GitTables: A Large-Scale Corpus of Relational Tables

MADELON HULSEBOS[*], University of Amsterdam, Netherlands
ÇAĞATAY DEMIRALP, Sigma Computing, United States
PAUL GROTH, University of Amsterdam, Netherlands

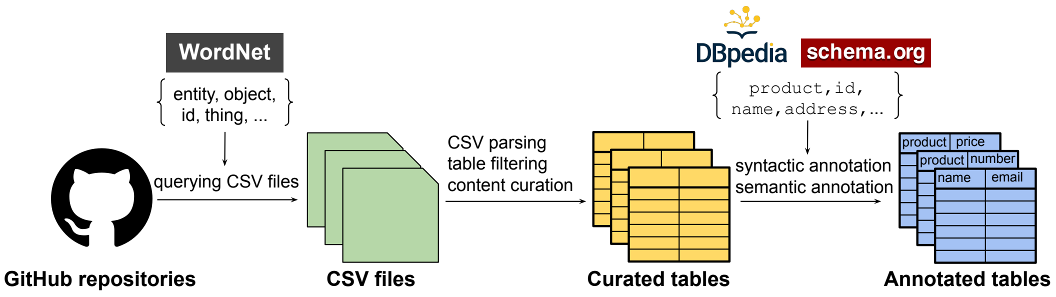


Fig. 1. The pipeline for creating GitTables consists of 1) querying CSV files from GitHub repositories based on WordNet, 2) parsing CSV files and curating tables, and 3) annotating tables with column semantics from DBpedia and Schema.org.

The success of deep learning has sparked interest in improving relational table tasks, like data preparation and search, with table representation models trained on large table corpora. Existing table corpora primarily contain tables extracted from HTML pages, limiting the capability to represent offline database tables. To train and evaluate high-capacity models for applications beyond the Web, we need resources with tables that resemble relational database tables. Here we introduce GitTables, a corpus of 1M relational tables extracted from GitHub. Our continuing curation aims at growing the corpus to at least 10M tables. Analyses of GitTables show that its structure, content, and topical coverage differ significantly from existing table corpora. We annotate table columns in GitTables with semantic types, hierarchical relations and descriptions from Schema.org and DBpedia. The evaluation of our annotation pipeline on the T2Dv2 benchmark illustrates that our approach provides results on par with human annotations. We present three applications of GitTables, demonstrating its value for learned semantic type detection models, schema completion methods, and benchmarks for table-to-KG matching, data search, and preparation. We make the corpus and code available at https://gittables.github.io.

CCS Concepts: • **Information systems** → *Relational database model*; *Information integration*; • **Computing methodologies** → *Information extraction*; *Neural networks.*

Additional Key Words and Phrases: relational tables, data management, deep learning, datasets

ACM Reference Format:
Madelon Hulsebos, Çağatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1, Article 30 (May 2023), 17 pages. https://doi.org/10.1145/3588710

[*]Corresponding author (m.hulsebos@uva.nl)

Authors' addresses: Madelon Hulsebos, University of Amsterdam, Amsterdam, 1012 WX, Netherlands, m.hulsebos@uva.nl; Çağatay Demiralp, Sigma Computing, San Francisco, CA 94105, United States, cagatay@sigmacomputing.com; Paul Groth, University of Amsterdam, Amsterdam, 1012 WX, Netherlands, p.t.groth@uva.nl.

2836-6573/2023/5-ART30
https://doi.org/10.1145/3588710

1 INTRODUCTION

Deep learning (DL) models, in the past decade, have dramatically improved many long-standing computer vision and natural language processing tasks [25]. The practical success of DL has also spurred interest in its applications to tasks in other domains, including data management, ranging from data cleaning to annotation [45]. To train DL models for relational data, earlier work primarily relied on corpora consisting of tables scraped from HTML pages [10] such as WDC WebTables [46], the largest table corpus to date. These models have played an important role in facilitating data-driven data management research, with Web applications in particular [7, 49].

However, tables extracted from HTML pages on the Web (Web tables) provide a skewed representation of tables in the wild especially those residing in (enterprise) databases [24, 27]. The common attribute "ID", for example, does not appear among the twenty most frequent column names of the WDC WebTables corpus, and the dimensions of Web tables are significantly smaller than typical database tables [46]. It is therefore unsurprising that models based on Web tables have limited applications beyond the Web [7]. To broaden the impact of data-driven data management research, we need new table collections complementing existing corpora with tables resembling typical database tables.

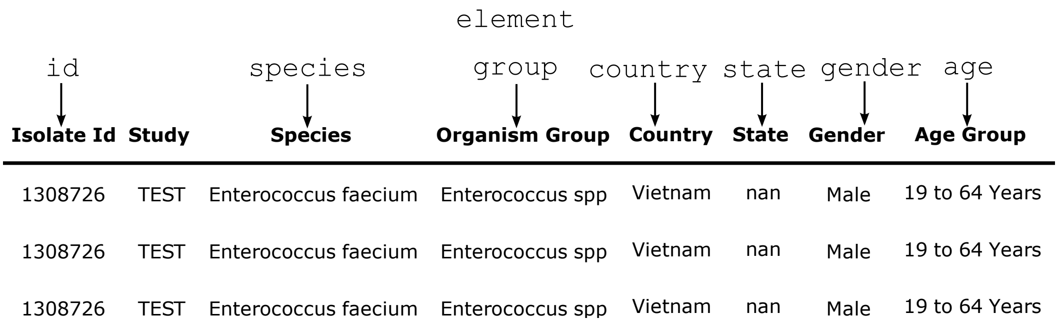
To address this demand, we introduce GitTables: a corpus with 1M relational tables extracted from CSV files in GitHub repositories. We will keep the corpus growing to have at least 10M tables to facilitate the extension of deep transfer learning to the relational domain, analogous to how large corpora of natural language have stimulated pretrained language models like BERT [16] and GPT [6]. We expect GitTables to stimulate similar progress in data management tasks like data search and preparation.

Besides having representative tables, many data management applications benefit from understanding the semantics of table columns. An intelligent data exploration system, for example, could recommend a map chart to visualize two data columns representing countries and sales revenues. Upon encountering missing values, the system could also automatically fill the gaps using a knowledge base. The system could, for instance, impute "France" in an empty cell in a column with country values, if the neighboring cell in a column with capital cities is "Paris".

To facilitate such functionalities, we enhance table columns with semantic types from DBpedia and Schema.org (e.g. `address`) using a syntactic as well as semantic annotation method (an example in Figure 2). Each semantic type is also associated with the expected atomic data type (e.g. `string` and `number`), a description, and hierarchical type relations. These semantically rich annotations together with DL present a unique opportunity to learn table semantics as demonstrated with TURL [15] and TaBERT [48].

