieval model, there is no answer-bearing evidence to train the reader model, and therefore a high fraction of training examples will be skipped (Lee et al., 2019).

One possible approach is to use a pretraining task for retrieval, such as the Inverse Cloze Task (Lee et al., 2019) or a retrieval-based masked language model (Guu et al., 2020). In the table augmentation task, there is the option of training with answer-bearing evidence as positives. Since the reader is purely extractive, any evidence that does not contain a correct augmentation string is necessarily a negative. However, not every table-part that contains an answer is a positive. We use a multiple instance learning setup for the positives: we train under the assumption that at least one of the table-parts containing a correct answer is a positive.

To gather the training data for retrieval we build an initial keyword index using Anserini[2]. We use BM25 (Robertson and Zaragoza, 2009) to retrieve potentially relevant table-parts for each table query.

From each training table we construct a query for row population, column population or cell filling. Since these queries are constructed from ablated tables, we know a (potentially incomplete) set of correct augmentations or answers. Note that there may be other equally correct augmentations. But since this is a self-supervised task, we consider only the headers or cell values that actually occurred in the table to be correct.

Formally, the query constructed from a training table is a pair of the ablated table, Q and the set of gold answers $\mathbf{G}$. The set of table-parts retrieved by the initial retrieval method, for example BM25, is given as $\mathbf{R}$. A retrieved table-part is in the positive set, $\mathbf{R}^+$, if it contains any gold answer, otherwise it is a hard negative, $\mathbf{R}^-$.

$$\mathbf{R}^+ = \{d : d \in \mathbf{R} \wedge \exists a \in \mathbf{G}, a \in d\}$$
$$\mathbf{R}^- = \mathbf{R} - \mathbf{R}^+$$

Following Karpukhin et al. (2020), we use batch negatives along with the retrieved "hard negatives". The batch $B = [\langle q_1, \mathbf{R}_1\rangle, \langle q_2, \mathbf{R}_2\rangle, ..., \langle q_{bz}, \mathbf{R}_{bz}\rangle]$ is processed to produce vectors for all queries and retrieved table-parts. All query vectors are multiplied with all table-part vectors to produce scores between all pairs. A softmax is applied per-query to give the normalized scores. Finally, the loss is

[2] https://github.com/castorini/anserini

the negative log-likelihood for the positive scores.

$$\mathcal{R} = \bigcup_{i=1}^{bz} \mathbf{R}_i$$
$$\rho_i = softmax([score(q_i, d) : d \in \mathcal{R}])$$
$$loss = -\sum_{i=1}^{bz} log \left(\sum_{d \in \mathbf{R}_i^+} \rho_{i,d} \right)$$

Note that since we are summing over the probability of all table-parts in the positive set, $\mathbf{R}^+$, it is not necessary for *all* answer-bearing retrieved table-parts to be high scoring. Instead, it follows the multiple instance learning framework. All instances marked negative are negative, while at least one instance in the positive set is positive.

4 WebTables Dataset

Prior work on table augmentation has focused on tables derived from Wikipedia (Zhang and Balog, 2017; Iida et al., 2021; Deng et al., 2022; Zhang and Balog, 2019; Zhang et al., 2019b). In order to better assess the proposed methods and provide the research community with a new benchmark, we introduce a new dataset for table augmentation: WebTables.

We construct this dataset using the tables crawled and extracted by Cafarella et al. (2008). We start from the English relational tables of WDC Web Table Corpus 2015. We further filter the dataset to remove the most common types of noisy tables: calendars formatted as tables, lists of forum posts and torrent links, tables with less than four rows or columns, and tables that format large blocks of text. Because previous work on table augmentation focused so heavily on Wikipedia tables, we exclude from this dataset any tables crawled from any "wikipedia" domain. We also deduplicate the corpus, ensuring that there are no two tables with the same content in their cells.

Following filtering and deduplication we sample 10 thousand tables each for the development and test sets and one million tables for training. However, in our experiments we use only 300 thousand training examples to limit the computational cost.

To parallel the setting of EntiTables we use the "key column" identified by Cafarella et al. (2008) as the target column for row population and we consider entities to be those strings that occur at least three times in the key column for any table in the train set.

5 Experiments

We experiment on two datasets of tables across three tasks. Table 1 gives statistics on these datasets.

EntiTables (Zhang and Balog, 2017) contains 1.6M tables collected from Wikipedia where entity mentions are normalized into its name in DBPedia. For row and column population, we use the development and test sets released by Zhang and Balog (2017) each containing 1,000 randomly sampled queries. For cell filling, we use the test set released by Zhang and Balog (2019). The test set contains 1,000 queries uniformly sampled from four main column data types: entity, quantity, string, and datetime. Though Zhang and Balog (2019) use human annotations as gold labels, we notice that the human annotations are of low quality, so we use the original values in the table cells as gold labels.

