Tough Tables: Carefully Evaluating Entity Linking for Tabular Data

Vincenzo Cutrona[1 [0000−0001−7830−7596]], Federico Bianchi[2 [0000−0003−0776−361X]], Ernesto Jiménez-Ruiz[3,4 [0000−0002−9083−4599]], and Matteo Palmonari[1 [0000−0002−1801−5118]]

[1] University of Milano - Bicocca, Milan, Italy
{vincenzo.cutrona, matteo.palmonari}@unimib.it
[2] Bocconi University, Milan, Italy
f.bianchi@unibocconi.it
[3] City, University of London, London, United Kingdom
ernesto.jimenez-ruiz@city.ac.uk
[4] University of Oslo, Oslo, Norway

Abstract. Table annotation is a key task to improve querying the Web and support the Knowledge Graph population from legacy sources (tables). Last year, the SemTab challenge was introduced to unify different efforts to evaluate table annotation algorithms by providing a common interface and several general-purpose datasets as a ground truth. The SemTab dataset is useful to have a general understanding of how these algorithms work, and the organizers of the challenge included some artificial noise to the data to make the annotation trickier. However, it is hard to analyze specific aspects in an automatic way. For example, the ambiguity of names at the entity-level can largely affect the quality of the annotation. In this paper, we propose a novel dataset to complement the datasets proposed by SemTab. The dataset consists of a set of high-quality manually-curated tables with non-obviously linkable cells, i.e., where values are ambiguous names, typos, and misspelled entity names not appearing in the current version of the SemTab dataset. These challenges are particularly relevant for the ingestion of structured legacy sources into existing knowledge graphs. Evaluations run on this dataset show that ambiguity is a key problem for entity linking algorithms and encourage a promising direction for future work in the field.

Keywords: Entity Linking · Instance-level matching · Cell Entity Annotation · Semantic Labeling · Table annotation

1 Introduction

Tables are one of the most used formats to organize data. Every day, both data practitioners and business people have to handle tables that have been extracted from databases of sales, pricing, and more. Using these tables to build a new knowledge graph (KG), populate an existing one [10], or enrich the data in the table with additional information available in existing KGs [3] requires the

source data to be manipulated, interpreted within a graph-based schema (e.g., an ontology), transformed and linked to a reference or to existing KGs. The latter step, in particular, consists in an entity linking task, that is, in connecting cells to reference identifiers (e.g., URIs) that are used to describe, in larger KGs, the entity referred to the cell.

The task of interpreting the table under a graph-based schema and link cells to entities described in a reference KG is referred in the literature to as Table Annotation, Table Interpretation, or Semantic Labeling. It requires the introduction of semantic algorithms, namely semantic table interpreters, that link cells to elements in a KG. Recently, the SemTab 2019 [7] challenge was introduced to unify the community efforts towards the development of performing annotations. The challenge consists of different rounds in which tables of various difficulties have to be annotated. However, it is often difficult to understand what are the shortcoming of each algorithm and how difficult the tables are. For example, are algorithms able to correctly annotate tables that contain homonymic names of people? In Figure 1, we show a case of ambiguity. In the Semantic Web community, this issue has already been highlighted, and it becomes necessary to have tables that resemble real use-cases [7].

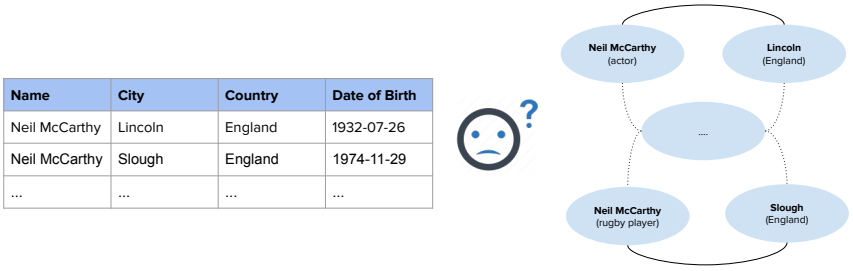

Fig. 1. Ambiguities make table annotation more difficult.

In this paper, we propose a manually curated dataset for table annotation useful for evaluating specific aspects of the annotation task, and, in particular, of the entity linking task in table interpretation. This dataset is complementary to the datasets already introduced in the SemTab challenge. We manually checked the tables with the following question in mind: "Would a human annotator be able to disambiguate this table?". Considering the intrinsic ambiguity of references appearing in tables, we want to ensure that the dataset includes those tables that can be effectively disambiguated, based on the information available in the table by a human annotator. In fact, we report some cases where the human annotators found very hard to match cells of some tables, due to the high ambiguity of their content; when the correct link was hard to be decided based on the table content (the context supporting the disambiguation), we opted for the conservative approach and decided to not annotate the cell (e.g.,

for the table with the list of bank failures since 2,000, it was not always clear who acquired the bank: whether it was a bank or its holding company).