Our analysis of GitTables confirms the different nature of the tables: the tables have significantly larger dimensions (rows and columns). A machine learning model for detecting data shift between GitTables and Web tables accurately distinguishes table columns from each source corpus. This reflects the structurally different content of these corpora. The semantic type distribution also deviates significantly from the semantic distribution of Web tables, illustrating its distinctive topical coverage. We make GitTables available through https://gittables.github.io. We contribute:

(1) GitTables, a new large-scale corpus of 1M tables. To the best of our knowledge, GitTables is the first large-scale relational table corpus with topical coverage and content structurally different than tables extracted from HTML pages.
(2) A scalable column-annotation method using distant-supervision. We annotate the columns in GitTables with semantics consisting of semantic types, atomic data types, hierarchical relations, and descriptions, making GitTables the largest annotated table corpus to date.


id
species
element group
country
state
gender
age

Isolate Id	Study	Species	Organism Group	Country	State	Gender	Age Group
1308726	TEST	Enterococcus faecium	Enterococcus spp	Vietnam	nan	Male	19 to 64 Years
1308726	TEST	Enterococcus faecium	Enterococcus spp	Vietnam	nan	Male	19 to 64 Years
1308726	TEST	Enterococcus faecium	Enterococcus spp	Vietnam	nan	Male	19 to 64 Years

Fig. 2. Snippet of an annotated table in GitTables. Annotations are provided with confidence scores. For example, the confidence of the type `species` for the column “Species” is 1, while `element group` for “Organism Group” is 0.7. These scores enable filtering based on the intended use case.

Table 1. Existing large-scale relational table corpora in comparison with GitTables which has considerably higher dimensions, resembling actual tables in databases. We report statistics related to relational tables, leaving out, for example, entity tables.

Name	Table source	# tables	Avg # rows	Avg # cols
WDC WebTables [26]	HTML pages	90M	11	4
Dresden Web Table Corpus [17]	HTML pages	59M	17	6
WikiTables [3]	Wikipedia tables	2M	15	6
Open Data Portal Watch [29]	CSVs from Open Data portals	107K	365	14
VizNet [20]	WebTables, Plotly, i.a.	31M	17	3
GitTables	**CSVs from GitHub**	**1M**	**142**	**12**

(3) Three applications demonstrating the value of GitTables: 1) we train a semantic type detection model and obtain high prediction accuracy, 2) we use GitTables as a resource for schema completion methods in typical database contexts, and 3) we present a starting point for benchmark datasets for typical data management tasks.

2 RELATED WORK

Web initiatives such as Common Crawl, Wikipedia, and Open Data have been cost-effective resources for curating unstructured and structured data at scale [3, 31, 32]. Below we discuss large-scale table corpora sourced from these initiatives and review prior work annotating column semantics of tables sampled from these corpora.

2.1 Large-scale table corpora

WDC WebTables and Dresden Web Table Corpus. These Web table corpora [17, 26] extract tables from HTML pages in the Common Crawl corpus as inspired by [8]. They provide an abundance of relational tables ranging from 59M to 90M and have been instrumental in advancing applications like table augmentation and integration [7, 49]. However, Web tables generalize poorly due to their small dimensions and different content [24].

WikiTables. To provide high-quality tables with semantics that are easier to detect than those of arbitrary Web tables, WikiTables extracts approximately 2M tables from Wikipedia [3]. This corpus is primarily suitable for tasks such as question answering that rely on the quality of the table contents. Unsurprisingly, the tables in WikiTables are just as small as those in WDC WebTables.

Table 2. Characteristics of annotated relational table datasets. Existing annotated corpora are limited in the number of (relational) tables and types, while GitTables provides a large-scale corpus annotated with over 2K semantic types.

Dataset	Table source	# tables	Avg # rows	Avg # cols	# types	Ontology
T2Dv2 [38]	WebTables	779	17	4	275	DBpedia
SemTab[1] [22]	WikiData, Wikipedia	132K	224	4	-	DBpedia
TURL [15]	WikiTables	407K	18	3	255	Freebase
GitTables	**CSV files on GitHub**	**962K**	**142**	**12**	**2.4K**	**DBpedia Schema.org**

[1] Statistics aggregated over all datasets included in the challenge.

Open Data Portal Watch. With 227K CSV files extracted from 260 Open Data portals [32], this is the first substantial corpus not sourced from HTML pages. 108K of these CSV files were parsed to tables and analysed based on their format, structure, and data type [29]. This analysis illustrates the different table dimensions and atomic data type distributions of such tables compared to Web table corpora, motivating the construction of a corpus like GitTables.

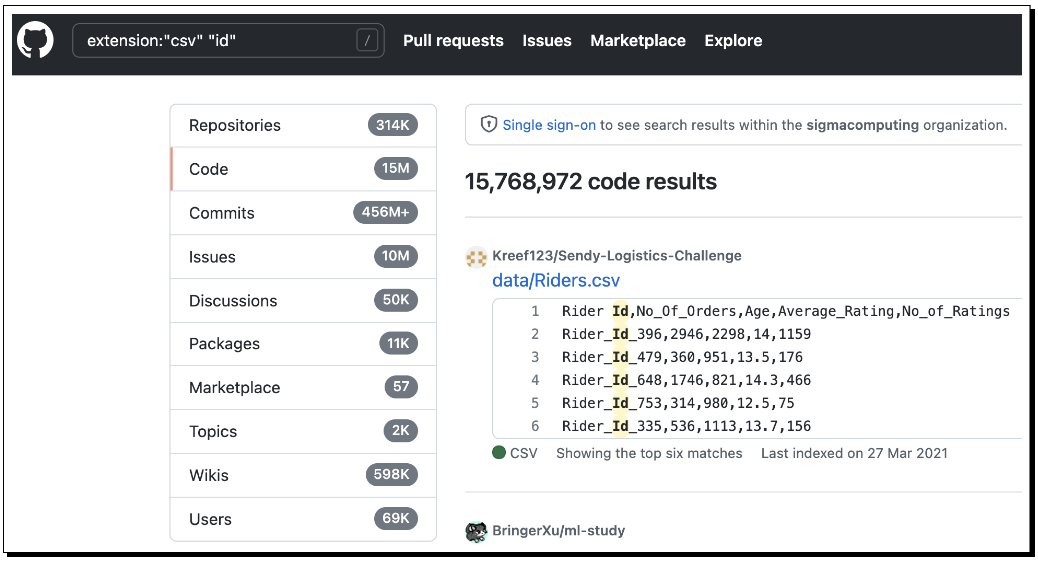
VizNet. VizNet was constructed to train and evaluate visualization methods with real-world tables [20]. It combines 31M tables from WebTables [8], ManyEyes [42], Plotly [34], and Open Data portals [32]. Analyses of VizNet suggest that tables not from the Web exhibit different internal structures, providing further evidence for differences between Web tables and tables from other sources.

