SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems

Ernesto Jiménez-Ruiz[1,2(✉)], Oktie Hassanzadeh[3], Vasilis Efthymiou[5], Jiaoyan Chen[4], and Kavitha Srinivas[3]

[1] City, University of London, London, UK
ernesto.jimenez-ruiz@city.ac.uk
[2] SIRIUS, University of Oslo, Oslo, Norway
ernestoj@uio.no
[3] IBM Research, Yorktown Heights, NY, USA
hassanzadeh@us.ibm.com, kavitha.srinivas@ibm.com
[4] University of Oxford, Oxford, UK
jiaoyan.chen@cs.ox.ac.uk
[5] IBM Research, San Jose, CA, USA
vasilis.efthymiou@ibm.com

Abstract. Tabular data to Knowledge Graph matching is the process of assigning semantic tags from knowledge graphs (e.g., Wikidata or DBpedia) to the elements of a table. This task is a challenging problem for various reasons, including the lack of metadata (e.g., table and column names), the noisiness, heterogeneity, incompleteness and ambiguity in the data. The results of this task provide significant insights about potentially highly valuable tabular data, as recent works have shown, enabling a new family of data analytics and data science applications. Despite significant amount of work on various flavors of this problem, there is a lack of a common framework to conduct a systematic evaluation of state-of-the-art systems. The creation of the *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* (SemTab) aims at filling this gap. In this paper, we report about the datasets, infrastructure and lessons learned from the first edition of the SemTab challenge.

Keywords: Tabular data · Knowledge graphs · Matching

1 Introduction

Tabular data in the form of CSV files is the common input format in a data analytics pipeline. However, a lack of understanding of the semantic structure and meaning of the content may hinder the data analytics process. Thus, gaining this semantic understanding will be very valuable for data integration, data cleaning, data mining, machine learning and knowledge discovery tasks. For example, understanding what the data is can help assess what sorts of transformation are appropriate on the data. Tables on the Web may also be the source of highly valuable data. The addition of semantic information to Web tables may

A. Harth et al. (Eds.): ESWC 2020, LNCS 12123, pp. 514–530, 2020.
https://doi.org/10.1007/978-3-030-49461-2_30

enhance a wide range of applications, such as web search, question answering, and knowledge base construction.

Tabular data to Knowledge Graph (KG) matching is the process of assigning semantic tags from KGs (e.g., Wikidata or DBpedia) to the elements of the table. This task however is often difficult in practice due to metadata (e.g., table and column names) being missing, noisy, incomplete or ambiguous. There exist several systems that address the tabular data to KG matching problem (*e.g.*, [5,8,26]) and use state-of-the-art datasets with ground truths (*e.g.*, [8,19,20]) or custom datasets. However, there does not exist a common framework to conduct a systematic evaluation of these systems, which leads to experimental results that are not easy to compare as they use different notions for true/false positives and performance measures. Furthermore, available datasets are either small in size (*e.g.*, [19,20]) or low in quality and messy (*e.g.*, [8]). The creation of the *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* (SemTab) [12] aims at filling this gap.

The main contributions of this paper are summarized as follows:

(i) We introduce an automated method for generating benchmark datasets for tabular data to KG matching.
(ii) We release 4 generated benchmark datasets (see Zenodo repository [13]), and the code for evaluating the systems results (see GitHub repository [3]).
(iii) We report and analyze the results of the systems that participated in the first edition of the SemTab challenge, using our 4 benchmark datasets.

The rest of the paper is organized as follows. Section 2 introduces the matching problems and its challenges. In Sect. 3, we discuss related initiatives. The automatic dataset generator is described in Sect. 4. Section 5 presents the SemTab evaluation. Finally, Sect. 6 provides the lessons learned and experiences from the SemTab challenge and points to future lines.

2 Background

In this section, we provide some basic definitions about KGs and tabular data. We also introduce the selected matching tasks and their associated challenges.

Knowledge Graph (KG). We consider RDF-based KGs which are represented as a set of RDF triples $\langle s, p, o \rangle$, where s represents a subject (a class or an instance), p represents a predicate (a property) and o represents an object (a class, an instance or a data value, *e.g.*, text, date and number). RDF entities (*i.e.*, classes, properties and instances) are represented by Uniform Resource Identifiers (URIs). A KG consists of a terminological component (TBox) and an assertion component (ABox). The TBox is often composed of RDF Schema constructs like class subsumption (*e.g.*, `dbo:Scientist rdfs:subClassOf dbo:Person`) and property domains (*e.g.*, `dbo:doctoralAdvisor rdfs:domain dbo:Scientist`). The ABox contains relationships among entities and semantic type definitions (*e.g.*, `dbr:Albert_Einstein rdf:type dbo:Scientist`). An OWL 2 ontology associated to the KG may provide more expressive constructors without a direct

Table 1. Excerpts of *(a)* a Web table about countries and capitals, *(b)* a real CSV file about broadband data, and *(c)* a custom table with start-ups from Oxford and their foundation year.