A good table interpretation algorithm should be able to balance different aspects that need to be considered in the linking process: if the algorithm weights too much the evidence provided by string matching, it will fail to recognize nicknames and different names for things; if it expects clean text, it will fail to identify misspelled entities; if it relies on popularity for disambiguation entities, it will give more weight to popular cities than to the homonymic cities with the same name; if it allows too much fuzziness in the search of the candidate entities, i.e., a pool of entities that are selected as potential links usually based on string matching criteria, it will generate a considerable amount of possible candidates, making the search for the correct one more difficult and prone to errors. We believe that it is helpful to consider these aspects separately in order to evaluate the real power of different entity linking methods in handling the different challenges that data in the tables present. Creating a dataset that features all the aforementioned aspects requires to collect non-artificial tables; in fact, building a dataset via generators (e.g., tables created by querying a SPARQL endpoint) has the advantage of creating a multitude of different tables quickly, but for example it is not possible to create tables with new content (i.e., with facts missing in the reference KG).

Compared to previous benchmarks, this new dataset has the following distinguishing features, which make it a very valuable resource for the fundamental task of table interpretation:

1. **Real Tables** - useful for testing how the table interpretation algorithms deal with the knowledge gap due to novel facts. It can be often the case that some cells in a table refer to entities that are described in the reference KG, for which the algorithm is expected to link the correct entity, and some cells refer to entities not described in the reference KG, for which the algorithm should decide not to link any of the existing entities.
2. **Tables with Ambiguous Mentions** - useful to test the algorithms' capability to handle the ambiguity and link also to non-popular entities (tail entities).
3. **Tables with Misspelled Mentions** - useful for testing the weight of lexical features used by the algorithms. We used the misspelled words as a generator to add controlled noise to other tables.
4. **Tables from Various Sources** - useful to understand which are the most difficult tables to deal with for an algorithm.
5. **Manually Verified Tables** - useful to prevent false positives while evaluating the algorithms; the dataset is of high quality and all the annotations have been manually verified.

2 Background

In this section we define more precisely the terminology we use along the paper. We refer to table annotation as the task of linking cells of a table to elements of

a KG. We distinguish from instance-level matching and schema-level matching: the first aims at linking mentions in the table to entities of the KG (e.g., "10. Alex Del Piero" and "Juventus" can be mapped to dbr:Alessandro_Del_Piero and dbr:Juventus_F.C. in DBpedia), while the latter is dedicated to the columns and their headers, i.e., the table schema, linking the heading cells to elements of the ontology provided by the KG (e.g., the columns "Name" and "Team" could be mapped to types dbo:SoccerPlayer and dbo:Team, and they can be linked with the property "dbo:team"). While the instance-level matching can always rely on the cell content, which represents a mention of the entity to be linked, sometimes the schema-level matching must deal with more challenging tables where the first row is empty (no headers cells), or contains meaningless cells (e.g., codes from the legacy sources). Recently, the SemTab challenge introduced a new terminology, splitting the table annotation task into three sub-tasks:

- **CTA** (Column-Type Annotation), that is the schema-level matching focused only on linking columns to ontology types;
- **CEA** (Cell-Entity Annotation), that is the instance-level matching; this is what is also referred to as entity linking, as we do in this paper.
- **CPA** (Columns-Property Annotation), that is the schema-level matching focused only on linking columns of the table through ontology properties.

Even if the challenge evaluates these three tasks separately, they are usually solved together, possibly, iteratively. A fully-automated table annotation approach can start by solving the CEA task, e.g., by finding some entities linked with a certain confidence, then it can use these entities to find the right type for the column, solving CTA, and finally it may go back to refine CEA, e.g., by using the inferred type to filter irrelevant entities and support the disambiguation. Finally, the CPA task is typically solved by combining the information collected during the other two tasks and selecting the properties that fit the pairs of types/entities that have been found. Due to the importance of the CEA task, and to the fact that this task is useful for data enrichment even if a full table to graph transformation is not required, we decided to focus our dataset on the evalation of the entity linking (CEA). By using these annotations then we have also derived type-level annotations of the table headers, thus providing also ground truth for the CTA task (further details in the Appendix A.2). We left the ground truth for the CPA task as future work.

3 Limitations in Related Datasets

In the last decade different benchmark datasets have been proposed in the literature. The most important and used are T2Dv2[1] (also referred to as T2D in this paper), Limaye [8] and W2D [4]. They come with different capabilities. As an example, while T2D provides tables with properties for the CPA task, W2D does not cover this task very well. We can then observe that, even if the above

[1] T2Dv2: `http://webdatacommons.org/webtables/goldstandardV2.html`

datasets differ in size (T2D has 200 tables, while W2D counts more than 485k), all of them are focused on web tables, i.e., small tables scraped from the web; these datasets are particularly suitable to benchmark table interpretation algorithms whose objective is to support query answering over the large amount of tables published on the web [11]. However, tables considered for the population of KGs are usually quite different from web tables, e.g., they are much larger, and these kind of tables are not representated in the available datasets. We can shortly summarize the difference between these two kinds of tables as follows:

- Legacy tables usually have many rows, while tables in existing benchmark datasets are small (the average number of rows per table is 123 for T2D, 29 for Limaye and only 15 for W2D, according to [4]). Large tables may prevent algorithms from using heuristics that consider the full table (e.g., infer the column type by looking at the whole column).
- Legacy tables, especially CSVs, usually contain de-normalised tables with several columns; this aspect is not well represented in the considered datasets (each table contains on average ∼1.77 columns with entities).
- Because of the usual de-normalization, legacy tables contain many columns with entity matches, but tables in existing benchmarks are mostly focused on "entity tables", i.e., tables where each row represents only one entity; in such a table, one column refers to the entity (it is also called *subject* column), and all the other contain attributes of the main entity; this scenario does not fit the case of de-normalised tables. We also report that in some cases (e.g., T2D), if the table contains more than one entity column, they are disregarded and not annotated.
- Entities in web tables are usually mentioned using their canonical name (e.g., Barack Obama is mentioned as "Barack Obama" - it is very unusual to see "B. Obama"); in legacy sources, we find acronyms, abbreviations, misspelled words that considerably increase the ambiguity of the table. For example, the misspelling of drug names is a very important problem in the health domain [5, 6].