2.2 Table datasets with column annotations

T2Dv2. This is a subset of WDC WebTables and was curated for benchmarking methods for knowledge base (KB) augmentation [37]. The rows, columns and tables were manually annotated with correspondences to DBpedia instances, properties and classes. This was found to be a trivial target for KB matching due to the many "obviously" linkable entities. Recent work also points out that Web tables might not be typical of tables used for KB augmentation [12].

SemTab. The SemTab challenge [22] provides datasets for benchmarking KG matching methods. Most of the tables were extracted from Wikidata and "refined" by adding for example noise. SemTab 2020 incorporates 180 large-sized Wikipedia tables which enriched with noise to mimic real tables [12]. Columns were annotated by linking column values to DBpedia types and aggregating these types to a column-level annotation. GitTables is a useful resource in future versions of the SemTab challenge for benchmarking table interpretation methods on database-like tables.

TURL. Inspired by pretrained language models (e.g., [6, 16, 36]), TURL [15] provides a framework for learning embedding representations of Web tables through a pretrained model. Pretrained models require fine-tuning with labeled data to be applied to specific tasks in domains. To do this, a set of tables from WikiTables is annotated with 255 semantic types from Freebase. Application of learned table representations for table understanding motivates the construction of a larger-scale and rich corpus to support this nascent line of research.



```
Rider Id,No_Of_Orders,Age,Average_Rating,No_of_Ratings
Rider_Id_396,2946,2298,14,1159
Rider_Id_479,360,951,13.5,176
Rider_Id_648,1746,821,14.3,466
Rider_Id_753,314,980,12.5,75
Rider_Id_335,536,1113,13.7,156
```

Fig. 3. The results when querying for CSV files with the term "id" as shown on GitHub its website. It illustrates the scale of CSV files (15M+ files for this particular query) and the database-like tables that are stored on GitHub.

3 GITTABLES

In this section, we summarize the design principles behind GitTables and describe the construction pipeline in detail.

3.1 Design principles of GitTables

Based on the gaps reported in the literature and our experience developing learned models for table interpretation tasks, we identified four criteria for GitTables:

C1 To facilitate learned table representations that generalize across different database contexts, the scale of the corpus should largely exceed the scale of the Open Data portals.

C2 To advance research beyond the Web, we need "database"-like tables which are considerably larger and more heterogeneous than typical Web tables.

C3 The corpus should have topical coverage and content that generalizes to enterprises, governments, and beyond.

C4 Tables should be enriched with semantic annotations to facilitate the development of learned models for data management tasks like data validation and preparation.

We considered different public interfaces to retrieve structured data files for extracting relational tables, like Zenodo and. Kaggle. On GitHub, we then performed a simple search for CSV files, yielding 92,191,141 files[1]. This suggests that GitHub can be an effective resource for collecting a relational table corpus at scale (criterion **C1**). We focused on the CSV format given its wide use for storing raw structured data [29], as reflected by the massive number of CSV files we found in our search.

[1]https://github.com/search?q=extension%3Acsv&type=Code (3 February 2022)

GitHub is commonly used by programmers, data scientists, and researchers, among others [23, 33]. Given the proximity of such users to actual databases, GitHub is a rich source for heterogeneous tables. Figure 3 shows the first CSV file retrieved by GitHub when it is queried for the term "id". This table with data about "riders" shows column names and table contents that are typical for database tables. This figure also illustrates the abundance of similar CSV files present on GitHub, as we can find close to 16M CSV files for this single query only. Prior analyses of CSV files from GitHub also found that these files have diverse formatting and the tables extracted from them have relatively large dimensions [23, 41], which is common across database contexts [24, 41]. Altogether, we consider CSV files from GitHub a suitable resource for database-like tables (**C2**).

Inspired by the construction of ImageNet [14], we select 67K unique English nouns from WordNet [40] yielding a set of diverse keywords (called "topics") to specify our search queries. Although it introduces some bias towards English tables, WordNet nouns have the desired topical coverage to ensure content diversity in GitTables (**C3**). To avoid the "WordNet effect" [4], we remove topics like "killing" which might yield tables with offensive content and are out of scope.

To satisfy criterion **C4**, we provide semantic annotations for table columns. We developed two annotation methods, one syntactic method informed by Sherlock [21] and the other leveraging a pretrained semantic model. We considered multiple ontologies to accommodate different use cases, and selected DBpedia [1] and Schema.org [19] as they are well-curated and provide complementary and diverse semantic types (as discussed further in Section 3.4).

Altogether, the high-level pipeline for constructing GitTables comes down to 1) extracting CSV files from GitHub, 2) parsing and curating tables from CSV files, and 3) annotating tables with column semantics. Figure 1 visualizes this pipeline.

3.2 Extracting CSV files from GitHub

GitHub restricts querying in multiple ways to avoid overloading its Search API. First, it is not possible to retrieve files larger than 438 kB which bounds the CSV files we extract in terms of their size. Although some organizations may use larger files, most CSV files in Open Data portals are found to be smaller than 100 kB [29]. We also observed repositories with larger tables that were split across multiple files (e.g. into daily snapshots), which may be recovered by unioning tables stemming from the same repository. A second restriction limits the resulting search responses to 1000 files. This restriction makes the process of extracting a massive set of CSV files nontrivial, as detailed below.

First, we construct an initial "topic query" for each topic from WordNet restricted to files with the CSV format. For example, we retrieve CSV files that contain the word "object" by the query `q="object" extension:csv`. We exclude results from forked repositories to minimize table duplication. We execute this initial topic query through the GitHub Search API and get the initial response size of this query representing the number of GitHub URLs pointing to CSV files containing the word "object".

Since the API restricts the number of files per query to 1000 and many topic queries return around 100K files, we segment the initial queries. We use the "size" qualifier to perform this segmentation, and generate sequences of file size ranges (in bytes) proportional to the number of files in the initial response. This results in segmented topic queries like `q="id" extension:csv size:50..100`, `q="id" extension:csv size:100..150`, and so on. We execute all segmented queries and collect the paginated responses, each of which contains approximately 1000 URLs. We traverse the paginated responses to extract all URLs for a given topic. We then iteratively write the raw contents pointed to by the URLs to CSV files.