(a) Web table

China	Beijing
Indonesia	Jakarta
Congo	Kinshasa
Brazil	
Congo	Brazzaville

(b) CSV file

Virgin	60	London
BT	60	East
BT	40	Scotland
Virgen	40	Wales
Orange	30	West Midlands

(c) Custom table

OST	2017
DeepReason.ai	2018
Oxstem	2011
Oxbotica	2014
DeepMind	2010

translation into triples, which will contribute to the inference of new triples via logical reasoning. A KG can typically be accessed via a SPARQL endpoint[1] and via fuzzy matching based on an index of the lexical information associated to the KG entities. The latter is often referred to as KG lookup (*e.g.*, Spotlight for DBpedia [21] and OpenTapioca for Wikidata [7]).

Tabular Data. Tabular data can be seen as a set of columns $C = \{c_1, \ldots, c_m\}$, a set of rows $R = \{r_1, \ldots, r_n\}$, or a matrix of cells $T = \{t_{1,1}, \ldots, t_{n,m}\}$, where a column $c_k = \{t_{1,k}, \ldots, t_{n,k}\}$ and a row $r_k = \{t_{k,1}, \ldots, t_{k,m}\}$ are tuples of cells. We assume that all columns and rows have the same size, with possibly cells with empty values. In arbitrary tabular data, unlike in relational tables, column names and row identifiers (*i.e.*, primary keys) may be missing. In Web tables and relational tables, rows typically characterize an entity, while in arbitrary tabular data (*e.g.*, typical CSV files in data science) there may not be a leading entity in each row (see for example Table 1b).

Matching Tasks. We have selected the following tasks for the semantic annotation of tabular data: *(i)* Column-Type Annotation (CTA), *(ii)* Cell-Entity Annotation (CEA), and *(iii)* Columns-Property Annotation (CPA). These matching tasks can be seen as subtasks that can serve the larger purpose of matching an entire table to a class, or matching a row of a table to an entity. The CTA task expects the prediction of the semantic types (*i.e.*, KG classes) for every given table column c_k in a table T, *i.e.*, $CTA(T, c_k, KG) = \{st_1, \ldots, st_a\}$.[2] The CEA task requires the prediction of the entity or entities (*i.e.*, instances) that a cell $t_{i,j} \in T$ represents, *i.e.*, $CEA(T, t_{i,j}, KG) = \{e_1, \ldots, e_b\}$. Finally, the CPA task expects as output a set of KG properties that represent the relationship between the elements of the input columns c_k and c_l, *i.e.*, $CPA(T, c_k, c_l, KG) = \{p_1, \ldots, p_c\}$. Note that CTA (resp. CEA) task focuses on categorical columns (resp. cells) that can be represented with a KG class (resp. KG entity). Some numerical values may also represent entities if they play a *foreign key* role, but this would involve a different data wrangling task not considered in this work.

[1] For example, DBpedia Endpoint: http://dbpedia.org/sparql.

[2] Note that one could annotate with more than one KG and merge the results.

Challenges. The above matching tasks are challenging for various reasons including but not limited to: *(i)* Lack of metadata or uninformative table and column names, a typical scenario in Web tables and real-world tabular data. *(ii)* Noisiness in the data (*e.g.*, "Virgen" in Table 1b). *(iii)* Knowledge gap, cells without a correspondence to the KG (*e.g.*, Oxford start-ups in Table 1c). *(iv)* Ambiguous cells pointing to more than one possible entity (*e.g.*, "Congo" in Table 1a or "Virgin" and "Orange" in Table 1b). *(v)* Missing data (*i.e.*, cells without a value) increasing the effect of the knowledge gap (*e.g.*, capital of "Brazil" in Table 1a). *(vi)* Short labels or acronyms, which typically bring more ambiguity to KG matching (*e.g.*, "BT" in Table 1b).

3 Related Work

Several benchmarks have been proposed for semantic table annotation.

T2Dv2 [19] includes common tables drawn from the Web.[3] It contains 779 tables, with around 400 entity columns covering contents about place, work, organization, person, species, etc., around 26,000 DBpedia entity matches, and around 420 DBpedia property matches.

Limaye et al. [20] proposed a benchmark containing tables from Wikipedia pages.[4] It has 428 entity columns, each of which has 23 cells in average, and around 5,600 DBpedia entity matches.

Efthymiou et al. [8] created a benchmark containing 485,000 Wikipedia page tables. It has around 485,000 tables, with around 4,500,000 DBpedia entity matches. 620 of its entity columns are annotated with DBpedia classes by [4].

IMDB and Musicbrainz are other popular benchmarks. IMDB contains over 7,000 tables from IMDB movie web pages, and Musicbrainz contains some 1,400 tables from MusicBrainz web pages [29]. The entity mention cells are annotated with Freebase topics.

Viznet [15] contains 31 million datasets mined from open data repositories and visualization data repositories. Although Viznet was initially derived for use in visualizations, it has been used in the context of column-to-type matching (CTA task) of tables in a system called SHERLOCK [16]. SHERLOCK provides a total of 11,700 crowdsourced annotations from 390 human participants. However the annotations are not publicly available yet.

NumDB [18] is a dataset of 389 tables generated from DBpedia where the primary emphasis is on creating tables for identifying numerical columns. It allows the varying of the size of the table, as well as injection of different degrees of noise in the data, particularly in the textual data that can be used to match 'key' columns to test the robustness of any numerical matching approach.

