RECA: Related Tables Enhanced Column Semantic Type Annotation Framework

Yushi Sun
HKUST
Hong Kong, China
ysunbp@cse.ust.hk

Hao Xin
HKUST
Hong Kong, China
hxinaa@cse.ust.hk

Lei Chen
HKUST
Hong Kong, China
HKUST(GZ)
Guangzhou, China
leichen@cse.ust.hk

ABSTRACT

Understanding the semantics of tabular data is of great importance in various downstream applications, such as schema matching, data cleaning, and data integration. Column semantic type annotation is a critical task in the semantic understanding of tabular data. Despite the fact that various approaches have been proposed, they are challenged by the difficulties of handling wide tables and incorporating complex inter-table context information. Failure to handle wide tables limits the usage of column type annotation approaches, while failure to incorporate inter-table context harms the annotation quality. Existing methods either completely ignore these problems or propose ad-hoc solutions. In this paper, we propose Related tables Enhanced Column semantic type Annotation framework (RECA), which incorporates inter-table context information by finding and aligning schema-similar and topic-relevant tables based on a novel named entity schema. The design of RECA can naturally handle wide tables and incorporate useful inter-table context information to enhance the annotation quality. We conduct extensive experiments on two web table datasets to comprehensively evaluate the performance of RECA. Our results show that RECA achieves support-weighted F1 scores of 0.853 and 0.937 with macro average F1 scores of 0.674 and 0.783 on the two datasets respectively, which outperform the state-of-the-art methods.

PVLDB Reference Format:
Yushi Sun, Hao Xin, and Lei Chen. RECA: Related Tables Enhanced Column Semantic Type Annotation Framework. PVLDB, 16(6): 1319 - 1331, 2023.
doi:10.14778/3583140.3583149

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/ysunbp/RECA-paper.

1 INTRODUCTION

In various data mining applications, column type annotation plays a crucial role: In schema matching, column type annotation can help identify semantic relationships [15, 27]; Accurate annotation of data types can efficiently facilitate automated data cleaning [19, 28]; Integration of tabular data from different sources can also benefit from the accurate annotation of column types [32, 36]. In general, table column types can be divided into two categories: *atomic* and *semantic* [36]. *Atomic* types, such as **Integer** and **Boolean**, provide fundamental and general information about the data type of the column content. *Semantic* types, such as **BirthPlace** and **Person**, provide fine-grained information of the column content. The fine-grained *semantic* type is helpful for various downstream applications that require a deep understanding of the column content semantics. However, although most existing systems can accurately annotate the *atomic* types, they are challenged by the difficulty of accurately annotating the *semantic* types solely based on the column content. As shown in Figure 1, the first columns in all three tables have similar content, which are the names of artworks. However, the semantic label of the first columns in Table T_1 and T_2 is **TelevisionShow**, while that in Table T_3 is **VideoGame**. The ambiguity of the column content makes the column semantic type annotation a great challenge to the database community.

In response, various approaches have been proposed to utilize additional context information in the tables to resolve the ambiguity and improve the annotation quality: Hulsebos et al. proposed Sherlock [15] based on hand-crafted statistical features. Sato [36] incorporates table topic features and inter-column co-occurrence relationships into generating the column representations. TaBERT [35] utilizes BERT [9] as a base model to capture the table content features. TABBIE [16] refines the idea of TaBERT by encoding rows and columns respectively to provide a more comprehensive view of tables. DODUO [30] designs a Transformer-based structure to encode the intra-table contextual information inside each table. Among these methods, TABBIE and DODUO achieve state-of-the-art performance. Although these methods achieve state-of-the-art performance on the column semantic type annotation task, they are challenged by the following difficulties: **(1) Difficulty in handling wide tables:** In the era of big data, the size of tables is growing. As shown in [24], the average number of columns of tables in Open Data is 16, with a large variance: some tables have hundreds of columns. Therefore, column semantic type annotation approaches that effectively handle these wide tables are needed. However, the existing state-of-the-art column semantic type annotation methods cannot handle the wide tables well due to the maximum input length limits of the language models (LMs) used by these methods. For instance, TABBIE [16] and DODUO [30] use BERT as the core LM, which has a hard maximum input length limit of 512 tokens [9]. In response, TABBIE [16] ignores the issue of wide tables; DODUO [30] suggests designing user-defined splitting rules,

Proceedings of the VLDB Endowment, Vol. 16, No. 6 ISSN 2150-8097.
doi:10.14778/3583140.3583149

splitting the wide tables into clusters, and then encoding each cluster to perform column semantic type annotation for wide tables. Unfortunately, DODUO [30] fails to provide a detailed design of the table splitting rules and the method of combining the annotations provided by each cluster. Besides, designing such splitting rules is non-trivial: rich domain knowledge is required to design rules that are both effective and generalizable. As a result, handling wide tables properly remains a challenge to the existing LM-based column semantic type annotation methods. **(2) Difficulty in incorporating complex inter-table context information:** TCN [33] draws schema-related tables from various websites and extracts inter-table context information. As suggested by [33], inter-table context information is potentially useful for column semantic type annotation. By incorporating the inter-table context information, ambiguity in column content is expected to be reduced, and the annotation quality can be improved. As shown in Figure 1, the first columns of T_1 and T_2 refer to TV series. Annotating the first column of T_1 solely based on the content of T_1 is difficult: the number of rows in T_1 is few, so it's difficult for existing methods to resolve the ambiguity between the similar semantic types, such as **TelevisionShow**, **VideoGame**, etc. By aligning T_1 and T_2, we augment the content of T_1, the model receives more useful information for annotation and the annotation quality is thus improved. However, both TABBIE [16] and DODUO [30] ignore the useful inter-table context information. Incorporating inter-table context information is non-trivial: Since the table schema information is generally not available [16, 30, 36], it is difficult to align tables that potentially have similar schema solely based on the table content. Although TCN [33] incorporates inter-table context information, it is limited to handling relational tables with a known table schema[1] and page topic. Unfortunately, valid table schema and page topic annotations are generally unavailable in real-world web tables datasets [6, 14]. Therefore, effectively incorporating inter-table context information solely based on the table content without additional table schema and page topic annotations remains a challenge to the column semantic type annotation task.

In order to handle wide tables, we first studied the reasons why previous approaches have failed. One key observation is that they incorporate intra-table context to enhance the annotation quality. However, since the intra-table context (columns except for the one that we aim at annotating) grows linearly with respect to the table width, the design of incorporating intra-table context without splitting can easily lead to the exhaustion of input tokens for LM-based approaches. As a result, previous approaches cannot process entire wide tables properly. To circumvent this problem, we suggest utilizing inter-table context instead to enhance the annotation quality. In order to handle the difficulty of incorporating complex inter-table context information, we define and adopt a novel named entity schema for web tables. Based on this novel named entity schema, we can successfully align tables that have similar schemata and extract useful inter-table context information from the tables aligned with the original table. In this way, the difficulty of incorporating inter-table context without accessing table schemata and page topics can be resolved.

In general, RECA is a self-contained data-driven representation learning framework, i.e. it requires no external linkage to a knowledge base nor additional manual annotations of table schema and page topic. RECA considers the target column only in the original table so that it can easily accommodate wide tables. We define a novel named entity schema with table filtering and alignment strategies for RECA to efficiently extract and process the inter-table context to obtain rich context information for the annotation. RECA further leverages BERT [9] as the core LM to encode the target column and the inter-table context information. Finally, RECA combines the embeddings generated and annotates the target column. Specifically, we have made the following contributions:

- We propose a novel self-contained data-driven representation learning framework called RECA for column semantic type annotation. RECA extracts and leverages inter-table context to enhance the annotation quality of the target column, thus resolving the wide table issue.
- We define a novel named entity schema for RECA to efficiently align related and sub-related table, which resolves the difficulty of incorporating inter-table context without manual table schemata annotations and page topics.
- We conduct extensive experiments on two real-world web table datasets to show that RECA outperforms all the state-of-the-art methods. The result demonstrates the effectiveness of utilizing the inter-table context to annotate column semantic types accurately.
- We show that RECA is data efficient and learning efficient, since it requires shorter input token sequences and fewer training data to achieve high performance. The result demonstrates the robustness and power of RECA on the column semantic type annotation task.

2 RELATED WORK

In general, there are two lines of research related to our work: feature-based data interpretation and pre-trained model-based representation learning.

Feature-based data interpretation Research in feature-based data interpretation aims to capture the semantics in tabular data by extracting hand-crafted, statistical, and semantic features. The extracted features are then employed to resolve the column semantic type annotation problem.

Some early works extract the semantics of data by designing hand-crafted features. SemanticTyper [29] first separates the tabular data into textual and numeric types and then suggests using Term Frequency-Inverse Document Frequency (TF-IDF) and a two-sample Kolmogorov-Smirnov test [22] to extract the semantics from the tabular data. Pham et al. [26] include Mann-Whitney test [22] and Jaccard Similarity for textual data to further improve the performance of SemanticTyper.