3.3 Parsing and curating tables from CSV files

CSV parsing. Once we have the CSV files, we parse them to tables using the CSV parser from the Pandas library, a widely used data processing and analysis library for Python [28]. We leverage the integrated functionality of Python's Sniffer tool to determine the delimiter of the CSV files. We parse tables assuming that the first rows correspond to the header rows, as is conventional for CSV files [39]. Random samples of tables informed some exceptions to this; lines at the beginning of the file are skipped in case they are empty or start with a '#', which often indicates commented lines.

We remove rows in case they are considered "bad lines", such as empty lines, commented lines or lines with extra delimiters. Additional experiments revealed that some tables include separation characters without any values at the end of all rows, resulting in a mismatch of the number of attributes and the number of values per row. We realigned the header and table values in these cases by removing the redundant separation characters. We discard CSV files that could not be parsed with the aforementioned rules. In total, our parsing procedure results in 99.3% of the CSV files being parsed into tables.

Table filtering. We aim for GitTables to be a large-scale corpus of quality and relevant tables that can safely be used by the community. To this end, we first filter out tables from repositories without a license that allows the distribution of the repository contents. We find that roughly 16% of the tables are associated with such a license, which is consistent with earlier studies. In our final corpus, we only publish tables coming from repositories with such a license.

We further curate the corpus by removing *extremely small tables* [10], i.e. tables with less than two rows or two columns as these tables likely do not carry relevant data or were observed to contain singular text columns. We also remove tables if more than half of the column names are unspecified, or if any of the column names are not of the type `string` [29]. Lastly, to avoid inclusion of offensive content from social media platforms potentially stored in tables, as shown to be present in large collections of text extracted from web pages [2], we exclude tables of which a column name contains "twitter", "tweet", "reddit" or "facebook". Altogether, this curation pipeline filters out 9% of the tables.

Content curation. Besides offensive content from social media, tables might contain personal data that is undesirable to disseminate beyond GitHub. To minimize the spread of personal identifiable information (PII), we anonymize tables that likely contain PII data informed by the semantic types from Schema.org. We do so, by replacing column values annotated with any of the PII semantic types (annotated as described in Section 3.4) with fake values using the Faker library [18] as in Table 3. In case a column was annotated with the type `name`, we anonymize it only if it co-occurs with another PII semantic type as `name` does not necessarily indicate a person's name. In total 0.3% of the columns in GitTables contain fake values. Given the relatively limited number of affected columns, this anonymization procedure does not significantly change the underlying data distribution of GitTables.

3.4 Annotating tables with column semantics

We annotate table columns in GitTables with semantic types extracted from DBpedia and Schema.org to facilitate its use-cases in applications like data preparation as explained in Section 1. We provide rich metadata like hierarchical type relations, domains, and descriptions, if available, as well. This information can be exploited for training and evaluating models.

Semantic types. In total, we extracted 2831 properties from DBpedia that we use as semantic types. From Schema.org we included properties as well as types that together sum to 2637 semantic

Table 3. Semantic types associated with PII, percentage of columns annotated with each type (based on a subset), and Faker class used to generate fake column values.

Semantic type	**Percentage columns**	**Faker class**
`name`	2.202%	`faker.name`
`address`	0.163%	`faker.address`
`person`	0.068%	`faker.name`
`email`	0.042%	`faker.email`
`birth date`	0.017%	`faker.date`
`home location`	0.008%	`faker.city`
`birth place`	0.003%	`faker.postcode`
`postal code`	0.003%	`faker.city`

types. We provide annotations from both ontologies, a column may therefore be annotated with a semantic from DBpedia and one from Schema.org.

Most semantic types from DBpedia relate to domains like `Person`, `Place` or `PopulatedPlace` while types in Schema.org are more scattered across domains topped by `CreativeWork`, `Organization`, `Person`, and `Offer`. Along with the semantic types, we attach their metadata like hierarchical relations and descriptions which can be exploited in model evaluation and training. For example, one could adopt a loss or evaluation function for a semantic type prediction model that favors a less granular type (e.g. the type `place` for a ground-truth column of type `city`), instead of predicting an unrelated type (e.g. `size`). We provide the following metadata per semantic type if available:

(1) semantic column type in English, e.g. `id` and `name`,
(2) atomic type, e.g. `Number` and `Text`,
(3) domain, e.g. `address` has domain `Person` and `Organization`,
(4) superclass or superproperty, e.g. `product id` → `id`, and
(5) description, e.g. for `id`: "The identifier property represents any kind of identifier for any kind of Thing, such as ISBNs, GTIN codes, UUIDs."

Annotation. Informed by analyses of public CSV files [29], we preprocess the semantic types and table headers by replacing underscores and hyphens, splitting camel-cased combined words, and converting strings to lower case. Experiments showed that few column names with numbers were annotated with semantic types that coincidentally contain a number. For this reason, the annotation pipeline does not annotate column names that include numbers.

The original column names are useful indicators of what a column's data consists of and provide a proxy for human annotations given the involvement of humans in naming table columns. To provide relatively strict annotations, we therefore leverage the preprocessed column names directly and syntactically match them to semantic types in the ontologies [21]. We call this the syntactic annotation method.

Recent successes in language models create the opportunity to annotate columns taking semantics into account. We embed column names and semantic types using FastText [5], and match them to each other. We use the character-level n-gram FastText model pretrained on the Common Crawl corpus and take the match based on the highest cosine similarity

Although users can decide on a similarity threshold relevant to their tasks, we discard annotations with very low similarity scores to ensure that the annotations in GitTables are useful out of the box. We call this method the semantic annotation method.

Table 4. Distribution of atomic data types showing the difference in numeric versus string data between GitTables and WDC WebTables.

Atomic data type	**GitTables**	**WDC WebTables**
Numeric	57.9%	51.4%
String	41.6%	47.4%
Other	0.5%	1.2%

4 ANALYSIS

Here, we analyse the first version of GitTables of 1M tables over 97 topic subsets, among the larger sets: "thing", "object", and "id". Once complete, GitTables will consist of approximately 10M tables. Tables analysed in this paper are distributed as a separate version: GitTables 1M.

4.1 Corpus statistics

Table statistics. The total analysis set of 1,021,143 tables consists of 144,833,144 rows and 12,369,120 columns, on average 142 rows and 12 columns (see Tables 1 and 2 for comparisons across corpora). On average, tables consist of 1,038 cells. Figure 4a shows that the distribution of table dimensions has long tails over the number of rows and the number of columns. Overall these dimensions differ significantly from Web table corpora, which have 5 columns and 15 rows on average, and come closer to dimensions of typical database tables. As summarized in Table 4, almost 58% of the columns are inferred to have numeric values versus 41% textual, which differs from the roughly 50%-50% distribution in Web tables. Together, these statistics contribute to criteria C1 and C2.