A recent study reported that many of the approaches tested on such datasets are focused on "obviously linkable" cells [14], showing that a tool like T2K [10] manages to match only 2.85% of a large corpus of Web tables to DBpedia. However, the performance of T2K evaluated on T2D relatively to the CEA task is very high (F1: 0.82, Precision: 0.90, Recall: 0.76). This suggests that many tables in T2D are easy to annotate. The authors themselves specify that the entity linking can be wrong sometimes, especially when the mention is ambiguous, spotlighting that there are a few ambiguous mentions in T2D. Recently, SemTab (Semantic Web Challenge on Tabular Data to Knowledge Graph Matching) was created to conduct a systematic evaluation of table annotation algorithms [7]. The organizers presented new datasets built by automatically generating tables from the results of SPARQL queries. The first rounds of SemTab 2019 also included tables from T2D and W2D datasets. The success of the challenge and the analysis of the results achieved by different competing approaches have revealed some challenges that should be addressed in future work:

- Wikipedia and DBpedia lookup mechanisms are effective for candidate selection tasks, but they might fail when the cell content is misspelled (e.g., Winsconsin vs Wisconsin) or is different from the most common one (e.g., football player nicknames: La Pulga vs Lionel Messi).
- Real world tables are noisy and in general not well formed. Algorithms should be evaluated also with respect to their ability to deal with such tables.
- Missing data can affect the results of the algorithms and thus this aspect should be correctly evaluated.
- Although the overall quality of the SemTab 2019 dataset is higher compared with the other datasets, a manual inspection of the tables in the SemTab brought to the surface some malformed and wrongly annotated tables, like empty rows mapped to an entity and long descriptions (with mentions of different entities) mapped to a single entity.[2]

Finally, existing datasets have contributed to the benchmarking of table interpretation algorithms, however, none of them provide fine grained information about the achievement of a certain score. Some tools have been developed to at least highlight which are the main error patterns, but those patterns must be manually inspected (e.g., [2]). Our dataset has been built to facilitate the understanding of the main limitations of a table interpretation system, since it provides a footprint about the uncovered aspects that led to certain performance score.

4 The 2T Dataset

In this paper we present Tough Tables (2T), a dataset designed to evaluate table annotation approaches on the CEA and CTA tasks. All the annotations are based on DBpedia 2016-10.[3] The structure of the dataset (depicted in Figure 2) allows the user to know Which aspects of the entity linking task are handled better/worse by different approaches. Indeed, the dataset comes with two main categories of tables:

- the *control* group (CTRL), which contains tables that are easy to solve; a table annotation algorithm should at least annotate these tables with relatively high performance.
- the *tough* group (TOUGH), which features only tables that are hard to annotate.

A complete algorithm should solve both the categories, because otherwise i) solving only the CTRL group means that the algorithm is able to only cope with obvious entities and ii) solving only the TOUGH tables highlights that the algorithm is too complex and cannot deal with the simpler cases.

[2] See Tables *53822652_0_5767892317858575530* and *12th_Goya_Awards#1* from Round 1 and Round 2, respectively. These errors come from the T2D and W2D datasets used in SemTab 2019.

[3] We checked our annotations against a private replica of the online DBpedia SPARQL endpoint in a local instance, loading the 2016-10 datasets listed at `https://wiki.dbpedia.org/public-sparql-endpoint`.

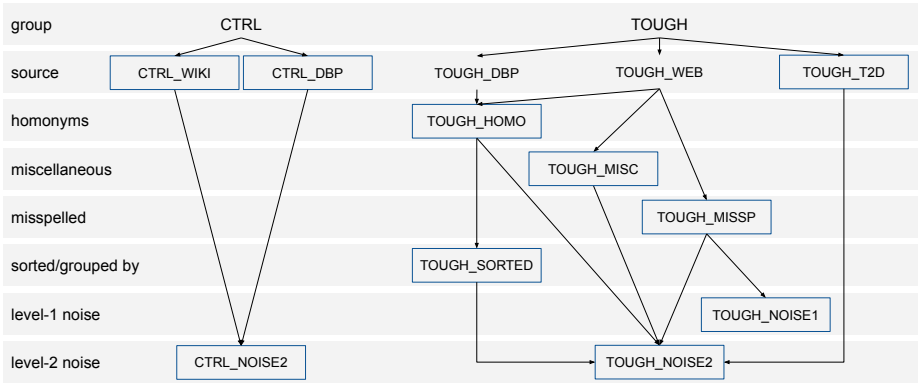

Fig. 2. 2T profile at a glance. Boxed categories are those considered during the evaluation phase. Each category is composed by some of/all the tables from the parent categories.