Although these benchmarks are widely used in recent studies, they still suffer from a few shortcomings: *(i)* some benchmarks like Limaye and T2Dv2 are quite

[3] http://webdatacommons.org/webtables/goldstandardV2.htm.

[4] There have been different versions of this dataset. The one by [8] is described here.

Fig. 1. Steps for automatic dataset generation.

small, with only limited contents; *(ii)* those large benchmarks like Efthymiou are often in short of high quality ground truths, especially when all the three tasks need to be evaluated at the same time; *(iii)* large benchmarks often have a large number of rows but simple relations and contents (classes); *(iv)* most benchmarks have ground truth annotations from only one KG.[5] Meanwhile, using a fixed benchmark limits the evaluation of some special cases, such as the big knowledge gap when a large part of cells have no entity correspondences, while a system for generating benchmarks with an ad-hoc configuration enables researchers to evaluate the performance in face of these special cases. Our efforts target this lacuna in benchmarks.

Benchmarks have been also developed for the related task of ontology matching, which is a well studied problem [10,11]. Our benchmarking effort was inspired by the yearly Ontology Alignment Evaluation Initiative (OAEI).[6] The main difference between our benchmarks and the OAEI benchmarks is the level of heterogeneity involved in the two data sources to be matched. Instead of two semantically rich ontologies, as those in the OAEI benchmarks, we consider one rich ontology corresponding to the KG, and one typically shallow table in terms of semantics. This heterogeneity creates an additional challenge, which ontology matching tools were not originally designed to cope with, but we believe that those tools can also benefit from our benchmarks. Therefore, we also provide our benchmark data in RDF format and experiment with publicly available ontology matching tools (e.g., LogMap [17]), to better evaluate their potential strengths and weaknesses from a different perspective than OAEI (cf. Sect. 4.4).

4 Benchmark Data Generation

To overcome the limitations of the existing benchmark datasets, and to create new benchmark datasets for each round of the challenge without extensive human annotation, we designed an automated data generator that creates tabular data given a SPARQL endpoint. The idea is to create tabular data similar to tables found on the Web, but ensure a reasonable diversity in terms of size and coverage of classes and properties from various domains. In what follows, we describe each of the steps in the data generation pipeline summarized in Fig. 1.

[5] To ease participation SemTab 2019 only used DBpedia as the target KG; however, as described in Sect. 4, the data generator can be fed with other KGs.

[6] http://oaei.ontologymatching.org.

4.1 Profiling

Although we used the English DBpedia as our source for this edition of the challenge, given that most state-of-the-art systems and the most widely used benchmarks use DBpedia mappings, our goal was to design a generic method of creating benchmark data that can go beyond DBpedia annotations. This way, DBpedia can be replaced with e.g. Wikidata, or a domain-specific KG. We can also switch to other languages or create a multilingual collection. Given this goal, the first step in data generation is a profiling step in which the list of classes, properties, and some basic statistics are extracted. The output of the profiling step is: *1)* a list of classes along with the number of instances per class; *2)* a list of properties for each class along with: *(i)* the number of instances that have a value for the property; *(ii)* the datatype for datatype properties and the range class for object properties. This information will be used in the next step to construct SPARQL queries.

Although our current profiling is simple, performing the necessary SPARQL queries over existing RDF stores could still be slow, and so a raw processing of RDF dumps may be required. Another option is to use a profiling tool such as Loupe [22]. For table generation with numeric columns, refer to [18].

4.2 Raw Table Generation

In this step, we go through the list of classes from the output of the profiling, and generate a set of SPARQL queries for each class. This way, each table will have one class as the main topic with each row containing values from the properties of an instance of the class (or its subclasses, if any). In order to pick a set of properties for each class to turn into a set of columns in the output table, we use a simple randomized method. We use the gathered statistics only to avoid properties with very few instance values that could in turn result in SPARQL queries with empty or very small result set. For each class, we randomly select a number of properties within a predefined range. For the tables generated for the challenge, we select a minimum of 3 and a maximum of 7 columns for each table. We then create a query to retrieve the (primary) label of the instance along with labels of object properties and values of data type properties. When multiple values are present, we only select a single value for the corresponding cell. We also ensure in the query that the type of the object property matches the expected range in the ontology (if any) since, particularly in DBpedia, there might be objects of various types as property values of the same property.

Finally, we need to ensure a diversity of classes in the output and a balanced collection in terms of table size so that we avoid very small tables, and larger classes (e.g. Person in DBpedia) do not end up dominating the collection. For small query result sets (less than 5 rows for this edition), we drop the query and try selecting a new random subset and repeat the process until all properties are included or no new tables can be generated. To deal with larger classes, we break larger query results into randomly sized subsets, and ensure that we do not have more than a fixed number (5 for this edition) of tables for the same

query, and no more than a fixed number (2,000 for this edition) of rows across the collection for a given class.

The final outcome is a collection of SPARQL queries, each resulting in tabular data with *(i)* columns that can be annotated with the expected type (class for the case of object properties), *(ii)* cell values that can be annotated with instance URIs, and *(iii)* pairs of columns that can be annotated with a property.