Several previous works consider statistical and semantic features in column semantic type annotation. Sherlock [15] extracts character-level, word-level, paragraph-level, and global-level features and then employs deep learning classifiers to achieve high

[1]TCN [33] manually cleans the headers to form the table schema, based on which it develops the inter-table relationship. However, according to the common practice of previous work [16, 30, 36], table headers should not be provided as a piece of known information to prevent introducing noise or incurring potential information leaks.

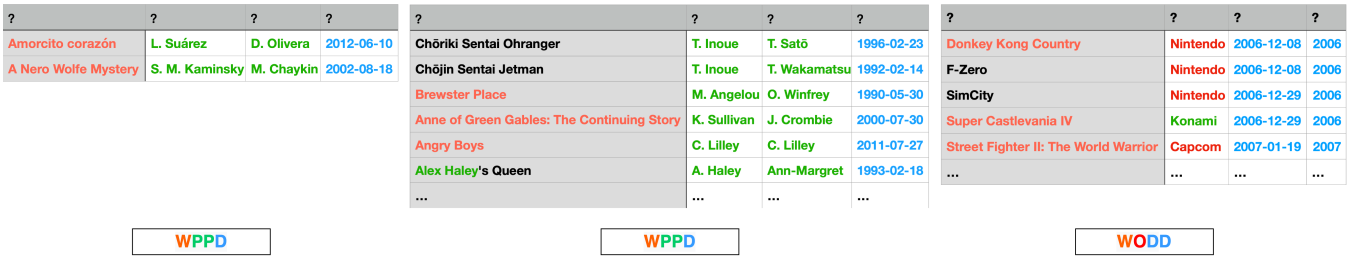


Figure 1: From the left to right we denote the tables as Table T_1, T_2, and T_3. T_1 and T_2 contain information about television series; T_3 is about video games. The named entities extracted from the tables by spaCy [13] are colored differently: Orange - WORK_OF_ART (W); Green - PERSON (P); Blue - DATE (D); Red - ORG (O). The bottom part shows the three tables' corresponding named entity schema strings.

annotation quality. Sato [36] furthers the capability of Sherlock [15] by incorporating the table topic features and intra-table context into the annotation task.

However, the hand-crafted, statistical, and semantic features used by these methods are too general to capture the rich semantics and form expressive representations from tables, which is a significant limitation. As suggested by [33], these methods cannot effectively capture the fine-grained semantics in tables because of the fact that the shallow networks of these methods have limited expressiveness.

Pre-trained model-based representation learning To annotate column types, pre-trained neural models are used by recent methods to learn representations of tables and generate predictions. In general, the pre-trained model-based representation learning approaches generally outperform the feature-based data interpretation approaches and thus serve as our main baselines.

TURL [8] employs a pre-training + fine-tuning framework to perform column semantic type annotation. A visibility matrix is used to mask out the components of tables that are structurally irrelevant to the transformer structure. The generated column embeddings are fed into a classifier for column semantic type annotation. TaPas [12] encodes the queries and the table content together. The positional embedding of the original BERT [9] structure is modified by TaPas to ensure a better understanding of tables. TaBERT [35] slices the web tables with the user queries and feeds them into the BERT structure to learn the representations of every column in the tables. Based on Sherlock [15], Zhou et al. [37] utilize a Star-Transformer [11] structure to encode the features extracted using Sherlock. TCN [33] suggests using both intra-table and inter-table information and applying multi-task training to perform column semantic type annotation. However, the information used by TCN such as table schema and page topic requires extra manual annotation and cleaning, so it cannot be widely applied on most web table datasets. (For instance, the Semtab2019 dataset [6] does not provide table schema annotations and page topic information for its tables, and the Webtables dataset [36] does not provide page topic information.) TABBIE [16] furthers the design of TaBERT [35] by considering two independent transformers to encode the rows and columns of the tables jointly. TABBIE [16] provides a more comprehensive view of tables in comparison with TaBERT [35] and therefore achieves better annotation quality in comparison with TaBERT. DODUO [30] uses a transformer structure to encode all the columns in the tables in one pass. The representations of all the columns in the tables are considered together when training the classification module. It also suggests applying multi-task learning to generate representations that contain rich semantic information.

Among these methods, TABBIE [16] and DODUO [30] achieve state-of-the-art performance on the column semantic type annotation task. However, they neglect the inter-table context information, which is beneficial in generating high-quality annotations. Besides, they cannot handle wide tables well. TABBIE and DODUO models directly feed all the columns into a transformer structure to learn the representation of each column, which leaves space for further improvement in terms of boosting the annotation performance.

3 NOTATIONS AND PROBLEM DEFINITION

We first discuss the notations used in this paper and then formally define the problem of column semantic type annotation.

3.1 Notations and Concepts

The general notations are introduced in Table 1. We further introduce some important concepts used in this paper.

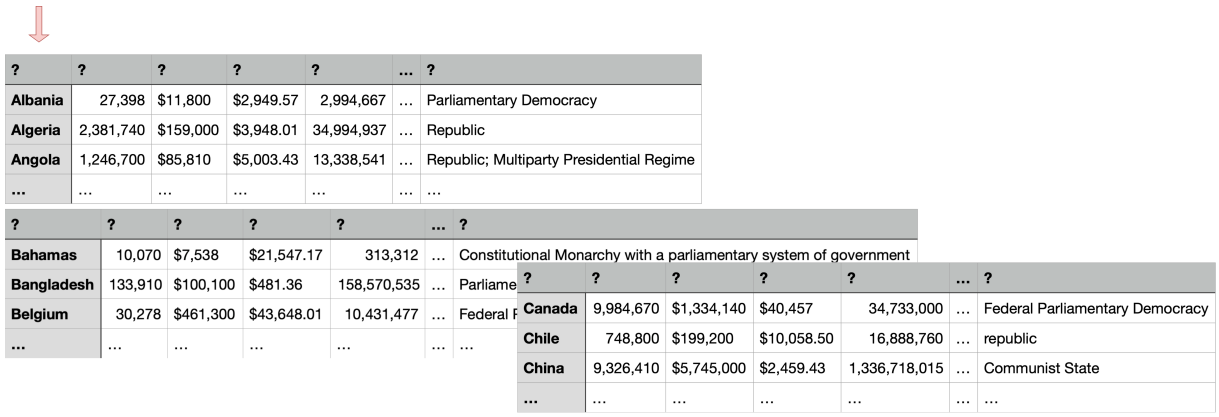
Definition 1: (Target Column): We define the target column to be the column that has a manual annotation of its semantic type in the table. Note that not all the columns necessarily have a manually annotated label.

Definition 2: (Named Entity Schema): Named Entity Schema is the table schema generated based on the named entity type extracted in each column. As shown in Figure 1, the named entity schemata are formed by selecting the most frequent named entity type in each column. Each named entity schema s is represented as a string.

Definition 3: (Edit distance): The edit distance $\Phi(s_i, s_j)$ between named entity schemata s_i and s_j is defined as the minimum number of changes required to convert s_i to s_j with the insertion, deletion and substitution editing operations [5, 23].

Definition 4: (Target Table): We define the table that contains the target column to be annotated as the target table. For instance, if we currently want to annotate the semantic type of column C_i^m, then the target table here refers to T_i.

Definition 5: (Candidate Table): A table T_j is a candidate table of T_i if the Jaccard similarity [17] between their content is less than or equal to a threshold.



?	?	?	?	?	...	?
Albania	27,398	$11,800	$2,949.57	2,994,667	...	Parliamentary Democracy
Algeria	2,381,740	$159,000	$3,948.01	34,994,937	...	Republic
Angola	1,246,700	$85,810	$5,003.43	13,338,541	...	Republic; Multiparty Presidential Regime
...	...	...	...	...	...	...

?	?	?	?	?	...	?
Bahamas	10,070	$7,538	$21,547.17	313,312	...	Constitutional Monarchy with a parliamentary system of government
Bangladesh	133,910	$100,100	$481.36	158,570,535	...	Parliame
Belgium	30,278	$461,300	$43,648.01	10,431,477	...	Federal P
...	...	...	...	...	...	...

?	?	?	?	?	...	?
Canada	9,984,670	$1,334,140	$40,457	34,733,000	...	Federal Parliamentary Democracy
Chile	748,800	$199,200	$10,058.50	16,888,760	...	republic
China	9,326,410	$5,745,000	$2,459.43	1,336,718,015	...	Communist State
...	...	...	...	...	...	...

Figure 2: We denote the table lying on the top as Table T_4; the two tables at the bottom are Table T_5 and T_6, respectively. Note that these tables describe the detailed information of countries, so it can potentially grow very wide and thus causes challenges to existing methods. We notice that Table T_4, T_5, and T_6 share the same table schema, so we link them together and develop the main idea of RECA based on this.