4.1 Dataset Profile

Figure 2 provides an overview of the different table flavours included within the 2T dataset. The *control* group (CTRL) contains tables generated by querying the DBpedia SPARQL endpoint[4] (CTRL_DBP), leaving the label as extracted from the KG, and tables collected from Wikipedia (CTRL_WIKI), manually revised and annotated (further details about this procedure are available in Appendix A.1). Like the DBP ones, the WIKI tables are pretty clear and easy to annotate, since most of the mentions refer to entities in DBpedia with just slightly different labels from the ones contained in this KG.

The *tough* group (TOUGH) contains mainly tables scraped from the web. This group contains also a small subset of T2D (TOUGH_T2D), which we re-annotated considering the entities appearing in all the columns, and not only entities in the "subject" column like T2D. In addition, we collected tables from the web that contain nicknames or homonyms (TOUGH_HOMO), and misspelled words (TOUGH_MISSP). We enriched the TOUGH_HOMO category by adding a few tables generated via ad-hoc SPARQL queries. Along with these tables, we included other web tables (TOUGH_MISC), like non-cleaned Wikipedia tables and tables available as Open Data (in a limited quantity, due to motivations stated in Appendix A.1).

We selected some specific tables from the TOUGH_HOMO category and sorted them in a specific order to make the task of detecting the type more difficult (TOUGH_SORTED). If we are creating a table describing athletes we would probably intuitively organize them by category, having the soccer players in the first part, then all the basket players, then all the golf players, etc. Sorting tables might pose a problem for those algorithms that infer the column type by

[4] We used the online version at `http://dbpedia.org/sparql`.

only looking at the first n rows (with n usually small), and then use the inferred type as a filter for the entity linking.[5]

Noisy tables. We used the already collected tables to generate other noisy tough tables. Starting from tables in TOUGH_MISSP, we generated a new category (TOUGH_NOISE1) by adding a *level 1* noise: for each table, 10 additional tables with noise have been generated, where each of them contains an incremental percentage of misspelled mentions (increasing by 10% at a time). This noise resembles real noise since we use lists of real-world misspelled words and use them to generate noise. From the tables in the CTRL and TOUGH categories (excluding the TOUGH_NOISE1 category), we also created two new categories (CTRL_NOISE2 and TOUGH_NOISE2) via a *level-2* noise, i.e., random noise that changes a bit the labels of randomly selected columns/rows (e.g., it randomly duplicates a symbol). Tables in this new category feature a noise that is random and artificial, thus it does not always resemble a real world scenario.

Novel facts. One of the main applications of the table interpretation is to extract new facts from tables, especially for KG population/completion tasks. In data integration pipelines, entity linking and new triples generation play an equally important role. Novel facts detection is not considered in the standard CEA evaluation,[6] but we outline that our dataset can be used to test algorithms in finding new facts. 2T tables contain 3,292 entity mentions across 42 tables without a corresponding entity in DBpedia 2016-10. In the CEA asset, a good table annotation algorithm is expected to decide that such cells should not be linked (similar to the *NIL* prediction in Named Entity Linking/Recognition). In more comprehensive assets, like KG construction/population, we expect the algorithm to generate a new triple (with rdf:type) using the discovered column type. Depending on the context, such particular cells might be used in the future to test novel knowledge discovery algorithms.

Overview. Benchmarking a table annotation system using 2T allows the developers to understand why their algorithms achieved a certain score as follows:

- if the algorithm performs well only on the control tables, then it relies too much on the performance of simple string matching strategies like label lookup (i.e., it looks only for exact matches, or considers only the canonical name of entities);
- if the performance is good also on the control tables with level-2 noise, then the algorithm adopts a kind of fuzziness in its lookup phase (e.g., edit distance), which is still not enough to solve the tough tables;
- if the algorithm performs well on some tough tables and bad on others, then the user can better understand the weaknesses of the algorithm by looking at the performance on different categories of tables:

[5] This strategy, that might look naive, is the same implemented in OpenRefine, where the first 10 rows are used to suggest the possible types of the current column.

[6] The SemTab 2019 challenge provided the *target* file with the full list of cells to annotate, disregarding novel facts.

- if the tables with columns sorted by type are annotated in the wrong way, then the entity linking algorithm is constrained by the type inferred looking at the first n rows, with n too small;
- if homonyms or nicknames have been wrongly matched, that means that the algorithm employs popularity mechanisms (e.g., page rank), or it is based on a lookup service that returns the most popular entities first (e.g., DBpedia Lookup).[7] Annotating nicknames requires the algorithms to cover aspects of semantics that go a bit beyond simple heuristics;[8]
- if the tables with level-1 noise are not properly annotated, then the algorithm cannot deal with real-world noise (that can be trickier than the artificial level-2 noise);
- if the annotations are wrong for the tables containing nicknames, it might be the case the algorithm only focuses on the canonical names of the entities.

Tables 1 and 2 show statistics for 2T and existing benchmark datasets. Compared to existing datasets, 2T has a higher average number of rows per table, pushing the size of individual tables towards the size of real legacy sources; the number of matches is slightly grater than the number available in SemTab 2019 Rounds 2 and 3 (ST19 - R2 and ST19 - R3 in Table 2), considering that 2T comes with a number of tables that is up to two orders of magnitude smaller. Since some tables in 2T are built starting from the same core table, we observe a small number of unique entities. Finally, 2T tables have a lower average number of columns per table, but the highest number of columns with at least a match: this aspect helps in having more columns to annotate in the CTA task, and it is also a starting point for future extensions of 2T, i.e., covering CPA task.