WebTables is based on Cafarella et al. (2008) – 154M relational tables extracted from HTML tables in Common Crawl. We process the corpus as described in Section 4. For column population we use the original development and test sets of 10,000 tables each. While for row population we necessarily exclude any tables without any entities in the key column after the first n_{seed} rows. For cell filling, we use heuristic rules to classify cell values into three types: quantity, string and datetime. Then, we sample 3,000 queries uniformly from the three types as test set and sample another 3,000 queries as development set.

Dataset	Task	Train	Dev	Test
EntiTables	row pop.	187k	1k	1k
EntiTables	column pop.	602k	937	950
EntiTables	cell filling	100k	-	972
WebTables	row pop.	563k	6.6k	6.8k
WebTables	column pop.	1M	10k	10k
WebTables	cell filling	1M	3k	3k

Table 1: Dataset statistics.

We compare our method with two deep learning-based baselines, TABBIE (Iida et al., 2021) and BART (Lewis et al., 2020a). Both TABBIE and BART have no retrieval component involved.

TABBIE, described in Section 2, uses three transformers: one for cell values, one for rows, and one for columns. It produces vector embeddings for each cell and each row and column of a table. We follow Iida et al. (2021) for the row and column population and base our experiments on the par-

tial released code and pretrained model[3]. To apply TABBIE to cell filling, we formulate it as classification on the concatenation of the row and column embedding vectors, similar to row and column population. The classification vocabulary is collected from the training corpus: all cell values that occur at least ten times. We also report the published results for TABBIE on the EntiTables dataset, although we were unable to reproduce these results for row population.

BART is a sequence-to-sequence model that takes the linearized table as the source text and generates the row entities, cell headers, or cell value as the target text. We use a beam search in decoding (beam size = 35) to produce a ranked list of predictions. We use the FAIRSEQ toolkit (Ott et al., 2019) for these experiments. For RAG we use the implementation in Hugging Face transformers (Wolf et al., 2019). For both BART and RAG, the sequence representation of the query tables is the same as in RATA.

On the EntiTables dataset, we also compare against probabilistic methods that first retrieve tables from the table corpus and next select values for table augmentation. We compare against the published results of Zhang and Balog (2017) for row and column population, and against TMatch (Zhang and Balog, 2019) for cell filling.

For evaluation, we report Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain over the top ten outputs (NDCG@10) for the final prediction performance of row population, column population, and cell filling. To evaluate the performance of DTR retrieval, we also report answer-bearing MRR, where a retrieved table-part is considered correct if it contains one of the correct answers. To determine the significance of these results we use a 95% confidence interval on the t-distribution. We also applied a sampling permutation test, but this did not change any conclusions regarding significance.

6 Results

Table 2 contains our results for the row population task. Our model, RATA, is able to greatly outperform all other methods on both datasets. Using the non-parametric knowledge of the table corpus is very advantageous for the large and specific vocabulary of entities in key columns.

[3]https://github.com/SFIG611/tabbie

	EntiTables		WebTables	
	MRR	NDCG	MRR	NDCG
TaBERT*	56.0	46.4	-	-
TABBIE*	57.2	47.1	-	-
TABBIE†	25.18	15.2	12.44	11.93
BART	45.30	32.76	29.25	19.30
RAG	56.95	43.48	33.20	22.23
RATA	**77.15**	**60.34**	**45.13**	**26.70**
	±2.32	±2.18	±1.10	±0.73

Table 2: Test results for row population, $n_{seed} = 2$.
* As reported in Iida et al. (2021) † Our results

	EntiTables		WebTables	
	MRR	NDCG	MRR	NDCG
TaBERT*	60.1	54.7	-	-
TABBIE*	62.8	55.8	-	-
TABBIE†	63.9	55.8	84.1	78.96
BART	73.36	65.37	87.40	85.05
RAG	78.64	72.81	89.39	87.58
RATA	**88.12**	**81.01**	**94.07**	**89.94**
	±1.91	±1.97	±0.44	±0.49

Table 3: Test results for column population, $n_{seed} = 2$.
* As reported in Iida et al. (2021) †Our results

	EntiTables		WebTables	
	MRR	NDCG	MRR	NDCG
TABBIE	10.62	11.56	24.79	26.17
BART	21.25	22.48	**37.06**	**39.19**
TMatch	30.54	32.23	-	-
RAG	18.65	19.71	34.80	36.34
RATA	**34.32**	**36.25**	33.58	35.33
	±2.80	±2.82	±1.60	±1.61

Table 4: Test results for cell filling.