Definition 6: (Related Table): A candidate table T_j is a related table of the target table T_i if $s_j = s_i$.

Definition 7: (Sub-related Table): A candidate table T_j is defined as a sub-related table of the target table T_i if the edit distance $\Phi(s_i, s_j)$ is less than or equal to a threshold.

Definition 8: (Identified Column): An identified column is the column in the related or the sub-related table that has the same column index and the same named entity type as the target column.

Definition 9: (Inter-table context): Given a target table, the inter-table context refers to the collection of the identified columns in related tables and sub-related tables of the target table.

3.2 Problem definition

Aligning with the problem definition of previous methods [15, 16, 30, 36], the goal of column semantic type annotation is to predict the semantic type of the target column from a pre-defined semantic type set. The pre-defined semantic type sets are normally defined using existing ontologies to ensure the coverage of column semantic types [15, 18]. The prediction of the target column type should be solely based on the table content without accessing the table headers (table schema) in alignment with the common practice of previous work [15, 16, 30, 36]. The problem is formulated as a multi-class classification problem.

Problem 1: (Column semantic type annotation): Given a web table T (without table headers) from the dataset D, denote the target column as C^t in T. The column semantic type annotation model W annotates C^t with a semantic type $\bar{y}^t = W(C^t, T, D) \in S$, such that $\bar{y}^t$ best fits the semantics of C^t (being the closest to the ground truth semantic type y^t).

Table 1: General notations with corresponding descriptions.

Notation	Description
$D = \{T_1, T_2, ..., T_n\}$	web table dataset that contains n tables
T, (T_i)	the (i-th) web table without headers
M, (M_i)	the number of columns in the table T, (T_i)
N, (N_i)	the number of rows in the table T, (T_i)
C^m, (C_i^m)	the m-th column in the table T, (T_i)
C, (C_i)	the set of all the columns in T, (T_i)
$c^{m,n}$, $(c_i^{m,n})$	the cell at the m-th column, the n-th row of the table T, (T_i)
E^m, (E_i^m)	the collection of named entities in the m-th column of the table T, (T_i)
y^m, (y_i^m)	the semantic type ground truth label of the m-th column in the table T, (T_i)
s, (s_i)	the named entity schema of table T, (T_i)
R, (R_i)	the set of related tables of table T, (T_i)
X, (X_i)	the set of sub-related tables of table T, (T_i)
I_R^t, $(I_{R_i}^t)$	the set of identified columns corresponding to the target column C_i^t in related tables R, (R_i)
I_X^t, $(I_{X_i}^t)$	the set of identified columns corresponding to the target column C_i^t in sub-related tables X, (X_i)
S	the pre-defined column semantic type set
Ψ	Dictionary-based mapping from each named entity type to a distinct English character

4 METHODOLOGY

In order to handle the wide tables, a naive design encodes the target column only. However, such a design discards the table context information, which is potentially useful for annotating the target column [36]. In order to handle the wide table issue without compromising the quality of annotations, we incorporate inter-table context information. In Figure 2, we aim at annotating the first columns, which have **Country** type. Tables T_5 and T_6 are the tables that are potentially related to table T_4. Since table T_4 is wide, it cannot be properly processed by existing methods, such as DODUO [30], without splitting. The point is that tables T_4, T_5, and T_6 are similar in table schema and table content, so they can thus be

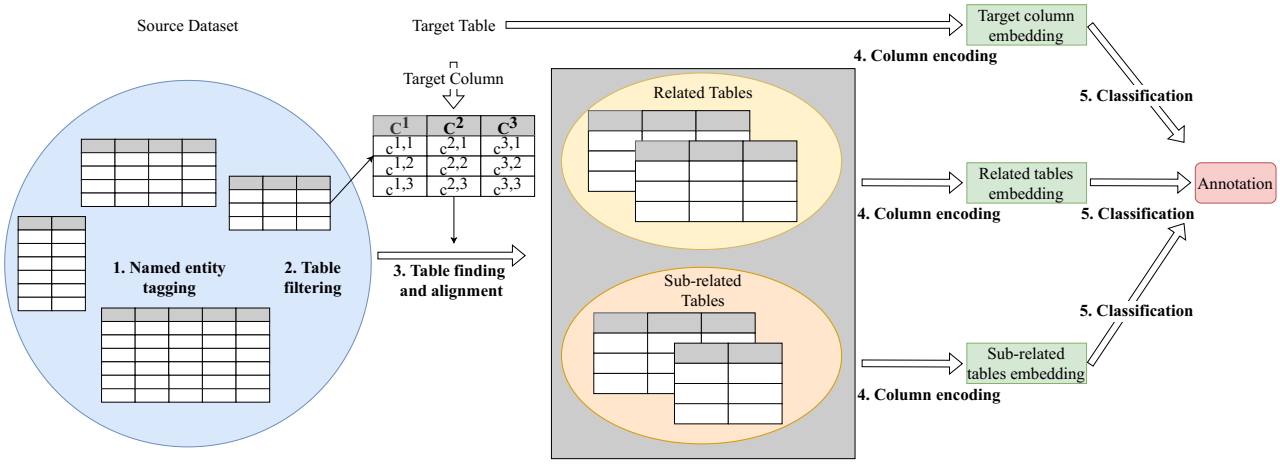


Figure 3: The general Column Semantic Type Annotation process of RECA.

viewed as a holistic collection of data records. The inter-table context from tables T_5 and T_6 can extend the information in the original table T_4's target column and thus serve as a perfect replacement for the intra-table context, which constrains LMs' ability to handle the wide tables.

In order to incorporate inter-table context information without additional table schema and page topic annotations, designing effective related table alignment and filtering rules is necessary. As shown in Figure 1, table schema and page topic are not available for general web tables. Therefore, we need to design a special approach to filter out irrelevant tables and align the related tables with the original tables. To this end, we define a novel table schema called a named entity schema. As shown in Figure 1, named entity types can be detected from the table and we can form the so-called named entity schema for the web tables by selecting the most frequent named entity type for each column. Intuitively, web tables with similar named entity schemata are likely to describe the same collection of data records. For example, the named entity schema in tables T_1 and T_2 is [WORK_OF_ART, PERSON, PERSON, DATE], while the named entity schema in table T_3 is [WORK_OF_ART, ORG, DATE, DATE]. Despite the fact that the first columns in both T_2 and T_3 are the names of artwork (and the fact that the first columns in T_2 and T_3 look similar in terms of content and format), T_2 and T_3 are not likely to belong to the same data record collection because of the difference in their named entity schemata. As a result, the possibility that the first columns of T_2 and T_3 refer to the same column type is lower than that of the first columns of T_2 and T_1. Indeed, the first columns in both Tables T_1 and T_2 include the names of TV series, while that in T_3 includes the names of games. In this way, we can gather together the tables that are similar in schemata. Table column alignment can then be applied thereafter, and we can find columns that likely extend the target column and form the inter-table context for column semantic type annotation.

The general process of RECA is shown in Figure 3: RECA first extracts the named entities and assigns types from a pre-defined named entity type set, which is discussed in Section 4.1. Based on the named entity types extracted, it further generates the named entity schema for the input table. In Section 4.2, we discuss how RECA filters out the topic-irrelevant tables. Then it finds the related tables and sub-related tables of the target table based on their corresponding named entity schemata, which is introduced in Section 4.3. RECA then encodes the target column and its corresponding inter-table context information, which is discussed in Section 4.4. Finally, we introduce how RECA performs column type annotation based on the encoding generated, which is discussed in Section 4.5.

4.1 Named entity tagging

First note that under our problem formulation, the table schema (header) information is not provided as input to the model. Therefore, we propose forming an approximated schema based on named entities in each column to identify related and sub-related tables. In order to achieve this, the first step is to identify the named entities in the tables.

Given a table T from dataset D that has M columns and N rows, RECA utilizes tagging tools such as spaCy [13] to identify the named entities in each column $C^m \in C$ and tag them into basic named entity types as shown in Figure 4. The tagged named entities in each column are denoted as $\bar{E}^m$.

Apart from the basic named entity types, we notice that several named entity types have different representation formats, such as DATE and PERSON, which can be further divided into sub-types based on simple hand-crafted rules. For instance, as shown in Figure 1, the fourth columns of both tables T_1 and T_3 are annotated as DATE, while the format of the content in the columns is different (YYYY-MM-DD & YYYY). The difference in the data format suggests that the two tables are likely to be from different sources. Therefore, the possibility that the two tables describe the same data collection is relatively low.