The majority of the tables originate from distinct repositories as 75% of the source repositories contribute at most 5 tables. With an overall average of 34 tables per repository, a few repositories contribute a large number of tables in this first version of GitTables. Manual inspection revealed that such repositories contain snapshots of the same or similar databases[2]. These tables, and the corresponding source URL referring to the associated repository, can be used for constructing larger tables through unions and joins.

We observed the advantage of separating the tables from the query topics. For example, the query for "organism" tables retrieves many tables related to biological and medical entities, of which a typical one is shown in Figure 2. Such subsets can be leveraged for training domain-specific models, or to incorporate these topics as semantics for table embeddings.

Annotation statistics. The syntactic method annotates, on average across the two ontologies, 730K out of 1M tables with at least 1 column annotation. On average, this yields 2.7M annotations with 756 unique types. The semantic annotation approach annotated on average 960K tables, yielding a total of 8.4M column annotations across 2.4K unique types on average. Statistics with regard to the different ontologies (DBpedia and Schema.org) are presented in Table 5. Depending on the use case, users can select a suitable ontology and annotation method to filter the relevant annotations for each table.

If column context is of importance, for example for contextual models, a high table-annotation coverage is key. We find that the semantic method yields annotations for, on average, 71% of the table columns, while the syntactic method annotates 26%. Figure 4b shows the overall distribution of the percentage of annotated columns per table aggregated over both ontologies.

[2]For example, the repository: https://github.com/Emanuele-Falzone-PhD-Thesis/srbench

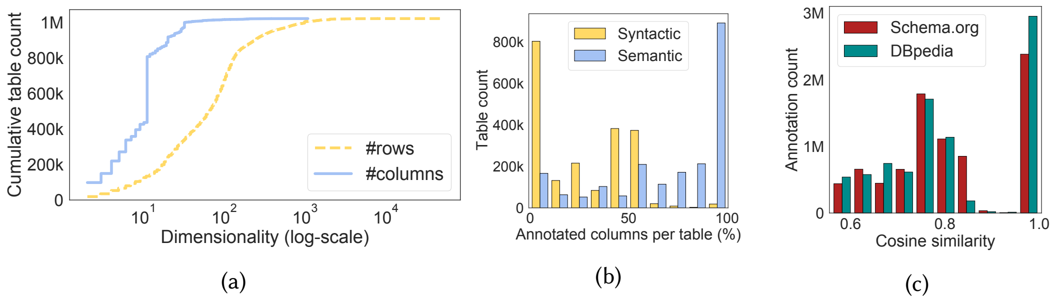


Fig. 4. (a) Cumulative table counts across table dimensions, illustrating the spread around the mean column count (12) and mean row count (142). (b) Percentage annotated columns per table, for each annotation method. The semantic method yields on average more annotations per table. (c) Cosine similarity of annotated columns (both ontologies). The peak at 1 relates to high syntactic resemblance of column names and types.

Table 5. Statistics of annotations by method and ontology.

Statistic	Syntactic		Semantic	
	DBpedia	**schema.org**	**DBpedia**	**schema.org**
# annotated tables	723K	738K	958K	962K
# annotated columns	2.9M	2.4M	8.5M	8.4M
# types	835	677	2.4K	2.4K
# types (#columns > 1K)	96	83	432	491

The cosine-similarity scores, that we attach with all semantic annotations, reflect the confidence of each annotation. From the distribution of these similarity scores as shown in Figure 4c, we observe that many annotations have a similarity score around 1, which indicates syntactic resemblance, while the remaining distribution is centered around 0.75. Users GitTables can set the desired threshold based on their needs and, for example, select only annotations with a high similarity score reflecting annotations with high confidence.

4.2 Corpus content

Table content and topical coverage. Beyond the structural properties of tables in GitTables, we compare their contents to tables from VizNet which combines most existing corpora [20]. We interpret the comparison of table contents as a data shift detection problem and evaluate if the data distributions significantly differ by training a domain classifier [35]. For this, we randomly sample 5K deduplicated columns from each corpus and extract 1,188 features as used for training semantic column type detection model Sherlock [21], capturing column-level statistics like column entropy and skewness, aggregations from word-embeddings, and aggregated character-level statistics (e.g. number of '@' characters per cell).

We then train a Random Forest classifier with default settings to separate whether a column originated from VizNet or from GitTables. Using a 10-Fold cross-validation setup, we find that this domain classifier is able to predict for 93% (±0.04) of the columns from which corpus it originates. This separability indicates the different data distributions in GitTables and VizNet, hence the complementary value of GitTables.

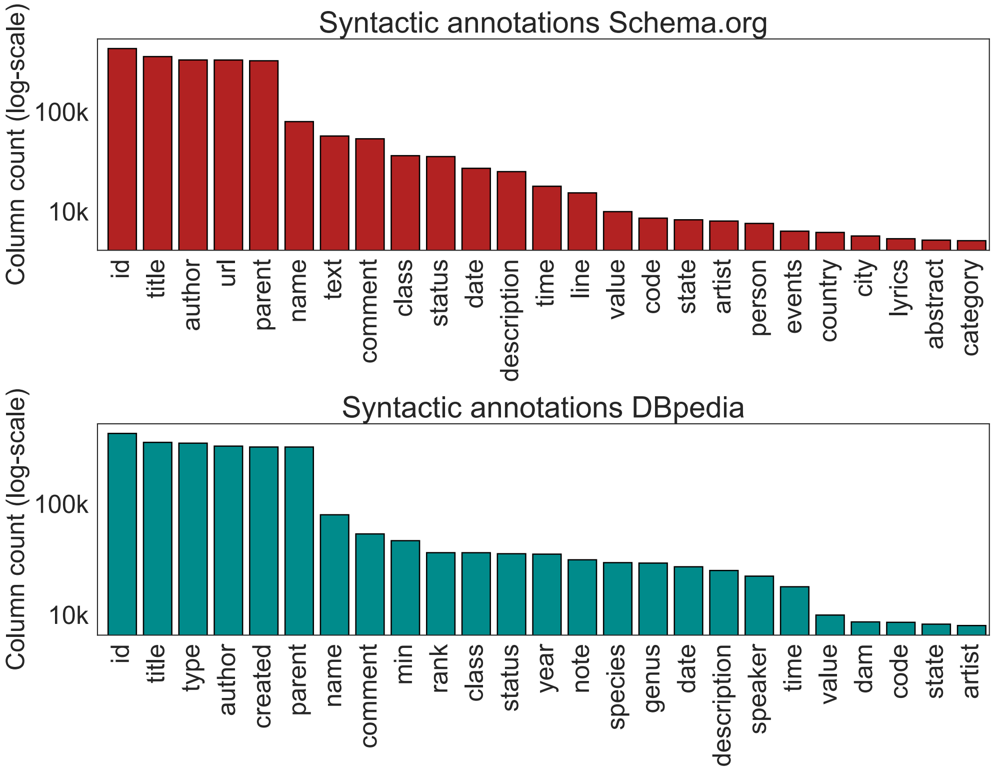


Fig. 5. Column annotation counts of top 25 column semantic types across the two annotation methods and two target ontologies, illustrating the semantic coverage and diversity of GitTables.