4.2 Evaluation

We ran experiments to evaluate the toughness of our dataset and its capability of spotting the weaknesses of an annotator. We setup an environment that resembles the CEA task of SemTab 2019, i.e., target cells are known and extra annotations are disregarded. We also used the available code to score the algorithms in the same way.[9] We introduce two simple baselines, DBLookup and WikipediaSearch, which run a query against the corresponding online lookup services using only the actual cell content.[10] Both the baselines use the first re-

[7] We point out that some homonyms are very easy to solve using DBpedia (e.g., US cities are easy to find, since just appending the state of a city to its canonical name points directly to the right city, e.g., the Cambridge city in Illinois is dbr:Cambridge,_Illinois in DBpedia).

[8] Note that it is possible to solve this problem using a mapping dictionary if available, but this is not a desired solution: this will not make the algorithm *smart*; the same is true for looking up on Google Search.

[9] `https://github.com/sem-tab-challenge/aicrowd-evaluator`

[10] We used the WikipediaSearch online service available at `https://en.wikipedia.org/w/api.php`, while we recreated the DBLookup online instance on a dedicated virtual machine.

Table 1. Detailed statistics for 2T. Values are given as *avg* ± *st.dev. (total, min, max)*

Category	Cols	Rows	Matches	Entities	Cols with Matches	Tables
ALL	4.47±1.90 (804, 1, 8)	1080.38±2805.25 (194468, 5, 15477)	3687.94±10142.48 (663830, 6, 61908)	438.77±1241.97 (16464, 6, 7032)	2.99±1.17 (538, 1, 6)	180
CTRL WIKI	5.73±1.28 (86, 4, 7)	66.00±81.39 (990, 10, 263)	241.93±333.21 (3629, 20, 1041)	157.93±224.06 (1940, 15, 771)	3.47±1.41 (52, 2, 6)	15
CTRL DBP	4.40±0.91 (66, 3, 6)	709.60±717.65 (10644, 120, 2408)	2510.67±2573.68 (37660, 360, 7820)	343.40±217.08 (4976, 68, 618)	3.53±0.64 (53, 3, 5)	15
CTRL NOISE2	5.07±1.28 (152, 3, 7)	387.77±599.15 (11633, 10, 2408)	1375.93±2140.04 (41278, 20, 7820)	250.53±236.31 (6745, 15, 770)	3.50±1.07 (105, 2, 6)	30
TOUGH T2D	5.82±1.83 (64, 3, 8)	78.09±77.26 (859, 6, 232)	165.36±150.28 (1819, 6, 464)	94.45±80.05 (991, 6, 248)	2.09±0.94 (23, 1, 4)	11
TOUGH HOMO	3.36±1.12 (37, 2, 5)	1648.82±3272.12 (18137, 13, 8302)	6422.55±13168.36 (70648, 25, 33208)	1469.27±2719.02 (8331, 24, 7032)	3.00±0.77 (33, 2, 4)	11
TOUGH MISC	6.50±1.31 (78, 4, 8)	122.25±162.86 (1467, 11, 561)	366.33±416.95 (4396, 22, 1215)	222.92±261.32 (2374, 16, 770)	3.67±1.44 (44, 2, 6)	12
TOUGH MISSP	3.50±1.29 (14, 2, 5)	4175.50±7549.48 (16702, 52, 15477)	16386.50±30381.91 (65546, 178, 61908)	204.00±350.86 (775, 13, 730)	3.25±0.96 (13, 2, 4)	4
TOUGH SORTED	3.50±2.12 (7, 2, 5)	4215.00±5779.89 (8430, 128, 8302)	16732.00±23300.58 (33464, 256, 33208)	3602.00±4850.75 (7201, 172, 7032)	3.00±1.41 (6, 2, 4)	2
TOUGH NOISE1	2.50±1.13 (100, 1, 4	2000.30±3701.62 (80012, 5, 14008)	5738.15±11197.12 (229526, 15, 42024)	204.00±307.73 (775, 13, 730)	2.25±0.84 (90, 1, 3)	40
TOUGH NOISE2	5.00±1.97 (200, 2, 8)	1139.88±3183.53 (45595, 6, 15477)	4396.75±12774.85 (175870, 6, 61908)	697.33±1823.97 (11820, 6, 7032)	2.98±1.21 (119, 1, 6)	40

sult returned by the online service as the candidate annotation. Alongside the baselines, we looked for a real algorithm to test among the ones that participated in the SemTab 2019. We contacted the authors of MTab [9], CVS2KG [13], and Tabularisi [12], the tools that performed the best, but their tools were not publicly available. Only the authors of MantisTable [1], winner of the outstanding improvement award at SemTab 2019 (CEA task), provided us with a prototype of their tool.[11] Since the performance obtained by all the tools were similar to each other, we think that MantisTable is a good representative for the evaluation. We run MantisTable and the two baselines on 2T, obtaining the results depicted in Figure 3. We underline here that our dataset adopts the same standard format defined in SemTab2019, making it compatible with all the systems that participated in the challenge.[12]

Results. In some cases, MantisTable could not annotate all the table cells (e.g., it failed to annotate tables with thousands of rows); this might be due to some general limits of the MantisTable prototype for which the solution is out of the

[11] A fork of the original code repository is available at `https://bitbucket.org/vcutrona/mantistable-tool.py`.