To this end, we designed hand-crafted rules to further classify types DATE and PERSON into sub-types based on the data format[2]. Detailed descriptions of the sub-types are introduced in Table 2. We only divide the DATE and PERSON types into sub-types, since these two types are common, and designing hand-crafted rules to

[2]In reference to the data formats in the Document Automation Language of Oracle

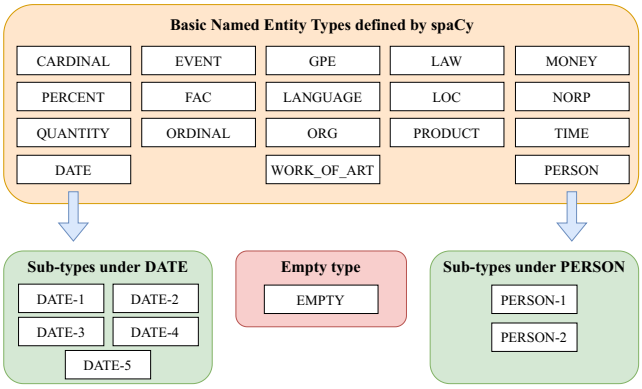


Figure 4: The basic named entity types defined by spaCy [13] (orange); the sub-types defined under DATE and PERSON (green); the EMPTY type that handles the case when no named entity is detected (red).

Table 2: Sub-types and the corresponding descriptions.

Sub-type	Description
DATE-1	dates in YYYY format. *e.g. 2022*
DATE-2	dates that contain month names or abbreviations of month names. *e.g. January 16th, 2022*
DATE-3	dates in DD-MM-YYYY or MM-DD-YYYY or YYYY-MM-DD format. *e.g. 18-11-1998*
DATE-4	dates that contain numerical months and days only. *e.g. 02-29*
DATE-5	dates in other formats.
PERSON-1	person names that contain abbreviations. *e.g. J. K. Rowling*
PERSON-2	person names in other formats.

classify entities in these two types is relatively simple[3]. Note that the sub-types we defined are general and may be further divided, but further dividing the sub-types may require more complicated hand-crafted rules, which would not be desirable for maintaining the generalizability of the RECA framework on other web table data. In other words, there is a trade-off between the generalizability of the hand-crafted rules and the granularity of the sub-types generated. Furthermore, we also include a new type called EMPTY to handle cases where no named entity is extracted from the column. As a result, we obtain the finer-grained named entity types, based on which we can generate the updated tagged named entities in each column m as E^m.

After obtaining the updated tagged named entities for each column, we further generate the named entity schema for the table T. For each E^m obtained, we denote the most frequent named entity type[4] in E^m as $\hat{e}^m$. Based on this, we obtain the named entity schema string s of table T: $s = (\Psi(\hat{e}^1), \Psi(\hat{e}^2), ..., \Psi(\hat{e}^M))$, where each entity type shown in Figure 4 is mapped to a distinct English character through a dictionary-based mapping Ψ[5]. In this way, we can generate the table schemata for all the tables in the dataset D (Lines 3 and 5, Algorithm 1).

4.2 Table filtering

In order to obtain candidate tables that are topically similar to the target table, we need to filter them based on the table content. In reference of [10], we adopt the Jaccard similarity [17] to achieve this goal. For each target table T_i and each table T_j, where $(j \neq i)$, we denote the set of words in T_i as A_i and the set of words in T_j as A_j. Then we compute the Jaccard similarity between T_i and T_j:

$$\text{Jaccard}(A_i, A_j) = \frac{|A_i \cap A_j|}{|A_i \cup A_j|} \tag{1}$$

We include T_j as a candidate table if the similarity is smaller than or equal to a threshold δ. After performing table filtering, we obtain the candidate tables that are topically related to the target table (Line 9, Algorithm 1).

4.3 Table finding and alignment

Note that Section 4.2 considers the relatedness of tables mainly based on the table topics. We now consider the relatedness of tables based on the structural information (schema and location). Given a table T_i in the dataset D, we now want to find its corresponding related and sub-related tables based on the named entity schemata extracted. The alignment of the related tables is straightforward: we simply identify the candidate tables T_j that have the same schema s_j as the schema s_i of the table T_i, where $j \neq i$. We denote the set of these related tables T_j as R_i.

In order to identify sub-related tables that are 'approximately' aligned with the target table, we consider these requirements:

- **Schema similarity**: The named entity schema of the sub-related table should not be much different from that of the target table.
- **Column location alignment**: The named entity type of the target column matches with that of the column at the identical location in the sub-related table.

The intuition of schema similarity is that sub-related tables that share similar schemata with the target table are likely to have related or similar table content as the target table. For column location alignment, if the inner order of column types in a sub-related table differs much from the target table, then the inner schema structures of the two tables are not similar. Therefore, the possibility that the sub-related table and the target table describe the same data collection is low.

In considering the schema similarity, we compute the edit distance [23] between the table schema string of a candidate table T_k and that of the target table T_i. If the edit distance $\Phi(s_i, s_k)$ is larger than a threshold[6], then the candidate table cannot be regarded as a sub-related table of the target table. If $\Phi(s_i, s_k)$ is larger than 0 and

[3]These rules can be applied as a complement on other tagging tools, since the rules are general and the types DATE and PERSON are also common in other tagging tools.
[4]In case a tie happens, for simplicity, we select the named entity type that occurs first in the column from the most frequent types.
[5]We constructed a dictionary that maps the named entity types and sub-types into 24 distinct English characters.
[6]We empirically set the threshold to be the square root of the width of the target table. i.e., $\sqrt{M_i}$.

less than or equal to the threshold, we add T_k to X_i. If $\Phi(s_i, s_k) = 0$, then T_k is a related table and is added to R_i. We denote the set of sub-related table candidates that are 'schema similar' to the target table as X_i (Line 14, Algorithm 1).

We obtain the identified columns as follows: For the target column t in T_i, we consider each related table T_k in R_i and include C_k^t in $I_{R_i}^t$. For each sub-related table in X_i, we consider the following exact alignment rule to achieve column location alignment: We denote the named entity column type of the target column as $\Psi(\hat{e}_i^t)$. For a sub-related table candidate T_k, if $\Psi(\hat{e}_k^t) = \Psi(\hat{e}_i^t)$, then the t-th column of T_k is the identified column, we include C_k^t in $I_{X_i}^t$ (Line 17, Algorithm 1).

4.4 Column encoding

After obtaining the inter-table context (identified columns in related and sub-related tables, i.e., $I_{R_i}^t$ and $I_{X_i}^t$), the next step is to generate embeddings for the annotation. As shown in Figure 3, the target column, the identified columns in the related tables, and the identified columns in the sub-related tables are encoded independently. Specifically, for the target column C_i^t, we concatenate the cell content in the t-th column to generate the input string L_i^t for BERT model:

$$L_i^t = \text{Concatenate}(c_i^{t,1}, c_i^{t,2}, ..., c_i^{t,N_i}) \quad (2)$$

Similarly, we concatenate the cells from the identified columns of the related table set $I_{R_i}^t$ (or the sub-related table set $I_{X_i}^t$) to obtain the concatenated column set Z_{R_i} (Z_{X_i}):

$$Z_{R_i} = \{\text{Concatenate}(c_k^{t',1}, c_k^{t',2}, ..., c_k^{t',N_k}) | \text{for table } T_k \in R_i\} \quad (3)$$

$$Z_{X_i} = \{\text{Concatenate}(c_k^{t',1}, c_k^{t',2}, ..., c_k^{t',N_k}) | \text{for table } T_k \in X_i\} \quad (4)$$

where t' is the identified column index in each related table (or sub-related table).

Then we generate the input strings $L_{R_i}^t$ and $L_{X_i}^t$ for related tables and sub-related tables:

$$L_{R_i}^t = \text{Concatenate}(Z_{R_i}) \quad (5)$$

$$L_{X_i}^t = \text{Concatenate}(Z_{X_i}) \quad (6)$$

Upon obtaining the input strings for the target column, identified columns of related tables, and sub-related tables, we tokenize the strings and append an additional [CLS] token at the front of each string. Each string is passed to BERT and transformed into a 768-dimension embedding. We denote the embeddings of the target column, the identified columns of related tables and the identified columns of sub-related tables as v_i^t, r_i^t, x_i^t (Line 24, Algorithm 1).

Note that BERT [9] has a maximum input sequence length limit B (which by default is 512). For target columns that exceed this limit, we truncate the sequence to fit within the limit. As for the identified columns in related and sub-related tables, we promote fair allocation of the input tokens (e.g., If there are four identified columns in the related table set, each identified column will take $\frac{B}{4}$ tokens. If the identified column exceeds this limit, we truncate it). The main reason why we adopt such a fair allocation is that we hold an optimal view of the related and sub-related tables aligned: All the related and sub-related tables should be of high quality in terms of relatedness and thus should be considered. Besides, if we consider only some but not all of the related and sub-related tables, then we are at risk of biasing the model towards some tables that have a skewed distribution of data.