The columns in the WDC WebTables 2012 corpus have also been matched to DBpedia which resulted in the top 10 semantic types: `name`, `date`, `title`, `artist`, `description`, `size`, `type`, `location`, `model`, and `year`. For GitTables, the topical coverage and diversity of semantic types per annotation method and ontology is illustrated in Figure 5. Although top types in both corpora overlap such as `name` and `title`, we observe clear differences with top types in GitTables like `id`, `type`, `value`, and `min`. Especially the dominant type `id`, one of the most common types in typical databases, indicates that GitTables meets criteria C2 and C3.

Content biases. The analysis of the topical distribution present in GitTables shows a few semantic types that could potentially make methods informed by GitTables lean towards certain subpopulations, industries, or geographic areas. To further profile our dataset to this end, we adopted 2 categories proposed in [43], being “person” and “geography” and analyse a subset of tables on biases towards these categories.

In Table 6, we give a deeper understanding of distributions along these dimensions informed by column values associated with relevant semantic types from Schema.org. This analysis confirms that tables in GitTables containing geographic data (slightly over 1%), primarily represent English-speaking countries and cities. The value distribution of the few columns in GitTables with semantic types that reflect population segments, e.g. `gender`, `race`, and `ethnicity`, indicate a relatively higher concentration of data representing Western countries (see Table 6).

Table 6. Semantic types indicating subregions and subpopulations along with the most frequent column values. Tables with these types primarily represent Western and English-speaking regions and populations.

Semantic type	Percentage columns	Frequent values (most frequent first)
`country`	0.086%	United States[2], Canada, Belgium, Germany
`city`	0.056%	New York (X), London, Coquitlam, Cambridge
`gender`	0.040%	Male, Female, F, M
`ethnicity`	0.030%	French, Dutch, Spanish, Mexican
`race`	0.007%	Men, Human, White
`nationality`	0.003%	Hispanic, White, Caucasian (White)

[2] "United States" counts are merged with counts of "USA".

4.3 Annotation quality

As we have no access to the ground truth of semantic column types, we use the T2Dv2 benchmark hand-labeled with DBpedia types to evaluate our annotation approaches [38]. Although the annotation quality on T2Dv2 does not ensure the quality of annotations of GitTables, it is a good proxy for the annotation quality produced by our approaches. We consider table columns from files that we could parse and that were annotated by T2Dv2 as well as our annotation approaches. In total, we have 321 and 187 columns for evaluation with the semantic and syntactic approaches, respectively.

We find that our semantic approach yields in 54% (173 columns) the same annotation as T2Dv2. From the incorrect annotations, our approach annotated 47% (69 columns) with a DBpedia type that syntactically matches the column name as the corresponding similarity scores are 1.0. For example, our semantic approach annotated a column with cities (e.g. Pittsburgh, Buffalo) named "City" with the type `city` while T2Dv2 annotated this column with the less granular type `location`. This motivated a manual review of the 148 columns for which we find different annotations between the semantic approach and the annotations from T2Dv2[3]. Based on our manual review (n=3), we find that on average in 63 (± 14) out of 148 the semantic approach yields better annotations, in 37 (± 3) out of 148 the T2Dv2 annotations were clearly better. In 33 columns (± 14) they were just as good or bad, and undetermined in 15 cases (± 17).

The syntactic approach yields in 61% (114 columns) the same annotation as T2Dv2. At first, we found occurrences where the syntactic annotations were better. For example, a column with Latin names of birds named "Latin name" was annotated as `synonym` instead of DBpedia type `latin name`. We therefore followed the same manual review process as for the semantic approach to review 73 annotations where our pipeline yields annotations different from T2Dv2. Based on inter-annotator agreement we find that the syntactic approach clearly yields more accurate annotations for 21 columns and T2Dv2 had better annotations for 9 columns.

These findings indicate that our annotation approaches provide the necessary quality for training and evaluating DL models (C4). We also observe that the annotations of the T2Dv2 benchmark might require a review and potentially revision in future work.

5 APPLICATIONS

In this section, we demonstrate how GitTables can be used for models for semantic column type detection and schema completion, and discuss its use for data management benchmarks in

[3]The reviews can be found on https://gittables.github.io.

Table 7. F1 scores of semantic type detection models trained and evaluated on different corpora. These results suggest that models trained on GitTables can achieve high predictive performance, but that models trained on Web tables may not generalize to GitTables.

Train corpus	**Evaluation corpus**	**F1-score (macro)**
GitTables	GitTables	0.86
VizNet	VizNet	0.77
VizNet	GitTables	0.66

enterprise contexts. We note that the utility of GitTables stretches beyond these use cases to a wide range of data management tasks.

5.1 Semantic column type detection

Capturing the semantics of a table through its semantic column types is instrumental for many analytical tasks like data exploration as discussed in Section 1, but also for GDPR compliant data processing like anonymizing PII data as implemented for GitTables. Many enterprise databases however lack descriptive or standardized column names across tables. Column- and table representations learned from large table corpora have shown to accurately represent the semantics of Web tables [44], and we believe that these models easily extend to domains beyond the Web when trained on GitTables.

To detect the semantic types of columns in a given table, we train Sherlock [21], a deep learning model for semantic column type detection, on table columns from GitTables. For this experiment, we select five semantic types `address`, `class`, `status`, `name`, and `description` and randomly sampled 500 deduplicated columns per type from our analysis subset. For each column we extract the same low-level column features as used in Section 4.2, i.e. paragraph embeddings, word embeddings, character level counts, and aggregated statistics. We train Sherlock using a 5-Fold cross-validation setup. As Table 7 shows, this model achieves a macro F1 score of on average 0.86 (± 0.02) illustrating the utility of training a model on GitTables for semantic column type detection.