4.3 Refinement

The outcome of the previous step is a collection of tables with all their contents completely based on values in the source (English DBpedia for this edition) which is somewhat unrealistic as real tables often have noise as well as columns/rows/values that cannot be matched with our knowledge source. For this edition, we implemented only a few simple refinement strategies to make the tables more realistic and so the matching task more difficult. We plan to significantly improve this refinement step to create more realistic collections and also collections geared towards particular features, e.g., the ability to handle certain kinds noise or the so-called "NIL detection".

The first simple refinement step includes adjusting some label values in a rule-based approach. For this edition, we do this only for Person entities, by abbreviating first names. It is possible to do this string value manipulation based on introducing errors (e.g. typos, using the method used in the UIS data generator [14]) or using sources of synonym terms and alternative labels.

To further make the matching tasks more challenging, we have used another refinement process which is applied over a number of automatically generated collections (which differ due to the random creation of SPARQL queries described above). The goal of this refinement is to retain only a subset of benchmark tables from the generated collections, after discarding fairly easy matching cases. This process can be further divided into three sub-processes: (*i*) identifying tables in which the matching tasks is more challenging, (*ii*) identifying rows in a challenging table that are still fairly easy to match (CEA task), and (*iii*) adapting the benchmark tables and the ground truths accordingly.

For sub-process (*i*), we use the so-called refined lookup approach [8] to identify more challenging tables. In summary, this two-step approach first looks up the contents of each table cell in a KB index, and for each top-ranked result, it stores its rdf:type. In the second step, it performs the same lookup operation, but this time, it restricts the results to only those belonging to the 5 most frequent types per column, as retrieved from the first step. Despite its simplicity, this approach provided decent effectiveness results compared to more sophisticated methods. We set an empirical threshold for F1-score (0.4), and we report all the tables for which the simple lookup method returns an F1-score lower than the threshold. The tables in the final benchmark dataset will only consist of tables that are reported in this step, i.e., easier tables are ignored.

For sub-process (*ii*), we scan in depth the error logs of the previous subprocess, in which we report how many wrong results were reported per row and per column in a table. We remove the rows for which the simple baseline method

provided only correct results (0 errors), as long as the pruned table has more than 3 rows. Finally, for sub-process (*iii*), we adapt the ground truth files to reflect the refinement step. We first remove all the information about tables that were entirely discarded, and for the remaining tables, we adapt the row numbering to reflect the changes made in sub-process (*ii*).

4.4 RDF Data

In order to allow ontology matching tools to use our benchmark datasets, we also provide our datasets in RDF format, as described by a simple OWL ontology that we generate automatically from the tables [9]. Note that this process is currently only applicable when column headers are available in a table.

In summary, we assume that each table corresponds to an OWL Class, with each row being an instance of this class. The table columns correspond to either data type or object properties, which have as domain the class corresponding to the table. We detect a special label column (using heuristics, as in [8,26]), which we use as the rdfs:label property. Based on the values of each column we define the range of each data property (e.g., xsd:integer, xsd:date, xsd:string) and object property. In the case of object properties, the range class is defined as a new class, named after the header of the corresponding column. This way, the values for the columns that describe object properties are treated as instances of the OWL class, which is the range of this column.

In the example of Table 1a, assume that we have an additional row at the beginning, with the values: "Country", "Capital". In that example, we would create an OWL ontology with the classes *Country* and *Capital*, and the object property *hasCapital*. The OWL class describing the table would be *Country*, and this class would also be the domain of all the properties (in this case only *hasCapital*). The range of *hasCapital* would be the class *Capital*. Finally, each row in the table corresponds to an instance of a Country, with the rdfs:label of each instance defined from the value of the *Country* column (which is determined as the label column). For example, the RDF triples generated for the first row would be: `:China rdf:type :Country`, `:China rdfs:label` "`China`", `:China :hasCapital :Beijing`, and `:Beijing rdf:type :Capital`.

5 Benchmarking Systems

The 2019 edition of the SemTab challenge was collocated with the *18th International Semantic Web Conference* as a Semantic Web Challenge and with the *14th Ontology Matching workshop* as a special OAEI evaluation track.

5.1 Evaluation Methodology

The SemTab challenge started in mid April and closed in mid October 2019. It was organised into four evaluation rounds where we aimed at testing different datasets with increasing difficulty.

Table 2. Statistics of the datasets in each SemTab round.

	Round 1	Round 2	Round 3	Round 4
Tables #	64	11,924	2,161	817
Avg. rows # (± Std Dev)	142 ± 139	25 ± 52	71 ± 58	63 ± 52
Avg. columns # (± Std Dev)	5 ± 2	5 ± 3	5 ± 1	4 ± 1
Avg. cells # (± Std Dev)	696 ± 715	124 ± 281	313 ± 262	268 ± 223
Target cells # (CEA)	8,418	463,796	406,827	107,352
Target columns # (CTA)	120	14,780	5,752	1,732
Target column pairs # (CPA)	116	6,762	7,575	2,747

Evaluation Framework. We relied on AIcrowd[7] as the platform to manage the SemTab challenge tasks: CTA, CEA and CPA. AIcrowd provides a number of useful functionalities such as challenge presentation, participant registration, automatic evaluation, ranking, submission limitation, and so on. For the (automatic) evaluation, an AIcrowd Python code template was provided, according to which the SemTab evaluation interface and metrics were implemented and deployed [3].