Algorithm 1 Training process of RECA

Input:
Number of epochs P
Training dataset D_{tr}
Loss function $\mathcal{J}$
Output:
Trained RECA model W_{tr}

1: **for** $i = 1, 2, ..., |D_{tr}|$ **do**
2: **for** $j = 1, 2, ..., M_i$ **do**
3: Named Entity tagging, obtain $\hat{e}_i^j$ from C_i^j
4: **end for**
5: Generate $s_i = (\Psi(\hat{e}_i^1), \Psi(\hat{e}_i^2), ..., \Psi(\hat{e}_i^{M_i}))$
6: **end for**
7: **for** $i = 1, 2, ..., |D_{tr}|$ **do**
8: **for** $j = 1, 2, ..., i-1, i+1, ..., |D_{tr}|$ **do**
9: Compute Jaccard(A_i, A_j), if Jaccard$(A_i, A_j) \leq \delta$, include T_j to T_i's candidate table set Q_i
10: **end for**
11: **end for**
12: **for** $i = 1, 2, ..., |D_{tr}|$ **do**
13: **for** each table T_k in Q_i **do**
14: Compute edit distance $\Phi(s_i, s_k)$, if $\Phi(s_i, s_k) = 0$ add T_k to R_i; if $0 < \Phi(s_i, s_k) \leq \sqrt{M_i}$ add T_k to X_i
15: **end for**
16: **for** each target column t in T_i **do**
17: Align and store $I_{R_i}^t$ and $I_{X_i}^t$ to D_{tr} as side information
18: **end for**
19: **end for**
20: Initialize the LM core of W_{tr} with pre-trained BERT weights
21: **for** epoch = 1 to P **do**
22: Randomly split D_{tr} into batches $\{B_1, B_2, ..., B_f\}$
23: **for** $l = 1, 2, ..., f$ **do**
24: Encoding and classification $L = \mathcal{J}(B_l, W_{tr})$
25: $W_{tr} = \text{Backpropagate}(W_{tr}, L)$
26: **end for**
27: **end for**
28: **return** W_{tr}

4.5 Classification

The generated embeddings for the target column and the corresponding identified columns of related tables and sub-related tables are fed into the classification module for the annotation of the target column type as shown in Figure 3.

Note that the identified columns in the related and sub-related tables are expected to be of the same semantic type as the target column yet have different content to provide a broader view of context information for the target column. Therefore, we design a three-network classification module to compute the annotations based on the content of the target column, related tables, and sub-related tables. Specifically, we adopt the standard two-layer classification

module [34] for each network: A dropout layer that avoids overfitting and a linear layer that generates the annotations based on the input embeddings. We denote the outputs of the three networks as $\hat{v}_i^t$, $\hat{r}_i^t$, $\hat{x}_i^t$ respectively. We then generate the final annotation a_i^t of the target column by combining the three outputs:

$$a_i^t = \alpha * \hat{v}_i^t + \beta * \hat{r}_i^t + \gamma * \hat{x}_i^t \tag{7}$$

where α, β, and γ are trainable weights.

Cross-Entropy loss is used as the loss function. The formula for the loss $\mathcal{J}$ is as follows:

$$\mathcal{J} = -\sum_{k=1}^{|S|}(p_k * log(q_k)) \tag{8}$$

where $p_k \in \{0, 1\}$ is the ground truth label for the k-th type. q_k is the k-th entry of a_i^t. The predicted column type $\bar{y}_i^t = \arg\max_k q_k$ (Line 24, Algorithm 1).

4.6 Analysis

Algorithm 1 shows the complete training process of RECA. We perform named entity tagging in each table first and formulate the named entity schemata (lines 1 to 6). Then we compute the Jaccard similarity between each pair of tables to obtain candidate tables (lines 7 to 11). After that, we compute the edit distance between candidate table pairs (lines 13 to 15) so as to perform column alignment (lines 16 to 18). Then we adopt BERT to encode the columns and perform classification (line 24). For a dataset D with n tables, we denote the maximum number of columns of tables as $\hat{M}$; the maximum number of rows of tables as $\hat{N}$; BERT embedding dimension as d; BERT maximum input sequence limit as B; the size of the pre-defined label set as $|S|$; the number of epochs as P; the number of batches as f.

Time complexity For each table, named entity tagging is performed to generate its named entity schema: we need to go through each column and each row, which takes $O(\hat{M}\hat{N})$ time complexity. For all the n tables, we need $O(\hat{M}\hat{N}n)$ time. Computing the Jaccard similarity takes $O(\hat{M}\hat{N}n^2)$ time since there are $O(n^2)$ pairs to compute and computing the Jaccard similarity for each pair requires going through the table content and thus takes $O(\hat{M}\hat{N})$. Computing edit distance takes $O(\hat{M}^2n^2)$ time. Since we compute the pairwise edit distance, each pair needs $O(\hat{M}^2)$ time. Aligning identified columns takes $O(\hat{M}n^2)$ time, since each table contains at most $\hat{M}$ target columns, each target column has at most n identified columns, in total there are n tables, aligning a pair of columns takes $O(1)$. Preparation for the input sequence takes $O(\hat{N}n)$ for each column so in total $O(\hat{M}\hat{N}n^2)$ time. Encoding each column with its related and sub-related identified columns takes $O(Bd(B+d))$ time [9, 31]. So preparing input and encoding the columns takes overall $O(\hat{M}\hat{N}n^2 + Bd(B+d)nP)$ time. Classification takes $O(d^2|S|nP)$ time. So the overall time complexity is $O(\hat{M}\hat{N}n + \hat{M}\hat{N}n^2 + \hat{M}^2n^2 + \hat{M}n^2 + \hat{M}\hat{N}n^2 + Bd(B+d)Pn + d^2|S|Pn) = O(\hat{M}^2n^2 + \hat{M}\hat{N}n^2 + (B^2 + Bd + d|S|)dPn)$.

Space complexity For named entity tagging, we need $O(\hat{M}\hat{N} + \hat{M}n)$ memory space to load each table and store the named entity schemata. The Jaccard similarity computation step requires $O(\hat{M}\hat{N} + n^2)$ space since $O(\hat{M}\hat{N})$ is used for computing the Jaccard similarity and $O(n^2)$ is used to store the pairwise Jaccard values between tables. Computing the edit distance requires $O(2\hat{M} + n^2)$ space: $O(2\hat{M})$ space to perform the computation and the edit distance results require $O(n^2)$ memory space. Aligning identified columns needs $O(\hat{M}n^2 + \hat{M}\hat{N}n)$ space, where we need $O(\hat{M}n^2)$ to perform alignment and store the alignment relationship (since each table contains at most $\hat{M}$ target columns and each target column has at most n identified columns, there are n tables in total) and $O(\hat{M}\hat{N}n)$ to store the table content. Preparing input and encoding columns requires $O(\hat{N}n + (B+d)d)$ space: $O(\hat{N}n)$ space for preparing the input sequence (since each column has at most $\hat{N}$ cells and each column has at most n identified columns), $O(Bd)$ space for the input and output embeddings and $O(d^2)$ for self-attention matrices. The classification step requires $O(d|S|)$ space. So the overall space complexity is $\max(O(\hat{M}\hat{N} + n^2), O(\hat{M}n(\hat{N} + n)), O(\hat{N}n + (B + d)d), O(d|S|))$.

5 EXPERIMENT

We compared RECA with the following baselines: Sherlock [15], TaBERT [35], TABBIE [16], and DODUO [30]. Comparison with Sato [36] is omitted, since it is outperformed by DODUO. We do not compare with TCN [33], since TCN requires additional manual annotations and cleaning for the table schemata and page topics.

Table 3: Statistics of the Semtab2019 dataset and the WebTables dataset.

	# types	# tables	# cols	avg rows	avg cols
Semtab2019	275	3,045	7,603	69.0	4.5
WebTables	78	32,262	74,141	20.0	2.3

5.1 Datasets

We selected the Semtab2019 dataset [6] and the WebTables dataset from the VizNet corpus [14] used in [30, 36] to evaluate the performance of RECA. These datasets contain vertical relational web tables with valid semantic labels of different levels of granularity. Specifically, we selected rounds 1, 3, and 4 in the Semtab2019 dataset. In total, there are 3,045 tables with 275 distinct semantic types. For the WebTables dataset, we selected the tables with multiple columns. There are 32,262 tables with 78 distinct types. As shown in Table 3, the tables in the Semtab2019 dataset tend to be larger in size than those in the WebTables dataset. Besides, the number of semantic types in the Semtab2019 dataset is larger than that of the WebTables dataset, while the WebTables dataset has more tables than the Semtab2019 dataset, and every column in the WebTables dataset is annotated. Considering these differences in the properties of the two datasets, conducting experiments on both datasets provided a comprehensive and deep understanding of the properties of RECA. All the table headers are excluded following the common practice of previous work [15, 16, 30, 36], the main reasons are: a) Several datasets [6, 7, 18] contain index-like headers, which do not contain useful semantic information to generate meaningful table schema. b) The commonly-used benchmark WebTables dataset utilizes the column headers as the ground truth semantic type labels. In this case, utilizing table headers as input is likely to incur information leaks since the model will learn to directly annotate the columns based on the headers instead of the

table content. Incorporating column headers as input would be unfair when performing a comparison with other approaches in the evaluation process.