Table 3 contains our results for the column population task. RATA is again substantially better than the other methods, although not by as wide a margin as the row population task. The BART baseline is the best performing of the alternatives with an MRR lower by 6% to 15%.

Results on cell filling task are in Table 4. Our method outperforms all baselines on both datasets. TABBIE performs the worst due to the large classification vocabulary and out-of-vocabulary issue. On EntiTables dataset, retrieval-based methods including TMatch and RATA significantly outperform non-retrieval methods including TABBIE and BART. Figure 3 shows an example output from RATA. On WebTables, however, BART outper-

Query table:

National Board of Review Award for Best Film 1940s

Year	Winner	Director(s)
1947	Monsieur Verdoux	?
1948	Paisan	Roberto Rosellini

Gold answer: Charlie Chaplin

Retrieved table:

List of Crime Films of the 1940s Notes

Title	Director	Cast	Country	Notes
Kiss of Death (1947 film)	Henry Hathaway	Victor Mature		Crime thriller
The Long Night (1947 film)	Anatole Litvak	Henry Fonda		Crime drama
Monsieur Verdoux	Charlie Chaplin	Charlie Chaplin		Crime comedy

Output answer: Charlie Chaplin

Figure 3: RATA example output on EntiTables dataset. The output answer is correct, and the retrieved table provides sufficient evidence for the answer.

	Row Population		Column Population		Cell Filling	
	EntiTables	WebTables	EntiTables	WebTables	EntiTables	WebTables
BM25	54.44±2.72	41.16±1.06	62.93±2.73	84.17±0.65	28.98±2.59	38.48±1.62
DTR (initial)	74.34±2.39	47.88±1.10	**90.07**±1.79	**94.91**±0.41	**34.78**±2.72	**40.80**±1.64
DTR (post-RATA)	**80.98**±2.17	**49.62**±1.11	**90.97**±1.72	**94.94**±0.41	**37.48**±2.81	**40.26**±1.66

Table 5: Retrieval answer-bearing MRR (%).

forms RATA. We notice that BART can achieve high scores by either copying values from other rows (as in Figure 5 and Figure 6a), or producing values similar with in other rows (as in Figure 6b and Figure 6c). As shown in the examples, this strategy is able to achieve good performance.

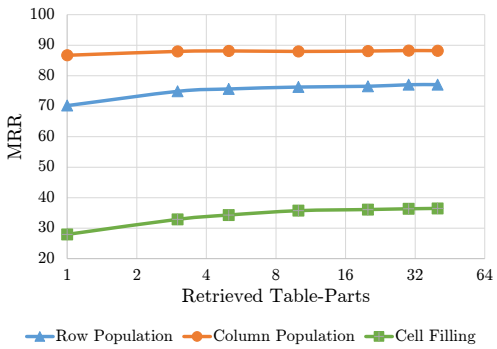


Figure 4: MRR gain as number of retrieved table-parts increases on the EntiTables dataset

Effect of Retrieval To analyze the effectiveness of the DTR component, we report answer bearing MRR in Table 5. We notice that DTR is well trained after the initial retrieval training phase and achieves higher answer bearing MRR compared to BM25. End-to-end training provides meaningful supervision for retrieval and further improves MRR on most tasks. By comparing Table 2, 3, 4 with Table 5, we notice that the final task MRR is close to answer bearing MRR. When the correct answer is present in the retrieved table, the reader can select the correct answer at high accuracy. This indicates that the bottleneck of our system is retrieval.

Number of Retrieved Table-Parts RATA was trained with 5 retrieved table-parts for all tasks. This relatively small number for the retrieval size provides good efficiency during training, since train time scales roughly linearly with the number of query / table-part pairs that must be processed by the reader transformer component. But during inference, we are able to adjust the number of retrieved table-parts more freely. Figure 4 shows that table augmentation performance monotonically increases as more evidence is retrieved for row population and cell filling, but column population performance does not improve past 5.

7 Conclusion

Our retrieval-based transformer architecture for table augmentation, RATA, is able to greatly advance the state-of-the-art in three table augmentation tasks: row population, column population, and cell filling. The non-parametric knowledge in the table corpus is able to substantially enhance the table augmentation capabilities. Furthermore, by training an effective table-to-table retrieval model we are able to provide provenance for the system's proposed augmentations. We also introduce a new benchmark dataset for table augmentation: WebTables and evaluate our model and two recent transformer baselines. Our code for RATA and the newly introduced dataset are available as open source[4].