[12] The standard format introduced in SemTab2019 is directly derived from the T2Dv2 one, thus the number of algorithms that can be tested is potentially greater.

Table 2. Comparison with existing benchmark datasets.

Dataset	Cols (avg)	Rows (avg)	Matches	Entities	Cols with matches (avg)	Tables
T2Dv2	1,157 (4.95)	27,996 (123)	26,124	-	-	234
Limaye	-	-	142,737	-	-	6,522
W2D	- (5.58)	7,437,606 (15)	4,453,329	-	-	485,096
ST19 - R1	323 (5.05)	9,153 (143.02)	8,418	6,225	64 (1.00)	64
ST19 - R2	66,734 (5.60)	311,315 (26.12)	463,796	249,216	15,335 (1.29)	11,920
ST19 - R3	9,736 (4.51)	154,914 (71.69)	406,827	174,459	5,762 (2.67)	2,161
ST19 - R4	3,564 (4.36)	52,066 (63.73)	107,352	54,372	1,732 (2.12)	817
2T (ours)	804 (4.47)	194,468 (1080.38)	663,830	16,464	538 (2.99)	180

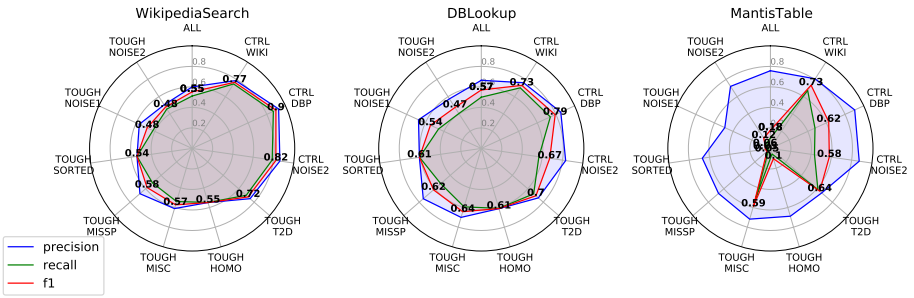

Fig. 3. Precision, Recall and F1 measures for the considered algorithms.

scope of the paper. Indeed, the algorithm is based on several heuristics, and processing big tables might lead to processing errors. For example, processing 180 tables took more than 24 hours. However, this does not compromise our evaluation, proving again that MantisTable can effectively annotate tables as shown in the SemTab 2019 challenge. The reader should take into account that the reported results may not reflect the full performance of MantisTable as they represent the output of a dry-run test without the involvement of the developers. Nevertheless, the results suggest that tough tables are effectively difficult to annotate for state-of-the-art algorithms. For the baselines, their precision and recall are quite similar since the online lookup services almost always return a result, and we set it as the annotation without further processing; focusing on the F1 measure, we observe that WikipediaSearch reaches 0.83 F1 on the CTRL group, which is high compared with state-of-the-art models, considering that the process only relies on the lookup service. The performance of DBLookup is good as well, but it decreases due to the CTRL_NOISE2 subcategory, reaching an average F1 of 0.73 on the CTRL group. Both algorithms are not able to annotate

the TOUGH tables, with DBLookup doing a bit better; this might be due to the fact that some of the tough tables have been built with SPARQL queries, giving some advantages to the DBLookup service. In general, the performance is low as expected (0.63 overall F1 for both the baselines), given that these algorithms do not use any kind of sophisticated semantic techniques. Looking at MantisTable results, we see that the tool is focusing on those cells that can be more easily linked to the KG. The semantic techniques employed in the algorithm pushes the precision on the control group, but due to the low recall, the algorithm performs worse (0.32 overall F1) than the previous baselines. Since the precision on CTRL_DBP tables is higher than CTRL_WIKI ones, we can assume that the lookup phase of the algorithm heavily depends on DBpedia (as a lookup service or SPARQL endpoint). The same is confirmed by the low recall on the TOUGH_NOISE1 tables, which are the ones with real-world misspelled mentions. For the subset of T2D tables that we chose and re-annotated, we spot a F1 score lower than the 0.98 obtained by MantisTable during the Round 1 of the SemTab challenge;[13] this confirms that the T2D tables are focused on obvious entities, disregarding the more difficult ones. The F1 score drastically decreases on the misspelled tables, highlighting that this aspect is still not fully covered in sophisticated state-of-the-art approaches like MantisTable.

4.3 Availability and Long-Term Plan

Resources should be easy accessible to allow data applicability and validation. We follow FAIR (Findable, Accessible, Interoperable and Reusable) guidelines to release our contributions.[14]

We release our dataset in Zenodo (DOI: `https://doi.org/10.5281/zenodo.3840646`), in such a way that researchers in the community can benefit from this. Our dataset is released under the Attribution 4.0 International (CC BY 4.0) License.[15] Together with the dataset we release the code that was used to collect the data and create it.[16] This should favor replicability and subsequent extensions.