In order to select training, validation, and testing sets from the Semtab2019 dataset [6], we randomly sampled 10% of the annotated columns to form the testing set. Then we randomly divided the rest of the data into five folds to conduct 5-fold cross-validation (80% for the training set and 20% for the validation set). We preserved the percentage of each type[7] when generating the training, validation, and testing sets. As for the WebTables dataset, we followed the 5-fold cross-validation setting[8] provided by [36].

5.2 Baselines

In order to evaluate the performance of RECA, we selected the following methods for comparison:

- Sherlock [15]: Sherlock extracts character-level and global-level statistical features from the tables. Besides, it considers semantic features from word-level and paragraph-level to form vector representations for table columns.
- TaBERT [35]: TaBERT jointly considers queries and table content to identify three salient rows in a table to generate table content snapshots. Based on the table content snapshots generated, TaBERT utilizes BERT to process and formulate representations for each table column for classification purposes.
- TABBIE [16]: TABBIE adopts a dual-transformer structure to encode the table columns and rows. The generated embedding for the target column is then used to annotate the column semantic types.
- DODUO [30]: DODUO utilizes a transformer structure to encode all the columns in the target table together in order to consider the intra-table context.

Among these methods, TABBIE and DODUO achieve state-of-the-art performances. To be fair for the evaluation, we used the open-sourced official implementations of the above-mentioned baselines. We preserved the experimental settings mentioned by the original papers as much as possible. Specifically, for Sherlock [15], we followed the official implementation [1]. For TaBERT [35], we adopted the prescribed implementation [2]. Note that the original purpose of TaBERT was to facilitate the question-answering task on table content. We followed the suggestions in [21] and provided a blank space as the query input of TaBERT. For TABBIE [16], we used the official implementation [3] in the paper. For DODUO [30], we adopted the implementation [4] released by the authors and maintained the experimental settings of the model.

5.3 Experiment Metrics

We adopted F1 scores as the evaluation metrics for RECA ($F1 = 2 \times \frac{precision * recall}{precision + recall}$). As suggested by [36], due to the imbalanced distribution of semantic types, we selected two different kinds of F1 scores, support-weighted F1 score and macro average F1 score to evaluate the performance of different methods comprehensively. Specifically, the Support-weighted F1 score is the weighted average of per-type F1 scores, where the weights are proportional to the support in each type. The macro average F1 score is the mean of all the per-type F1 scores, focusing on the types with fewer supports.

[7]We used StratifiedKFold in scikit-learn [25].
[8]https://github.com/megagonlabs/sato/tree/master/table_data

5.4 Experiment Settings

We trained RECA for 20 epochs on each fold of the Semtab2019 dataset and trained it for 15 epochs on each fold of the WebTables dataset. We used Adam optimizer [20], and we adopted spaCy [13] as the named entity tagging tool, which has 18 basic named entity types. After extending the named entity type set as shown in Section 4.1, we obtained 24 named entity types. The set of learning rates to be selected was $\{0.00005, 0.00001, 0.000005\}$. The batch size was selected from $\{8, 16, 32\}$. The weight decay of 0.01 was used to optimize the training process. The Jaccard threshold δ was set to 0.1. Following the settings of the classification module in the Huggingface [34] implementation of BERT [9], we set the dropout rate of the classification module to be 0.3. We followed a consistent model selection method with [30], where the best model was selected based on the performance on the validation set. The means and standard deviations of the F1 scores are reported in Table 4. We conducted all the experiments on Intel(R) Xeon(R) Gold 5220R @ 2.20GHz CPUs and GeForce RTX 3090 GPUs.

5.5 Experiment Results

We conducted a 5-fold cross-validation to evaluate the performance of RECA in comparison with the baselines. The mean F1 scores of the 5-fold are reported. ± denotes the standard deviation recorded for the F1-scores in 5-fold.

As shown in Table 4, Sherlock [15] is significantly outperformed by RECA. The relatively poor performance of Sherlock can be attributed to the fact that Sherlock encodes the target column based on the cells in the target column only. Besides, Sherlock encodes the columns with statistical and simple semantic features, which have limited expressiveness to capture the rich semantics in tables.

We further noticed that TaBERT [35] performs poorly for macro average F1 scores on both datasets, which coincides with the experimental results shown by [37]. Besides, its performance for support-weighted F1 scores is also outperformed by RECA. As suggested by [37], the main reason for this phenomenon is that TaBERT focuses on table semantic parsing. The column embeddings generated by TaBERT are more suitable for understanding the alignment between the input text and table schema, while they are less powerful for column type annotation.

We also observed that RECA outperforms the state-of-the-art models TABBIE [16] and DODUO [30] for both support-weighted F1 scores and macro average F1 scores on the two datasets. Specifically, RECA outperforms TABBIE by 6.8% and 11.0% for the support-weighted F1 score and macro average F1 score, respectively, on the Semtab2019 dataset. For the WebTables dataset, RECA outperforms TABBIE by 0.9% and 6.7% for the two F1 scores. Furthermore, RECA achieves performance uplifts of 4.0% and 7.0% for the F1 scores on the Semtab2019 dataset and improvements of 1.0% and 5.5% on the WebTables dataset. Note that both TABBIE and DODUO utilize LMs to process intra-table context while ignoring the inter-table context information when generating the embeddings of the target columns. However, RECA mainly focuses on extracting useful

Table 4: Experimental results on the Semtab2019 dataset and the WebTables dataset.

	Semtab2019 dataset		WebTables dataset	
Model names	Support-weighted F1	Macro average F1	Support-weighted F1	Macro average F1
Sherlock [15]	0.646 ± 0.006	0.440 ± 0.009	0.844 ± 0.001	0.670 ± 0.010
TaBERT [35]	0.768 ± 0.011	0.413 ± 0.019	0.896 ± 0.005	0.650 ± 0.011
TABBIE [16]	0.799 ± 0.013	0.607 ± 0.011	0.929 ± 0.003	0.734 ± 0.019
DODUO [30]	0.820 ± 0.009	0.630 ± 0.015	0.928 ± 0.001	0.742 ± 0.012
RECA *target only*	0.808 ± 0.017	0.586 ± 0.039	0.911 ± 0.001	0.688 ± 0.014
RECA *w/o re*	0.836 ± 0.012	0.641 ± 0.037	0.927 ± 0.001	0.748 ± 0.024
RECA *w/o sub*	0.848 ± 0.009	0.650 ± 0.019	0.936 ± 0.002	0.774 ± 0.011
RECA	**0.853** ± 0.005	**0.674** ± 0.007	**0.937** ± 0.002	**0.783** ± 0.014

inter-table context information to enhance the embeddings of the target columns. The performance uplifts of RECA over the state-of-the-art models show that the inter-table context information serves as a suitable replacement for the intra-table context information when generating the embeddings for the target columns. Besides, by incorporating inter-table context information in replacement of the intra-table context information, RECA can naturally handle wide tables. Furthermore, we noticed that the performance boost for macro average F1 scores is significant, which suggests that RECA is especially good at improving the performance for less-populated semantic types by incorporating useful inter-table context information.

5.6 Ablation Study

To better understand how different components of RECA benefit the model in annotating the column semantic type, we conducted an extensive ablation study to evaluate the performance of RECA and its following variants:

- RECA *target only*: The RECA model that only considers the target column without processing related and sub-related tables. This variant degenerates to the basic BERT model that only processes the cells in the target column.
- RECA *w/o re*: The RECA model that only considers the target column and the identified columns in the sub-related tables, without processing related tables.
- RECA *w/o sub*: The RECA model that only considers the target column and the identified columns in the related tables, without processing sub-related tables.

We present the results of RECA *target only* to establish a baseline performance for processing the content of the target column only, so as to analyze the importance of related and sub-related tables in generating target column representations. Then, we present the results of RECA *w/o re* and RECA *w/o sub* in order to see if related and sub-related tables provide useful context information that boosts the annotation quality of RECA.

As shown in Table 4, the performance of RECA drops significantly when both related tables and sub-related tables are removed (RECA *target only*). On the Semtab2019 dataset, 5.3% and 13.1% of the performance drops can be observed for the two metrics, while on the WebTables dataset, the F1 scores drop by 2.8% and 12.1%. These drops demonstrate the importance of incorporating inter-table context information when generating target column representations. To further understand the effects of incorporating related and sub-related tables, we evaluated the performances of RECA *w/o re* and RECA *w/o sub*. Note that the performances of RECA drop slightly when the sub-related tables are removed: 0.9% and 3.6% on the Semtab2019 dataset; 0.1% and 1.1% on the WebTables dataset. But the performance drops are more prominent when the related tables are removed: 2.0% and 4.9% on the Semtab2019 dataset; 1.1% and 4.5% on the WebTables dataset. Since related tables are 'more related' compared with sub-related tables in terms of the similarity in the named entity schema, removing sub-related tables causes less harm to the model performance than removing related tables, as is shown in the experimental results. We further notice that the performance drop on the macro average F1 metric is much larger than that on the support-weighted F1 metric, which suggests that the related and sub-related table information can improve the annotation quality of RECA on the less-populated semantic types.