Datasets and Rounds. Table 2 provides a summary of the statistics of the datasets. For example, Round 3 dataset was composed of 2,161 tables; there were 406,827 target cells in CEA, 5,752 target columns in CTA, and 7,575 target column pairs in CPA. Round 1 was based on the T2Dv2 dataset [19] and served as *sandbox* for the participating systems. As T2Dv2 provides only class annotations at table level, for CTA, we extended the annotation of types for the other (entity) columns. We also manually revised the original and the new column types. Round 2 dataset was composed of *(i)* 10,000 relatively clean tables from the Wikipedia tables presented in [8] (i.e., not including tables with multiple column/row span, and large textual cell contents as in [8], and *(ii)* an automatically generated dataset of 1,924 tables as described in Sect. 4. Rounds 3 and 4 were composed of an automatically generated dataset with enhanced characteristics and a focus on non-trivial annotations. The ground truth for all four rounds was based on DBpedia. In this edition of the challenge, the ground truth was blind during the competition; but the target cells, columns and column pairs were provided to the participants.

Format of Solutions. Participants executed the matching tasks as defined in Sect. 2 for each of the given target table elements. The solutions for the CEA task were expected in a file with lines having these fields: "Table ID", "Column ID", "Row ID" and "DBpedia entity (only one)" (*e.g.*, "table1", "0", "121", "dbr:Norway"). Similarly, CPA solutions had the following fields per line: "Table ID", "Head Column ID", "Tail Column ID" and "DBpedia property (only one)" (*e.g.*, "table1", "0", "1", "dbo:releaseDate"). For CTA, more than

[7] https://www.aicrowd.com/.

Table 3. Schedule of submissions in each round.

	Round 1	Round 2	Round 3	Round 4
Opening	April 15	July 17	Sept. 23	Oct. 15
Closing	June 30	Sept. 22	Oct. 14	Oct. 20

one type annotations, separated by a space, were accepted: 'Table ID", "Column ID" and "DBpedia classes (1 or more)" (*e.g.*, "table1", "0", "1", "dbo:Country dbo:Place"). Note that those annotations outside the targets were ignored. Multiple annotations to one target cell or column pair, and multiple lines associated to the same target element returned an error.

Submission and Schedule. Participants had to submit their solutions for the three matching tasks via the AIcrowd platform. The performance scores were automatically computed and systems were publicly ranked in the AIcrowd webpages.[8] In Rounds 1 and 2, the number of submissions was unlimited so that participants could fine-tune their systems. The number of submissions per day was limited in Rounds 3 and 4 to avoid the effect of over-tuning. Table 3 shows the opening and closing dates for each round. The objective of Round 4 and the limited time also aimed at identifying potential over-tuning in the participating systems.

Evaluation Metrics for CEA and CPA. For CEA and CPA, we compute Precision P, Recall R and $F1$ Score (primary score) as follows:

$$P = \frac{|\text{Correct Annotations}|}{|\text{System Annotations}|} \quad R = \frac{|\text{Correct Annotations}|}{|\text{Target Annotations}|} \quad F1 = \frac{2 \times P \times R}{P + R} \tag{1}$$

where target annotations refer to the target cells for CEA and the target column pairs for CPA. Note that it is possible that one target cell or column pair has multiple ground truths, as modern KGs often have duplicate components. One example is the wiki page redirected entities in DBpedia. An annotation is regarded as true if it is equal to one of the ground truths. The comparison for equality is case insensitive. Recall that at most one annotation was submitted for each target cell or column pair.

Evaluation Metrics for CTA. For CTA we used a different set of metrics to take into account the taxonomy (hierarchy) of classes in the KG, namely *Average Hierarchical Score* (AH) and *Average Perfect Score* (AP):

$$AH = \frac{|P| + 0.5 \times |O| - |W|}{|T|} \qquad AP = \frac{|P|}{|P| + |O| + |W|} \tag{2}$$

T denotes all the columns for annotation. We refer as *perfect* annotations (P) the most fine-grained classes in the (ontology) hierarchy that also appear in

[8] E.g., CEA leaderboard: https://www.aicrowd.com/challenges/iswc-2019-cell-entity-annotation-cea-challenge/leaderboards.

Table 4. Participation in the SemTab challenge.

	Round 1	Round 2	Round 3	Round 4
Overall	17	11	9	8
CEA task	11	10	8	8
CTA task	13	9	8	7
CPA task	5	7	7	7

the ground truth, while annotations involving the super-classes (excluding very generic top classes like *owl:Thing*) of perfect classes are referred to as *okay* annotations (O). Other annotations not in the ground truths are considered as *wrong* (W). *AH* gives a full score to the *perfect* annotation, a half score to the *okay* annotations, and a negative score to *wrong* class annotation. *AH* is used as the primary score as it considers both correct and wrong annotations, while *AP* is used as secondary score as it only considers the rate of perfect annotations.

5.2 Challenge Participation

Table 4 shows the participation per round. We had a total of 17 systems participating in Round 1. Round 2 had a reduction of participating systems (from 17 to 11), which helped us identify the core systems and groups actively working in tabular data to KG matching. Round 3 and Round 4 preserved the 7 core participants across rounds and all three tasks. It is worth mentioning that LogMap [17], a pure ontology alignment system, participated in Round 2. LogMap was given as input *(i)* the tabular data in RDF format as described in Sect. 4.4, and *(ii)* a relevant portion of the DBpedia KG. The obtained results were reasonable, but far from the specialised system in the challenge. This is expected as systems like LogMap rely on the semantics of the input ontologies or KGs, which is missing in the input tabular data.