5 Conclusions and Future Work

In this paper, we presented a novel dataset for benchmarking table annotation approaches on the entity linking and column type annotation tasks, equivalent to the CEA and CTA tasks as defined in current benchmarks [7]. The dataset comes with a mix of real and constructed tables, which resemble many real-world scenarios. We tested our dataset using a state-of-the-art approach, MantisTable, and two baselines. These baselines are represented by online lookup services, usually adopted as a building block of many table annotation approaches. We

[13] `https://www.cs.ox.ac.uk/isg/challenges/sem-tab/2019/results.html`
[14] `https://www.nature.com/articles/sdata201618`
[15] `https://creativecommons.org/licenses/by/4.0/`
[16] `https://github.com/vcutrona/tough-tables`

demonstrate that our tables are tough, and solving them requires the algorithms to implement sophisticated mechanisms that take into consideration many semantics aspects. In the near future, we intend to extend this dataset to cover also the CPA task; indeed, we observed that the 2T's profile fits well the benchmarking of this task because it contains many columns with matches, making the CPA more challenging (having averagely three possible columns to annotate instead of 2 increases by a degree of freedom the number of possible property annotations). The authors are also analysing the inclusion of this dataset in one of the rounds of SemTab 2020.[17]

Acknowledgments. We would like to thank the authors of MantisTable for sharing the prototype source code. This work was partially supported by the Google Cloud Platform Education Grant. EJR was supported by the SIRIUS Centre for Scalable Data Access (Research Council of Norway). FB is member of the Bocconi Institute for Data Science and Analytics (BIDSA) and the Data and Marketing Insights (DMI) unit.

References

1. Cremaschi, M., De Paoli, F., Rula, A., Spahiu, B.: A fully automated approach to a complete semantic table interpretation. Future Generation Computer Systems **112**, 478 – 500 (2020). https://doi.org/https://doi.org/10.1016/j.future.2020.05.019
2. Cremaschi, M., Siano, A., Avogadro, R., Jimenez-Ruiz, E., Maurino, A.: STILTool: a Semantic Table Interpretation evaLuation Tool. In: ESWC P&D (2020)
3. Cutrona, V., Paoli, F.D., Kosmerlj, A., Nikolov, N., Palmonari, M., Perales, F., Roman, D.: Semantically-enabled optimization of digital marketing campaigns. In: ISWC. pp. 345–362. Springer (2019). https://doi.org/10.1007/978-3-030-30796-7_22
4. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings. In: ISWC. pp. 260–277 (2017). https://doi.org/10.1007/978-3-319-68288-4_16
5. Hussain, F., Qamar, U.: Identification and Correction of Misspelled Drugs Names in Electronic Medical Records (EMR). In: ICEIS 2016 - Proceedings of the 18th International Conference on Enterprise Information Systems. vol. 2, pp. 333–338 (2016). https://doi.org/10.5220/0005911503330338
6. Jiang, K., Chen, T., Huang, L., Calix, R.A., Bernard, G.R.: A Data-Driven Method of Discovering Misspellings of Medication Names on Twitter. In: Building Continents of Knowledge in Oceans of Data: The Future of Co-Created eHealth - Proceedings of MIE 2018, Medical Informatics Europe. Studies in Health Technology and Informatics, vol. 247, pp. 136–140 (2018). https://doi.org/10.3233/978-1-61499-852-5-136

[17] The SemTab 2020 challenge is still in progress and it is organized to provide tables without known ground truth. For this reason, we will publish the full 2T dataset, including the ground truth files, at the end of SemTab 2020.

7. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In: ESWC. pp. 514–530. Springer (2020). https://doi.org/10.1007/978-3-030-49461-2_30
8. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. VLDB **3**(1), 1338–1347 (2010). https://doi.org/10.14778/1920841.1921005
9. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: MTab: Matching Tabular Data to Knowledge Graph using Probability Models. In: SemTab@ISWC. CEUR Workshop Proceedings, vol. 2553, pp. 7–14 (2019)
10. Ritze, D., Lehmberg, O., Oulabi, Y., Bizer, C.: Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016. pp. 251–261. ACM (2016). https://doi.org/10.1145/2872427.2883017
11. Sun, H., Ma, H., He, X., Yih, W.t., Su, Y., Yan, X.: Table Cell Search for Question Answering. In: WWW. p. 771–782. International World Wide Web Conferences Steering Committee (2016). https://doi.org/10.1145/2872427.2883080
12. Thawani, A., Hu, M., Hu, E., Zafar, H., Divvala, N.T., Singh, A., Qasemi, E., Szekely, P.A., Pujara, J.: Entity Linking to Knowledge Graphs to Infer Column Types and Properties. In: SemTab@ISWC. CEUR Workshop Proceedings, vol. 2553, pp. 25–32 (2019)
13. Vandewiele, G., Steenwinckel, B., Turck, F.D., Ongenae, F.: CVS2KG: Transforming Tabular Data into Semantic Knowledge. In: SemTab@ISWC. CEUR Workshop Proceedings, vol. 2553, pp. 33–40 (2019)
14. Zhang, S., Meij, E., Balog, K., Reinanda, R.: Novel Entity Discovery from Web Tables. In: Proceedings of The Web Conference 2020. p. 1298–1308. WWW '20 (2020). https://doi.org/10.1145/3366423.3380205

A 2T Ground Truth Generation Details

A.1 CEA Table Generation and Preprocessing

2T has been built using real tables. Here we clarify that as a "real table" we intend a table, also artificially built, which resembles a real table. Examples are "list of companies with their market segment", or "list of Italian merged political parties", which look like queries that a manager or a journalist could make against a database. The main reasons behind this choice are: *(i)* it is difficult to get access to real databases; *(ii)* open data make available a lot of tables, but mostly always tables are in an aggregated form that makes it difficult to annotate them with entities from a general KG like DBpedia. When the data are fine-grained enough, almost all the entities mentioned are not available in the reference KG. For example, in the list of bank failures got from the U.S. Open Data Portal, only 27 over 561 failed banks are represented in DBpedia.