Table 5: Learning Efficiency evaluation on the Semtab2019 dataset and the WebTables dataset.

Datasets	[%]	Support-weighted F1	Macro average F1
Semtab2019	25	0.697 ± 0.041	0.442 ± 0.074
Semtab2019	50	0.792 ± 0.020	0.566 ± 0.045
Semtab2019	75	0.820 ± 0.021	0.631 ± 0.047
Semtab2019	100	**0.853** ± 0.005	**0.674** ± 0.007
WebTables	25	0.909 ± 0.002	0.680 ± 0.008
WebTables	50	0.924 ± 0.004	0.738 ± 0.019
WebTables	75	0.930 ± 0.002	0.772 ± 0.013
WebTables	100	**0.937** ± 0.002	**0.783** ± 0.014

5.7 Efficiency

Learning Efficiency We further evaluated the effectiveness of RECA with less training data. We trained RECA with training sizes (25%, 50%, 75%, 100% of the full training set). As shown in Table 5, when using 75% of the training data, the F1 scores of RECA dropped by only 0.7% and 1.4% compared with training with the full training set on the WebTables dataset, while the F1 scores of RECA dropped by 3.9% and 6.4% compared with training with the full

Table 6: Data Efficiency evaluation on the Semtab2019 dataset and the WebTables dataset.

Datasets	Max	Support-weighted F1	Macro average F1
Semtab2019	8	0.540 ± 0.009	0.319 ± 0.010
Semtab2019	16	0.654 ± 0.013	0.436 ± 0.006
Semtab2019	32	0.728 ± 0.010	0.507 ± 0.020
Semtab2019	128	0.816 ± 0.017	0.620 ± 0.033
Semtab2019	256	0.851 ± 0.011	0.662 ± 0.024
Semtab2019	512	**0.853** ± 0.005	**0.674** ± 0.007
WebTables	8	0.907 ± 0.004	0.737 ± 0.011
WebTables	16	0.923 ± 0.002	0.762 ± 0.011
WebTables	32	0.931 ± 0.002	0.780 ± 0.010
WebTables	128	**0.937** ± 0.002	**0.783** ± 0.014
WebTables	256	0.936 ± 0.003	**0.783** ± 0.020
WebTables	512	0.936 ± 0.001	0.780 ± 0.011

training set on the Semtab2019 dataset. RECA with 75% training data still slightly outperformed the state-of-the-art methods TABBIE [16] and DODUO [30] on two datasets. While training RECA with 50% of the data, RECA achieves slightly worse performance than the state-of-the-art models TABBIE and DODUO. When RECA is trained with only 25% of the training data, it still outperforms Sherlock [15]. The learning efficiency results show that RECA has the advantage of training a high-performance model with a relatively small amount of training data.

Data Efficiency As suggested by [30], input data efficiency is an important issue worth considering when designing column semantic type annotation models. Due to the maximum sequence length limit of BERT, annotating columns accurately with only a small number of tokens is a crucial factor in evaluating the performance and robustness of column semantic type annotation models.

We conducted experiments on two datasets with different maximum sequence length limits to evaluate the data efficiency of RECA. As shown in Table 6, there is not much difference when the maximum sequence length limit is set to 128, 256, and 512 on the WebTables dataset. While the performance dropped by only 0.2% and 1.8% when set from 512 to 256 on the Semtab2019 dataset. We noticed that RECA still outperforms all the state-of-the-art methods on the WebTables dataset even if we reduce the maximum sequence length to 32, while still achieving comparable performance to the state-of-the-art approaches even if we further reduce the maximum sequence length to 16. This phenomenon demonstrates the high data efficiency of RECA. Besides, it also shows the robustness of RECA when dealing with data with small input size.

5.8 Discussion

We analyzed the effectiveness of the column location alignment requirement mentioned in Section 4.3 to see if aligning identified columns at the exact same location as the target column is indeed an effective strategy. Following the notations in Section 4.3, we compared RECA with the following alignment variants:

- RECA *set*: If not aligned by the exact alignment but if any $\Psi(\hat{e}_k^j) = \Psi(\hat{e}_i^t)$, $j = 1, 2, ..., M_k$ then the j-th column is the identified column. In case multiple j exist, select the one that is closer to t.
- RECA *prox*: If not aligned by the exact alignment but $\Psi(\hat{e}_k^{t-1}) = \Psi(\hat{e}_i^t)$ or $\Psi(\hat{e}_k^{t+1}) = \Psi(\hat{e}_i^t)$, then the t-1-th column or t+1-th column is the identified column. In case a tie happens, regard both columns as the identified columns.

Table 7: Alignment strategies evaluation on the Semtab2019 dataset and the WebTables dataset.

Datasets	F1	RECA *set*	RECA *prox*	RECA
Semtab2019	S	0.842 ± 0.008	0.843 ± 0.004	**0.853** ± 0.005
	M	0.660 ± 0.014	0.663 ± 0.009	**0.674** ± 0.007
WebTables	S	0.934 ± 0.003	0.935 ± 0.004	**0.937** ± 0.002
	M	0.778 ± 0.016	0.780 ± 0.017	**0.783** ± 0.014

S stands for support-weighted F1 scores; M stands for macro average F1 scores.

As shown in Table 7, the support-weighted F1 score of RECA drops by 1.2% and 1.3%, while the macro average F1 score drops by 1.6% and 2.1% when applying RECA *prox* and RECA *set* alignment strategies, respectively on the Semtab2019 dataset. Slight performance drops can also be observed on the WebTables dataset. We noticed that the performance drop when applying the RECA *set* alignment strategy is larger than applying the RECA *prox* alignment strategy, which is understandable since the requirements of the RECA *prox* strategy are stricter than that of the RECA *set* strategy: the shift of the identified column index from the target column index cannot be larger than 1. We attribute the drop in performance to the reason that by applying alignment strategies that are 'looser', some noisy columns that should not be aligned may be included as the inter-table context and thus mislead the model. To this end, we conclude that the column location alignment requirement is necessary since it considers the inner schema structures of two tables.

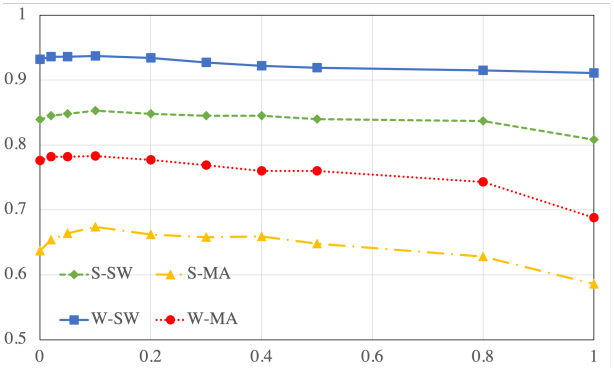


Figure 5: Sensitivity analysis of RECA on different values of the Jaccard threshold δ. The x-axis denotes the values of δ. The y-axis denotes the values of F1 scores. S-SW and S-MA stand for the support-weighted and macro average F1 scores on the Semtab2019 dataset; W-SW and W-MA stand for the support-weighted and macro average F1 scores on the WebTables dataset.

Fans' Rank	Title	Year	Director(s)	Overall Rank
201	The Grand Illusion	1937	Jean Renoir	90
202	Eternal Sunshine of the Spotless Mind	2004	Michel Gondry	176
203	Cinema Paradiso	1988	Giuseppe Tornatore	192
204	American Graffiti	1973	George Lucas	198
205	Traffic	2000	Steven Soderbergh	713
...	...	...	...	...

N	Title	Year	Director	IMDB Rating	IMDB Votes
1	The Social Network	2010	David Fincher	9.3	45993
2	Inception	2010	Christopher Nolan	8.9	333261
3	Alice in Wonderland	2010	Tim Burton	6.6	84086
4	Shutter Island	2010	Martin Scorsese	8.0	160301
5	How to Train Your Dragon	2010	Dean DeBlois	8.2	68159
...	...	...	...	...	

Rank	Title	Year	Director(s)	Overall Rank
1	Singin' in the Rain	1952	Stanley Donen and Gene Kelly	6
2	Some Like It Hot	1959	Billy Wilder	10
3	Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb	1964	Stanley Kubrick	12
4	Annie Hall	1977	Woody Allen	25
5	The Graduate	1967	Mike Nichols	39
...	...	...	...	...

Date	Movie	Year	Director
2012-01-23	Blue Valentine	2010	Derek Cianfrance
2012-01-21	Dogtooth	2009	Giorgos Lanthimos
2012-01-20	Serendipity	2001	Peter Chelson
2012-01-19	Tinker Tailor Soldier Spy	2011	Tomas Alfredson
2012-01-15	Transit	2012	Antonio Negret
...	...	...	...