Next, we provide a brief description of the core participants, who also submitted a system paper to the challenge.

MTab [24]. MTab is a system that can jointly deal with the three tasks CTA, CEA and CPA. It is based on the joint probability distribution of multiple table to KG matching, following the probabilistic graph model by [20]. However, the team improves the matching by using multiple services including DBpedia Lookup, DBpedia endpoint, Wikipedia and Wikidata, as well as a cross-lingual matching strategy.

IDLab [27]. The IDLab team developed an iterative matching procedure named CSV2KG with the following steps: *(i)* gets crude entity matching with cells; *(ii)* determines the column types and column relations with these entities; *(iii)* corrects the cell to entity matching with the column types and column relations; *(iv)* corrects the remaining cells with the head cells; and *(v)* calculates the column type again with all the corrected cell to entity matching.

ADOG [25]. This systems utilizes a NoSQL database named ArangoDB[9] to load DBpedia and index its components. ADOG then matches tabular data with the entities of DBpedia using Levenshtein distance, a string similarity metric.

Tabularisi [28]. The team developed a system with two steps: candidate generation and selection. The former uses the Wikidata API and a search index based on DBpedia labels to obtain a list of entities for each cell, while the later scores the candidates with lexical features which are based on lexical similarity metrics, and semantic features which capture the cell coherence of each column.

DAGOBAH [2]. This participant system proposes an embedding approach which assumes that entities in the same column should be closed in the embedding space. It gets candidate entities by KG lookup, and uses pre-trained Wikidata embeddings for entity clustering and cluster type scoring. The challenge of this method lies in the setting of hyper parameters such as the cluster number.

Team_sti [6]. This team developed a tool named MantisTable that can automatically annotate, manage and make the semantics of tables accessible to humans and machines. The tool has some built in functions for the three matching tasks, including a SPARQL query for entity matching, a relation annotator based on maximum frequency and a class annotator based on voting by entities. Note that this system also provides a web interface for manual annotation.

LOD4ALL [23]. This system implements a pipeline for the three tasks with five steps: *(i)* extracts ranked candidate entities of cells with direct search by ASK SPARQL queries and keywords; *(ii)* gets the type of each entity; *(iii)* determines the type of each column with a weighted combination of ratio score and a normalized class score; *(iv)* determines the entity of each cell with the type constraint; and *(v)* extracts the relation of entities in each row and select the inter-column relation by frequency.

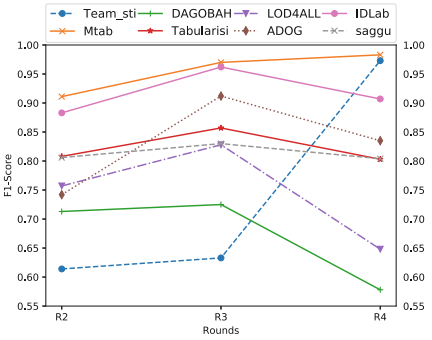
5.3 Challenge Evaluation

In this section, we report the results of the challenge Rounds 2–4 for the systems participating in at least two rounds, which include the above core participants and a system called *saggu* that only participated in CEA. Complete evaluation results are available from the challenge website [12].

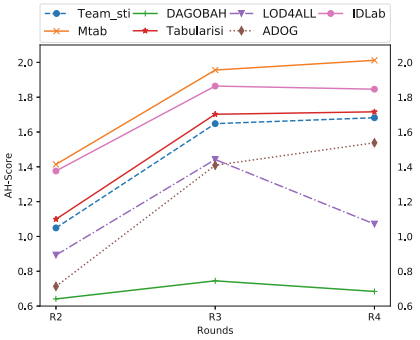
The results for all three matching tasks are presented in Fig. 2. MTab and IDLab were the clear dominants in all three tasks. Tabularisi was in a clear overall 3rd position in CTA and CPA. The overall 3rd position in CEA was shared among Tabularisi and ADOG. Special mention requires Team_sti which had an outstanding performance in Round 4 of CEA.

In terms of average scores, Round 2 was the most challenging one, although it is not comparable to Rounds 3 and 4 as it includes a different source dataset. Rounds 3 and 4 completely rely on the dataset generator. Round 4 aimed at being more challenging than Round 3 by only including non-trivial cases. This was partially achieved for CEA, with the exception of MTab and Team_sti.

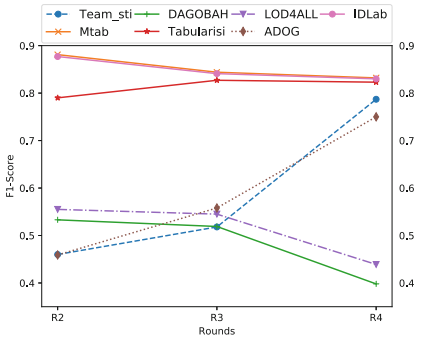
[9] https://www.arangodb.com.