In this section we describe the processes we adopted to collect real tables, or build tables that resembles real ones.

DBpedia Tables We used the DBpedia SPARQL endpoint as a table generator (SPARQL results are tables). We run queries to generate tables that include:

- entity columns: columns with DBpedia URIs that represent entities.
- "label columns": columns with possible mentions for the corresponding entities in the entity column. Given an entity column, the corresponding label column has been created by randomly choosing between `rdfs:label`, `foaf:name`, or `dbo:alias` properties.
- literal columns: other columns, with additional information.

Wikipedia Tables We browsed Wikipedia looking for pages containing tables of interest (e.g., list of presidents, list of companies, list of singers, etc.). We generated different versions of the collected Wikipedia tables, applying different cleaning steps. The following steps have been applied to Wikipedia tables in the *TOUGH_MISC* category:

- Merged cells have been split in multiple cells with the same value.
- Multi-value cells (slash-separated values, e.g., Pop / Rock, or multi-line values, e.g., Barbados
 United States, or in-line lists, e.g., <ul>, <li>) have been exploded into several lines. If two or more multi-value cells are on the same line, we exploded all the cells (cartesian product of all the values). If a cell contains the same information in more languages (e.g., anthem song titles), we exploded the cell in two or more columns (the creation of new lines would basically represent duplicates).

Wikipedia tables in the *CTRL_WIKI* group underwent the next additional cleaning steps:

- "Note", "Description", and similar long-text columns have been removed.
- Cells with "None", "null", "N/A", "Unaffiliated", and similar values have been emptied.
- Columns with only images (e.g., List of US presidents) have been removed.
- All HTML tags have been deleted from cells (e.g., country flag icons);
- Notes, footnotes, and any other additional within-cell information (e.g., birthYear and deathYear for U.S. presidents) have been removed.

Most of all the tables values are already hyperlinked to their Wikipedia page. We used the hyperlinks as the correct annotations (we trust Wikipedia as a correct source of information), following these criteria:

- If a cell content has several links, we took the most relevant annotation, given the column context (e.g., in table `https://en.wikipedia.org/wiki/List_of_presidents_of_the_United_States#Presidents` the "U.S. senator from Tennessee" cell in the "Prior office" column contains two annotations: `https://en.wikipedia.org/wiki/U.S._senator` and `https://en.wikipedia.org/wiki/Tennessee`; in this case we took only the `https://en.wikipedia.org/wiki/U.S._senator` annotation, as the column is about "Prior offices", not about places).
- Sometimes it happens that if the same value appears several times in the same column (e.g., music genres), only one instance has the hyperlink to the Wikipedia page. In these cases we copied the same hyperlink to all the instances.

– When the hyperlink is missing (e.g., Hard Rock labels in the table `https://en.wikipedia.org/wiki/List_of_best-selling_music_artists#250_million_or_more_records`), we manually added the right links by visiting the main entity page (e.g., `https://en.wikipedia.org/wiki/Led_Zeppelin`) and looking for the missing piece of information (e.g., under the "Genre" section on the Led Zeppelin page we can find Hard Rock linked to `https://en.wikipedia.org/wiki/Hard_rock`). In case when the information is missing in the main page (e.g., in the same table, Michael Jackson genres include "Dance", while on his Wiki page the genre is Dance-pop), we manually annotated the value with the most related entity in Wikipedia (in this case, the music genre Dance `https://en.wikipedia.org/wiki/Dance_music`).

Finally, we converted the Wikipedia links to their DBpedia correspondent links, by replacing `https://en.wikipedia.org/wiki/` with `http://dbpedia.org/resource` in the decoded URL, e.g., `https://en.wikipedia.org/wiki/McDonald\%27s` → `dbr:McDonald's`, if available, otherwise we manually looked for the right dbpedia link (e.g., `https://en.wikipedia.org/wiki/178889_United_States_presidential_election` → `dbr:United_States_presidential_election,_178889`). If this attempt also failed, we left the cell blank (no annotations available in DBpedia).

A.2 CTA Ground Truth Construction

Automatic CTA Annotations From CEA The 2T dataset focus is mainly on the entities because, in our opinion, the CEA task is the core task: with good performance in CEA, it is possible to approximate the CTA task easily. We exploited this observation to automatically construct the CTA annotations starting from the CEA ones, which we trust. For each annotated column, we collected all the annotated entities from the CEA dataset and retrieved the most specific type for all the entities from the DBpedia 2016-10 dump.[18] We then annotate the column with the most specific supertype, i.e., the lowest common ancestor of all the types in the DBpedia 2016-10 ontology.[19]

[18] The *instance types* file at `http://downloads.dbpedia.org/2016-10/core-i18n/en/instance_types_en.ttl.bz2` contains entities most specific types.

[19] `http://downloads.dbpedia.org/2016-10/dbpedia_2016-10.nt`