We repeat this experiment with columns from VizNet [20], which combines tables from all existing table corpora. To ensure a fair comparison, we sample 500 deduplicated columns per type of the exact same types. This model yields a macro F1 score of 0.77 (± 0.02) on samples from VizNet. However, when evaluating the VizNet-trained model on columns from GitTables, we find that it does not generalize well to samples from GitTables as the macro F1 score drops to 0.66 (± 0.03). This gap confirms that models trained on Web tables alone, may not generalize beyond the Web.

5.2 Schema completion

Inspired by Cafarella et al. [9], we use GitTables to inform completions of schema prefixes. Schema completion empowers many applications among which are general data- and knowledge base design, and data augmentation. We implement NearestCompletion as in Algorithm 1. Our algorithm completes a given (target) schema prefix of length N based on its similarity with schema prefixes from GitTables. We embed attributes from the target schema prefix of length N and schemas from GitTables using the Universal Sentence Encoder (USE) [11] which works well for multi-word attributes. Similar to how search engines provide multiple suggestions for search queries, we return the k "nearest completions". The nearest completions are based on schemas in GitTables with the smallest average cosine distance between the first N attributes of these schemas and attributes of the target prefix of length N.

Table 8. Suggested completions from GitTables (second column) for target schema prefixes from the CTU database, along with the cosine similarity between the original full schema of the target prefix and the full schema of the suggested completion. The completions are relevant within the context of the schema prefix as reflected by the cosine similarities around 0.5 on a range of [-1,1].

Header prefix	**Attributes from the nearest completion**	**Cosine similarity**
emp_no, birth_date, first_name	Title, TitleOfCourtesy, Address, HireDate, City	0.44
orderNumber, orderDate, requiredDate	ORDER_TRACKING_NUMBER, ORDER_TOTAL	0.50
WorkOrderID, ProductID, OrderQty	productType, inventoryId, articleId, productName	0.53

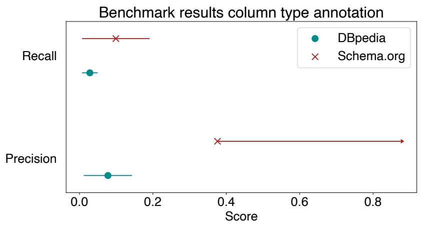
To evaluate the utility of GitTables for schema completion in typical industry contexts, we use schema prefixes from three actual table schemas from the CTU Prague Relational Learning Repository [30][4]. We extract prefixes of length $N = 3$ and return the $k = 10$ nearest completions as commonly used in information retrieval systems. The relevance of the suggested completions is calculated as the highest cosine similarity between USE embeddings of the original full schemas of the target prefix and the full schemas of the 10 suggested completions from GitTables. A few attributes of the most relevant completion are presented in the second column of Table 8.

```
Data: Schema prefix $p$, set of schemas $\mathcal{S}$ from GitTables, number of completions $k$
Result: Schema completions $c$
1  $N = |p|$
2  set of schema completions $C = \emptyset$
3  embedded schema prefix $p_e$ = [embed($a$) for attribute $a \in p$]
4  for schema $s \in \mathcal{S}$ do
5      embedded schema $s_e$ = [embed($a$) for attribute $a \in s$]
6      average prefix distance $d = \frac{\sum_{i=1}^{N} \text{cosine distance}(p_e[i], s_e[i])}{N}$
7      $C$.append($s$, $d$)
8  end
9  sort $C$ in ascending order of $d$
10 $c = C$.first($k$)
11 return $c$
```

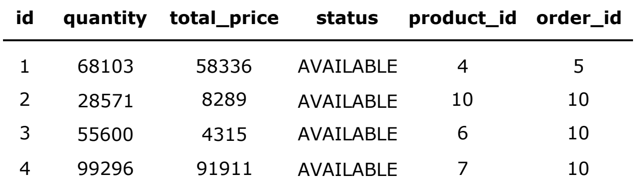
Algorithm 1: Schema completion procedure for returning k most relevant completions for an input schema prefix based on the cosine similarity with schema prefixes in GitTables.

Table 8 presents the target prefixes of the considered tables, attributes from the suggested completions, and the cosine similarities of the entire schemas. The semantics of these suggestions are clearly relevant to the target prefixes. For example, attributes from the most similar schema completion for the prefix from the "employees" table clearly relate to personal employee details such as "HireDate" and "Title". This relevance is also reflected by the cosine similarities between the entire original schemas and suggested schemas for the completions which, on average, is 0.49 on a scale of [-1,1]. These results underline the utility of GitTables as a resource for schema completion methods in typical databases.

[4]The "employees" table of the Employee database, the "orders" table from the ClassicalModels database, and the "WorkOrder" table from the AdventureWorks database.



(a)



id	quantity	total_price	status	product_id	order_id
1	68103	58336	AVAILABLE	4	5
2	28571	8289	AVAILABLE	10	10
3	55600	4315	AVAILABLE	6	10
4	99296	91911	AVAILABLE	7	10

(b)

Fig. 6. (a) Results from the SemTab '21 table-to-KG matching challenge for GitTables. The low precision and recall highlight the need for systems to better match table columns to KGs even if table values are not easily linked to KGs, as is the case for GitTables. (b) A table retrieved for search query "*status and sales amount per product*" indicating the relevance of GitTables for benchmarking data search systems in enterprise contexts.

5.3 Data management benchmarks

GitTables provides a rich source for database-like tables that can be used to compile subsets for evaluating data management tasks such as table understanding and data search.

We created a benchmark dataset for semantic column type detection. We manually curated 1,101 tables, each having at least 3 columns and 5 rows. The target columns are associated with 122 distinct types from DBpedia and 59 types from Schema.org, as obtained with the syntactic annotation method. We envision that this benchmark dataset will stimulate the development of systems for enhancing knowledge graphs from novel data sources [47]. A first step in this direction is the inclusion of this dataset in the SemTab challenge for table-to-KG matching [13]. The results on this dataset already reveal open challenges for this task, as many systems rely on matching column cell values to KG entities. This typically does not work for tables from GitTables, as reflected by the low scores across systems (4 systems for DBpedia, and 3 for Schema.org), shown in Figure 6a. The average precision on the Schema.org annotations is slightly higher due to pattern matching methods that detected few structural types well. Overall, our benchmark dataset poses new challenges for systems to match tables from sources beyond the Web to KGs.

Beyond table-to-KG matching, GitTables provides the basis for benchmarking data management tasks like data search. Evaluating systems on their search performance with GitTables is critical for understanding if they generalize to e.g. enterprise databases. To illustrate the relevance of GitTables for benchmarking this task, we implement a search procedure similar to Algorithm 1 but instead embed entire table schemas and compare them with embedded search queries in natural language. Figure 6b shows a typical database table with product order data which is retrieved for the query "*status and sales amount per product*". To develop this benchmark dataset further, one could collect a set of tables and queries and rank the most relevant tables for each query.