(a) CEA Task



(b) CTA Task



(c) CPA Task

System	CEA	CTA	CPA
MTab	**1.0**	**1.0**	**1.0**
IDLab	**2.3**	**2.0**	**2.0**
Tabularisi	**4.3**	**3.0**	**3.0**
ADOG	**4.3**	5.7	5.3
Team_sti	6.0	4.0	5.9
LOD4ALL	6.0	5.3	5.0
DAGOBAH	7.3	6.7	6.0
saggu	4.7	-	-

(d) Average ranking per task

Fig. 2. Results of systems competing in challenge Rounds 2, 3 and 4.

The relative performance of systems across rounds is similar in CEA and CTA with the exception of Team_sti in CEA, where there is an important improvement in Round 4, and LOD4ALL that decreased performance in Round 4 of CTA.

According to the results, complementing DBpedia with additional resources like Wikidata (*e.g.*, MTab and Tabularisi) brings an important value. In general, the use of elaborated lexical techniques seems to be the key for a good performance. Other approaches based on more sophisticated methods like semantic embeddings (*e.g.*, DAGOBAH) do not seem to bring the expected value to the final performance, but they may suffer a lighter impact with respect to changes in the datasets and the KG. Another factor that may impact their performance is the long time spent for learning or fine tuning the embeddings of a large KG like Wikidata and DBpedia.

Sponsorship and Awards. SIRIUS[10] and IBM Research[11] sponsored the prizes for the challenge. This sponsorship was important not only for the challenge awards, but also because it shows a strong interest from industry. Figure 2d shows the

[10] SIRIUS: Norwegian Centre for Research-driven Innovation: https://sirius-labs.no.
[11] https://www.research.ibm.com/.

average ranking of the participating systems in each task. MTab, IDLab and Tabularisi obtained the *1st*, *2nd* and *3rd* prize, respectively, across the three matching task. ADOG shared the *3rd* prize in CEA with Tabularisi. Finally, Team_sti obtained the *Outstanding Improvement* prize in CEA.

6 Lessons Learned and Future Work

In this paper, we have presented the datasets and the results of the first edition of the SemTab challenge. The experience has been successful and has served to start creating a community interested in the semantic enrichment of tabular data. Both from the organization side and the participation side, we aim at preparing a new edition of the SemTab challenge in 2020. Next, we summarize the issues we encountered during the different evaluation rounds, the lessons learned, and some ideas for the future editions of the challenge.

Importance of the Challenge. We received very positive feedback from the participants with respect to the necessity of a challenge like SemTab to conduct a systematic evaluation of their systems. Our challenge was also well-received from industry via the sponsorship of IBM Research and SIRIUS.

Minor Issues. We faced a few minor issues during the evaluation rounds, which will help us improve the future editions of the challenge. Next, we summarise some of them: *(i)* explicit reference to the version of the KG used; *(ii)* incompatible encodings when merging different datasets; *(iii)* low quality of the DBpedia wikiredirects; *(iv)* Wikipedia disambiguation pages as annotations; *(v)* property hierarchy was not considered; *(vi)* the average Hierarchical Score (AH) was not easy to interpret for participants as, in the way it is currently defined, it does not have a clear upper bound. Nevertheless, we believe these issues affected all participants in a similar way and they did not have an important impact in the relative comparison among systems.

Evaluation Platform. On the one hand, the AIcrowd platform makes the management of submissions, evaluation and ranking very easy. On the other hand, it has no interface for automatic deployment of the evaluation codes and data, which makes it inconvenient to deal with online errors or changes, as challenge organisers depend on the AIcrowd team. It was also hard to communicate with participants not using the AIcrowd discussion forum. For next editions, we may consider alternative solutions.

Number of Submissions. The limitation of number of submission per day was not welcomed by all participants. However, we find that unlimited submissions may lead to over-tuning the matching model that will have limited generalization performance. In future editions, we will try to better split the datasets for fine-tuning from the ones for testing.

Instance Matching. We produced an RDF version of the dataset in Round 2, but we did not attract the expected attention in the OAEI community and the participation of (ontology) instance matching or link discovery systems was

limited to LogMap. In future editions of the challenge, we aim at facilitating the participation of OAEI systems.

Real-World Datasets. Several participants highlighted the necessity of more realistic datasets, however manually annotated datasets are limited in quantity and size. A possible solution is to create a consensual ground truth by combining the output of several systems. This solution has already been used in several evaluation tracks of the OAEI campaign [1].

Reproducibility. As SemTab 2019 was the first edition of the challenge, our priority was to facilitate participation and allow participants to directly submit their solutions for each matching task. This plays a negative role in terms of reproducibility of the results. In future editions, we are considering to require from participants *(i)* the submission of a running system as in the OAEI campaign, or *(ii)* the publication of their system as a (Web) service.

Matching Targets. In SemTab 2019 we advocated to provide this information to the users to make the matching and the evaluation easier. In future editions we may hide this information to the participants. Participants will have less guidance which will especially be reflected in the CPA task. Evaluation will also be more challenging as incompleteness of the ground truth should not penalize potentially correct predictions.