Figure 6: Top left is the target table, bottom left is one of the exemplars related tables, and the two tables on the right are the exemplar sub-related tables. The target column and the corresponding identified columns are marked in yellow. Note that the cells marked in grey are masked out from the model input during experiments. For ease of demonstration, we only present one related table and two sub-related tables in the figure.

The similarity of the inner schema structures should be considered when designing the column alignment strategy for RECA.

5.9 Parameter Sensitivity

We analyzed the impact of the key parameter of RECA: the Jaccard threshold δ in Section 4.2. δ is chosen from $\{0, 0.02, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.8, 1\}$. Figure 5 presents the results. We notice that the performance of RECA increases as we increase δ from 0 to 0.1 since larger δ filters out the topically irrelevant inter-table context and thus reduces the noise. The performance decreases as we further increase δ to 1 because if the Jaccard threshold δ is too large, useful inter-table context is filtered out. We observe that the performance is generally stable for δ in the range of $[0, 0.3]$. In practice, we suggest finding the optimal values of δ based on the dataset.

5.10 Qualitative Evaluation

To better understand how inter-table context helps improve the annotation quality of column semantic types, we further conducted a qualitative evaluation of RECA and its state-of-the-art counterparts.

On the top left corner of Figure 6, we present Table #8 in the testing set of the Semtab2019 dataset. This table is about the rankings of films, the year of production, and the corresponding directors of the films. The second column is the target column in this table, and the ground truth label is **Film**. The intra-table content could be misleading since other similar semantic types describing WORK_OF_ART, such as **TelevisionEpisode** and **Musical**, can have very similar table content. TABBIE [16] and DODUO [30] are likely to be influenced by the intra-table context information since both of them annotate it as **TelevisionEpisode**. However, RECA finds the related tables and sub-related tables of the target tables and processes the identified columns as inter-table context information to provide more information for the semantic type of the target column. As shown in Figure 6, the bottom left shows one of the related tables, and the right shows two of the sub-related tables. We notice that the identified columns in the related tables and sub-related tables are all film names. By incorporating more film names as context information to RECA, the chance that it correctly annotates the target column as **Film** instead of other WORK_OF_ART types increases. Therefore, the inter-table context information can serve as an enhancement for the content in the target table. At the same time, it eliminates noisy information brought by the context in the target table.

6 CONCLUSION

In this paper, we propose RECA, an inter-table context enhanced column semantic type annotation framework based on pre-trained language models. Extensive experiments on two web table datasets demonstrated the effectiveness of RECA in providing high-quality annotations for table columns. Specifically, RECA achieves new state-of-the-art performance on the two datasets. Through our evaluation, we also confirm that the inter-table context information can be a powerful replacement for the intra-table context information, thus giving the model the ability to annotate wide tables. Furthermore, we have shown that RECA is learning efficient and input data efficient. As a future direction, we would like to explore the effectiveness of inter-table context on other table understanding tasks, such as column relation prediction and row population.

ACKNOWLEDGMENTS

Many thanks to Chengmin Wu for her valuable input. This work is partially supported by National Science Foundation of China (NSFC) under Grant No. U22B2060, the Hong Kong RGC GRF Project 16213620, RIF Project R6020-19, AOE Project AoE/E-603/18, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants MHX/078/21 and PRP/004/22FX, Microsoft Research Asia Collaborative Research Grant and HKUST-Webank joint research lab grants.

REFERENCES

[1] 2019. Sherlock. Retrieved Feb 1, 2023 from https://github.com/mitmedialab/sherlock-project

[2] 2020. TaBERT. Retrieved Feb 1, 2023 from https://github.com/facebookresearch/TaBERT

[3] 2021. TABBIE. Retrieved Feb 1, 2023 from https://github.com/SFIG611/tabbie

[4] 2022. DODUO. Retrieved Feb 1, 2023 from https://github.com/megagonlabs/doduo

[5] Arturs Backurs and Piotr Indyk. 2015. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, New York, NY, USA, 51–58. https://doi.org/10.1145/2746539

[6] Vincenzo Cutrona, Federico Bianchi, Ernesto Jiménez-Ruiz, and Matteo Palmonari. 2020. Tough tables: Carefully evaluating entity linking for tabular data. In *International Semantic Web Conference*. Springer, Cham, Switzerland, 328–343. https://doi.org/10.1007/978-3-030-62466-8_21

[7] Vincenzo Cutrona, Jiaoyan Chen, Vasilis Efthymiou, Oktie Hassanzadeh, Ernesto Jiménez-Ruiz, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, Daniela Oliveira, et al. 2022. Results of SemTab 2021. *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* 3103 (2022), 1–12.

[8] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. TURL: table understanding through representation learning. *Proceedings of the VLDB Endowment* 14, 3 (2020), 307–319.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. ACL, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[10] Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference*. ACM, New York, NY, USA, 2736–2742. https://doi.org/10.1145/3308558.3313629

[11] Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. ACL, 1315–1325. https://doi.org/10.18653/v1/N19-1133

[12] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly Supervised Table Parsing via Pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL, 4320–4333. https://doi.org/10.18653/v1/2020.acl-main.398

[13] Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear* 7, 1 (2017), 411–420.

[14] Kevin Hu, Snehalkumar'Neil'S Gaikwad, Madelon Hulsebos, Michiel A Bakker, Emanuel Zgraggen, César Hidalgo, Tim Kraska, Guoliang Li, Arvind Satyanarayan, and Çağatay Demiralp. 2019. Viznet: Towards a large-scale visualization learning and benchmarking repository. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, Article 662, 12 pages. https://doi.org/10.1145/3290605.3300892

[15] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, New York, NY, USA, 1500–1508. https://doi.org/10.1145/3292500.3330993

[16] 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*. ACL, 3446–3456. https://doi.org/10.18653/v1/2021.naacl-main.270

[17] Paul Jaccard. 1912. The distribution of the flora in the alpine zone. 1. *New phytologist* 11, 2 (1912), 37–50.

[18] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Kavitha Srinivas, and Vincenzo Cutrona. 2020. Results of semtab 2020. In *CEUR Workshop Proceedings*, Vol. 2775. 1–8.

[19] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444

[20] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint* (2014), 5. arXiv:1412.6980

[21] Aneta Koleva, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. 2021. Generating Table Vector Representations. *arXiv preprint* (2021), 5. arXiv:2110.15132

[22] Erich Leo Lehmann, Joseph P Romano, and George Casella. 2005. *Testing statistical hypotheses*. Vol. 3. Springer, New York, NY, USA.

[23] Frederic P Miller, Agnes F Vandome, and John McBrewster. 2009. Levenshtein distance: Information theory, computer science, string (computer science), string metric, damerau? Levenshtein distance, spell checker, hamming distance.

[24] Renée J Miller. 2018. Open data integration. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2130–2139.

[25] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[26] Minh Pham, Suresh Alse, Craig A Knoblock, and Pedro Szekely. 2016. Semantic labeling: a domain-independent approach. In *International Semantic Web Conference*. Springer, Cham, Switzerland, 446–462. https://doi.org/10.1007/978-3-319-46523-4_27

[27] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal* 10, 4 (2001), 334–350.

[28] Vijayshankar Raman and Joseph M Hellerstein. 2001. Potter's wheel: An interactive data cleaning system. In *VLDB*, Vol. 1. Morgan Kaufmann, San Francisco, CA, USA, 381–390.

[29] S Krishnamurthy Ramnandan, Amol Mittal, Craig A Knoblock, and Pedro Szekely. 2015. Assigning semantic labels to data sources. In *European Semantic Web Conference*. Springer, Cham, Switzerland, 403–417. https://doi.org/10.1007/978-3-319-18818-8_25

[30] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data*. ACM, New York, NY, USA, 1493–1503. https://doi.org/10.1145/3514221.3517906

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[32] Petros Venetis, Alon Y Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, and Gengxin Miao. 2011. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment* 4, 9 (2011), 528–538.

[33] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: Table Convolutional Network for Web Table Interpretation. In *Proceedings of the Web Conference 2021*. ACM, New York, NY, USA, 4020–4032. https://doi.org/10.1145/3442381.3450090

[34] 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* (2019), 1. arXiv:1910.03771

[35] 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*. ACL, 8413–8426. https://doi.org/10.18653/v1/2020.acl-main.745

[36] Dan Zhang, Madelon Hulsebos, Yoshihiko Suhara, Çağatay Demiralp, Jinfeng Li, and Wang-Chiew Tan. 2020. Sato: contextual semantic type detection in tables. *Proceedings of the VLDB Endowment* 13, 12 (2020), 1835–1848.

[37] Yiwei Zhou, Siffi Singh, and Christos Christodoulopoulos. 2021. Tabular Data Concept Type Detection Using Star-Transformers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, New York, NY, USA, 3677–3681. https://doi.org/10.1145/3459637.3482197