6 CONCLUSION

Relational databases are the bread and butter of enterprise applications. In this paper we introduce GitTables, a corpus of 1M relational tables complementing existing Web table corpora available to researchers. We extract tables from CSV files in open-source repositories on GitHub, curate the corpus for quality control, and enhance tables with semantic column type annotations from DBpedia and Schema.org. We analyse the corpus statistics, table contents, and annotation quality, which illustrate the complementary value of GitTables with respect to existing table corpora. We show that GitTables can be leveraged for 1) learned models for semantic column type detection with complementary coverage compared to existing corpora, 2) accurately suggesting schema completions in typical database contexts, and 3) developing challenging benchmarks for data management tasks. In future work, we are keen on exploring the broader utility of GitTables, which stretches beyond these use cases to a wide range of data management tasks.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A nucleus for a web of open data. *ISWC* (2007), 722–735.

[2] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 610–623.

[3] 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*. 18–26.

[4] Abeba Birhane and Vinay Uday Prabhu. 2021. Large image datasets: A pyrrhic win for computer vision?. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1536–1546.

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165* (2020).

[7] Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *Proceedings of the VLDB Endowment* 11, 12 (2018), 2140–2149.

[8] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[9] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the Power of Tables on the Web. *PVLDB* (2008), 538–549.

[10] Michael J Cafarella, Alon Y Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu. 2008. Uncovering the Relational Web.. In *WebDB*. 1–6.

[11] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Brussels, Belgium, 169–174. https://doi.org/10.18653/v1/D18-2029

[12] 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, 328–343.

[13] Vincenzo Cutrona, Jiaoyan Chen, Vasilis Efthymiou, Oktie Hassanzadeh, Ernesto Jiménez-Ruiz, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, Daniela Oliveira, and Catia Pesquita. 2021. Results of SemTab 2021. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021 (CEUR Workshop Proceedings, Vol. 3103)*. CEUR-WS.org, 1–12. http://ceur-ws.org/Vol-3103/paper0.pdf

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[15] 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.

[16] 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*. 4171–4186.

[17] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the dresden web table corpus: A classification approach. In *2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC)*. IEEE, 41–50.

[18] Daniele Faraglia and Other Contributors. 2014. Faker. https://github.com/joke2k/faker

[19] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. 2016. Schema. org: evolution of structured data on the web. *Commun. ACM* 59, 2 (2016), 44–51.

[20] Kevin Hu, Neil Gaikwad, Michiel Bakker, Madelon Hulsebos, 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 *CHI*. ACM.

[21] 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*. 1500–1508.

[22] Ernesto Jimenez-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.

[23] Laura Koesten, Pavlos Vougiouklis, Elena Simperl, and Paul Groth. 2020. Dataset Reuse: Toward Translating Principles to Practice. *Patterns* (2020), 100136.

[24] Sven Langenecker, Christoph Sturm, Christian Schalles, and Carsten Binnig. 2021. Towards Learned Metadata Extraction for Data Lakes. *BTW 2021* (2021).

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[26] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A Large Public Corpus of Web Tables Containing Time and Context Metadata. In *WWW Companion*. 75–76.

[27] Keqian Li, Yeye He, and Kris Ganjam. 2017. Discovering enterprise concepts using spreadsheet tables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1873–1882.

[28] Wes McKinney et al. 2011. pandas: a foundational Python library for data analysis and statistics. *Python for High Performance and Scientific Computing* 14, 9 (2011), 1–9.

[29] Johann Mitlöhner, Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. 2016. Characteristics of open data CSV files. In *2016 2nd International Conference on Open and Big Data (OBD)*. IEEE, 72–79.

[30] Jan Motl and Oliver Schulte. 2015. The CTU prague relational learning repository. *arXiv preprint arXiv:1511.03086* (2015).

[31] Hannes Mühleisen and Christian Bizer. 2012. Web Data Commons - extracting structured data from two large web corpora. In *LDOW*.

[32] Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. 2016. Automated quality assessment of metadata across open data portals. *Journal of Data and Information Quality (JDIQ)* 8, 1 (2016), 1–29.

[33] Jeffrey Perkel. 2016. Democratic databases: science on GitHub. *Nature News* 538, 7623 (2016), 127.

[34] Plotly. 2018. Plotly Community Feed. https://chart-studio.plotly.com/feed/

[35] Stephan Rabanser, Stephan Günnemann, and Zachary C Lipton. 2019. Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift. *33rd Conference on Neural Information Processing Systems* (2019).

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR* (2020).

[37] Dominique Ritze and Christian Bizer. 2017. Matching web tables to DBpedia – a feature utility study. *EDBT* 42, 41 (2017), 19.

[38] Dominique Ritze, Oliver Lehmberg, and Christian Bizer. 2021. T2Dv2 Gold Standard for Matching Web Tables to DBpedia. http://webdatacommons.org/webtables/goldstandardV2.html Accessed: 01-05-2021.

[39] Jeni Tenneson. 2016. CSV on the web: A primer. http://www.w3.org/TR/2016/NOTE-tabular-data-primer-20160225/

[40] Princeton University. 2010. About WordNet. https://wordnet.princeton.edu

[41] Gerrit JJ van den Burg, Alfredo Nazábal, and Charles Sutton. 2019. Wrangling messy CSV files by detecting row and type patterns. *Data Mining and Knowledge Discovery* 33, 6 (2019), 1799–1820.

[42] Fernanda B Viegas, Martin Wattenberg, Frank Van Ham, Jesse Kriss, and Matt McKeon. 2007. Manyeyes: a site for visualization at internet scale. *IEEE transactions on visualization and computer graphics* 13, 6 (2007), 1121–1128.

[43] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. 2020. REVISE: A tool for measuring and mitigating bias in visual datasets. In *European Conference on Computer Vision*. Springer, 733–751.

[44] Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. TCN: Table Convolutional Network for Web Table Interpretation. *arXiv preprint arXiv:2102.09460* (2021).

[45] Wei Wang, Meihui Zhang, Gang Chen, HV Jagadish, Beng Chin Ooi, and Kian-Lee Tan. 2016. Database meets deep learning: Challenges and opportunities. *ACM SIGMOD Record* 45, 2 (2016), 17–22.

[46] WebDataCommons. 2021. WDC Web Table Corpus 2012. http://webdatacommons.org/webtables/2012/relationalStatistics.html

[47] Gerhard Weikum. 2021. Knowledge graphs 2021: a data odyssey. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3233–3238.

[48] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *ACL*.

[49] Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 2 (2020), 1–35.

Received April 2022; revised July 2022; accepted August 2022.