Improved Data Generator. As outlined in Sect. 4, there are a number of ways to improve our data generator to create more realistic datasets. In particular, much work needs to be done in creating tables that are more challenging to match, and contain more variety of representations and contents that cannot be matched to the source KG. Also, our data generator has a number of parameters which can be adjusted to create different benchmarks each suitable for a different use case. We intend to work on these extensions, create more diverse and realistic collections, and make our data generator publicly available which will allow us to seek contributions from the community.

Acknowledgements. We would like to thank the challenge participants, the ISWC & OM organisers, the AIcrowd team, and our sponsors (SIRIUS and IBM Research) that played a key role in the success of SemTab. This work was also supported by the AIDA project (Alan Turing Institute), the SIRIUS Centre for Scalable Data Access (Research Council of Norway), Samsung Research UK, Siemens AG, and the EPSRC projects AnaLOG, OASIS and UK FIRES.

References

1. Algergawy, A., et al.: Results of the ontology alignment evaluation initiative 2018. In: 13th International Workshop on Ontology Matching, pp. 76–116 (2018)
2. Chabot, Y., Labbe, T., Liu, J., Troncy, R.: DAGOBAH: an end-to-end context-free tabular data semantic annotation system. In: SemTab, ISWC Challenge (2019)
3. Chen, J., Efthymiou, V., Hassanzadeh, O., Jiménez-Ruiz, E., Srinivas, K.: AIcrowd Evaluation Codes (Python code). https://github.com/sem-tab-challenge/aicrowd-evaluator. Accessed 6 Mar 2020

4. Chen, J., Jimenez-Ruiz, E., Horrocks, I., Sutton, C.: Learning semantic annotations for tabular data. In: IJCAI (2019)
5. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.A.: ColNet: embedding the semantics of web tables for column type prediction. In: AAAI, pp. 29–36 (2019)
6. Cremaschi, M., Avogadro, R., Chieregato, D.: MantisTable: an automatic approach for the semantic table interpretation. In: SemTab, ISWC Challenge (2019)
7. Delpeuch, A.: OpenTapioca: lightweight entity linking for Wikidata. arXiv preprint arXiv:1904.09131 (2019)
8. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10587, pp. 260–277. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68288-4_16
9. Efthymiou, V., Hassanzadeh, O., Sadoghi, M., Rodriguez-Muro, M.: Annotating web tables through ontology matching. In: OM, pp. 229–230 (2016)
10. Euzenat, J., Rosoiu, M., dos Santos, C.T.: Ontology matching benchmarks: generation, stability, and discriminability. J. Web Semant. **21**, 30–48 (2013)
11. Euzenat, J., Shvaiko, P.: Ontology Matching, 2nd edn. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38721-0
12. Hassanzadeh, O., Efthymiou, V., Chen, J., Jiménez-Ruiz, E., Srinivas, K.: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2019) (2019). http://www.cs.ox.ac.uk/isg/challenges/sem-tab/2019. Accessed 6 Mar 2020
13. Hassanzadeh, O., Efthymiou, V., Chen, J., Jiménez-Ruiz, E., Srinivas, K.: SemTab2019: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching - 2019 Data Sets (2019). https://doi.org/10.5281/zenodo.3518539. Accessed 6 Mar 2020
14. Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. In: ACM SIGMOD Conference on Management of Data, pp. 127–138 (1995)
15. Hu, K., et al.: VizNet: towards a large-scale visualization learning and benchmarking repository. In: CHI. ACM (2019)
16. Hulsebos, M., et al.: Sherlock: a deep learning approach to semantic data type detection. In: Knowledge Discovery and Data Mining (KDD) (2019)
17. Jiménez-Ruiz, E., Cuenca Grau, B.: LogMap: logic-based and scalable ontology matching. In: Aroyo, L., et al. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 273–288. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25073-6_18
18. Kacprzak, E., et al.: Making sense of numerical data - semantic labelling of web tables. In: Faron Zucker, C., Ghidini, C., Napoli, A., Toussaint, Y. (eds.) EKAW 2018. LNCS (LNAI), vol. 11313, pp. 163–178. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03667-6_11
19. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: WWW (2016)
20. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. VLDB Endow. **3**(1–2), 1338–1347 (2010)
21. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: I-Semantic, pp. 1–8. ACM (2011)
22. Mihindukulasooriya, N., Poveda-Villalón, M., García-Castro, R., Gómez-Pérez, A.: Loupe - an online tool for inspecting datasets in the linked data cloud. In: ISWC Posters & Demos (2015)
23. Morikawa, H.: Semantic table interpretation using LOD4ALL. In: SemTab, ISWC Challenge (2019)

24. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: MTab: matching tabular data to knowledge graph using probability models. In: SemTab, ISWC Challenge (2019)
25. Oliveira, D., d'Aquin, M.: ADOG - anotating data with ontologies and graphs. In: SemTab, ISWC Challenge (2019)
26. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML Tables to DBpedia. In: WIMS, pp. 10:1–10:6 (2015)
27. Steenwinckel, B., Vandewiele, G., De Turck, F., Ongenae, F.: CSV2KG: transforming tabular data into semantic knowledge. In: SemTab, ISWC Challenge (2019)
28. Thawani, A., et al.: Entity linking to knowledge graphs to infer column types and properties. In: SemTab, ISWC Challenge (2019)
29. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semant. Web **8**(6), 921–957 (2017)