[4] https://github.com/IBM/retrieval-table-augmentation

Limitations

A limitation of RATA is always assuming the answer is included in the retrieval corpus, which is not always true. When the corpus does not contain the correct answer, the desired behavior is to inform the user that the answer cannot be obtained, but RATA will provide a poorly supported answer. This also encourages RATA to learn spurious correlations when the retrieved tables coincidentally contain the same value, but does not really support the answer. This problem is especially serious when the answer is very generic (for example, numbers like "0") and same values by coincidence are common. This is related to the answerable question issue (Rajpurkar et al., 2018) or evidentiality issue (Lee et al., 2021; Asai et al., 2022) for question answering.

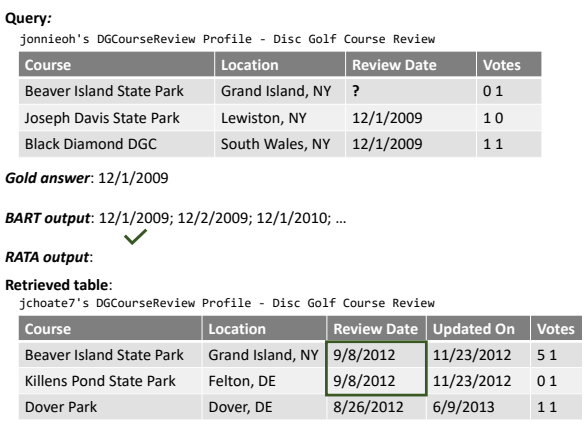


Figure 5: BART and RATA example outputs on WebTables.

For cell-filling on WebTables, BART outperforms RATA often by either copying values from other rows of the query table or producing values similar to those in other rows. However, as shown in Figure 5, RATA's retrieval is often not helpful. Usually, the information required to fill the query table is not repeated in the corpus, so the retrieved table cannot support the query. As a result, RATA is simply retrieving some similar table, and selecting similar values in the tables.

References

Ahmad Ahmadov, Maik Thiele, Julian Eberius, Wolfgang Lehner, and Robert Wrembel. 2015. Towards a hybrid imputation approach using web tables. In *2nd IEEE/ACM International Symposium on Big Data Computing, BDC 2015, Limassol, Cyprus, December 7-10, 2015*, pages 21–30. IEEE Computer Society.

Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2022. Evidentiality-guided generation for knowledge-intensive NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2226–2243, Seattle, United States. Association for Computational Linguistics.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA@KDD 2013, Chicago, Illinois, USA, August 11, 2013*, pages 18–26. ACM.

Felix Bießmann, Tammo Rukat, Philipp Schmidt, Prathik Naidu, Sebastian Schelter, Andrey Taptunov, Dustin Lange, and David Salinas. 2019. Datawig: Missing value imputation for tables. *J. Mach. Learn. Res.*, 20:175:1–175:6.

Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, pages 709–720. IEEE.

Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record*, 51(1):33–40.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1001–1012. IEEE Computer Society.

Yifan Gao, Qingyu Yin, Zheng Li, Rui Meng, Tong Zhao, Bing Yin, Irwin King, and Michael Lyu. 2022. Retrieval-augmented multilingual keyphrase generation with retriever-generator iterative training. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1233–1246, Seattle, United States. Association for Computational Linguistics.

Michael Glass, Mustafa Canim, Alfio Gliozzo, Saneem Chemmengath, Vishwajeet Kumar, Rishav Chakravarti, Avi Sil, Feifei Pan, Samarth Bharadwaj, and Nicolas Rodolfo Fauceglia. 2021a. Capturing row and column semantics in transformer based question answering over tables. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1212–1224, Online. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021b. Robust retrieval augmented generation for zero-shot slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1939–1949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715, Seattle, United States. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. Automl: A survey of the state-of-the-art. *Knowl. Based Syst.*, 212:106622.

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, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 512–519. Association for Computational Linguistics.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

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.

Martin Koehler, Edward Abel, Alex Bogatu, Cristina Civili, Lacramioara Mazilu, Nikolaos Konstantinou, Alvaro A. A. Fernandes, John A. Keane, Leonid Libkin, and Norman W. Paton. 2021. Incorporating data context to cost-effectively automate end-to-end data wrangling. *IEEE Trans. Big Data*, 7(1):169–186.

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

Kyungjae Lee, Seung-won Hwang, Sang-eun Han, and Dohyeon Lee. 2021. Robustifying multi-hop QA through pseudo-evidentiality training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6110–6119, Online. Association for Computational Linguistics.

Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. 2015. The mannheim search join engine. *J. Web Semant.*, 35:159–166.

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, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. 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, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for

knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2523–2544. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 817–828. ACM.

Ignacio G Terrizzano, Peter M Schwarz, Mary Roth, and John E Colino. 2015. Data wrangling: The challenging yourney from the wild to the lake. In *CIDR*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 97–108. ACM.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019a. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1029–1032. ACM.

Li Zhang, Shuo Zhang, and Krisztian Balog. 2019b. Table2vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1029–1032.

Shuo Zhang and Krisztian Balog. 2017. Entitables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 255–264.

Shuo Zhang and Krisztian Balog. 2019. Autocompletion for data cells in relational tables. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 761–770.

Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 847–864. ACM.

Appendix

A Model Hyperparameters

Our model is fine-tuned from two $\text{BERT}_{\text{BASE}}$ models for the retriever and one $\text{BERT}_{\text{LARGE}}$ model for the reader. This totals $2 \cdot 110M + 340M = 560M$ parameters.

Table 6 shows the hyperparameters used in our experiments.

Hyperparameter	DTR	Reader
learn rate	5e-5	3e-5
batch size	128	32
epochs	3	2
warmup instances	0	10%
learning schedule	linear	triangular
max grad norm	1	1
weight decay	0	0
Adam epsilon	1e-8	1e-8

Table 6: RATA hyperparameters

The only hyperparameter that varied for the tasks and datasets was the batch size.

B Dataset and Task Specifics

We use two types of answer normalization. For EntiTables column population we implement case-insensitive matching by normalizing both predictions and gold answers to lowercase. For all row

Dataset	Task	Batch Size
Entitables	All	32
WebTables	Row Population	32
WebTables	Column Population	32
WebTables	Cell Filling	64

Table 7: Batch size per task and dataset

and column population in WebTables we use a normalization that removes unicode accents and non-ASCII characters then lowercases. Cell filling does not use normalization.

For reproduction of results from TABBIE on Entitables we carry out the following steps.

Column Header Population Based on the above mentioned normalization we create a vocabulary of 182,909 column headers for the Entitables dataset which is approximately equal to the 127,656 possible header labels mentioned in the paper (Iida et al., 2021). Each of the possible headers occurs atleast twice in the training dataset.

Row Population Except for above mentioned normalization we use entities which have occurred atleast 7 times in the training dataset which lead to 308,841 possible entities. THis is approximately equal to the 300,000 entities mentioned in (Iida et al., 2021).

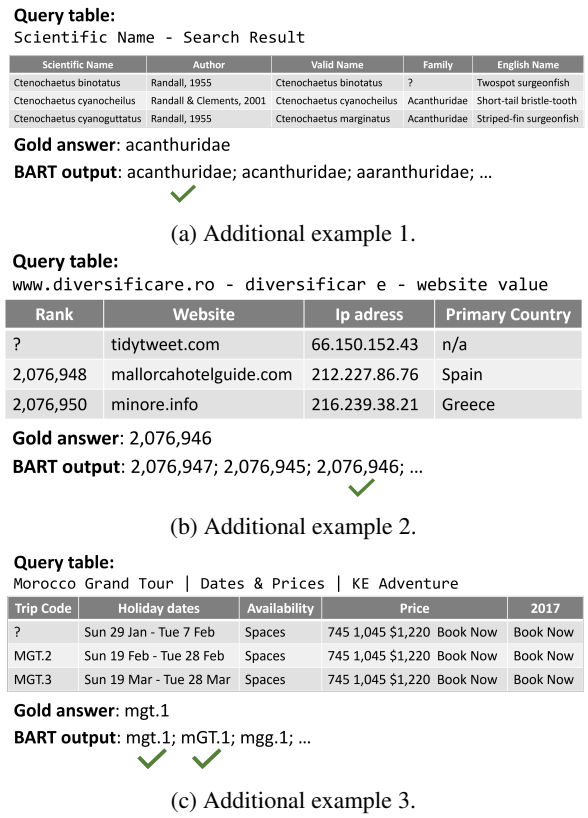
Cell Filling Except for the above mentioned normalization we use cell values which have occurred atleast 10 times in the training dataset.

C Cell Filling BART Examples

Additional BART cell filling output examples on WebTables dataset are in Figure 6.

D Compute Infrastructure

All row and column population experiments were done on a single P100 GPU. This gave train times of 24 to 48 hours. All cell filling experiments were done on a single A100 GPU, with train times of 24 hours.



(a) Additional example 1.

(b) Additional example 2.

(c) Additional example 3.

Figure 6: Additional BART output examples on WebTables dataset.