MTab4D: Semantic Annotation of Tabular Data with DBpedia

Phuc Nguyen [a,*], Natthawut Kertkeidkachorn [b], Ryutaro Ichise [a], Hideaki Takeda [a]

[a] *National Institute of Informatics, Japan, Tokyo*
E-mails: phucnt@nii.ac.jp, ichise@nii.ac.jp, takeda@nii.ac.jp
[b] *Japan Advanced Institute of Science and Technology, Japan Ishikawa*
E-mail: natt@jaist.ac.jp

Abstract. Semantic annotation of tabular data is the process of matching table elements with knowledge graphs. As a result, the table contents could be interpreted or inferred using knowledge graph concepts, enabling them to be useful in downstream applications such as data analytics and management. Nevertheless, semantic annotation tasks are challenging due to insufficient tabular data descriptions, heterogeneous schema, and vocabulary issues. This paper presents an automatic semantic annotation system for tabular data, called MTab4D, to generate annotations with DBpedia in three annotation tasks: 1) Cell-Entity (CEA), 2) Column-Type (CTA), and 3) Column Pair-Property (CPA). In particular, we propose an annotation pipeline that combines multiple matching signals from different table elements to address schema heterogeneity, data ambiguity, and noisiness. Additionally, this paper provides insightful analysis and extra resources on benchmarking semantic annotation with knowledge graphs. Experimental results on the original and adapted datasets of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2019) show that our system achieves an impressive performance for the three annotation tasks. MTab4D's repository is publicly available at https://github.com/phucty/mtab4dbpedia

Keywords: table annotation, tabular data annotation, knowledge graph, DBpedia

1. Introduction

Many tabular data resources have been made available on the Web and data portals, thanks to the Open Data initiative in recent years. The resources contain valuable information that helps establish transparency, improve human life quality, and inspire business opportunities. Although tabular data offers enormous potential, it is difficult to be used in applications due to insufficient descriptions, heterogeneous schema, and vocabulary issues.

One possible solution for the usability problems is to generate semantic annotation of tables, particularly matching table elements with knowledge graphs (KB) such as DBpedia. As a result, the meaning of tabular data could be interpreted or inferred by knowledge graph concepts; therefore, it is easy to be used in other downstream applications such as data analytics and management.

This paper presents MTab4D, an semantic annotation system for tabular data designed to address the three annotations tasks of the Semantic Web challenge on tabular data annotation with knowledge graphs (SemTab 2019)[1]. SemTab 2019 is a systematic benchmark for promoting a comparison of state-of-the-art annotation systems[1]. Fig. 1 illustrates the three semantic annotation tasks. Cell-Entity annotation (**CEA**) is the task of assigning an entity to a table cell (Fig. 1a). Column-Type annotation task (**CTA**) assigns entity types (e.g., DBpedia class hierarchy) to a table column (Fig. 1b). Column Pair-Property annotation (**CPA**) is the task of assigning a property or a predicate to the relation between two table columns (Fig. 1c).

[*]Corresponding author. E-mail: phucnt@nii.ac.jp.

[1]SemTab 2019: http://www.cs.ox.ac.uk/isg/challenges/sem-tab/

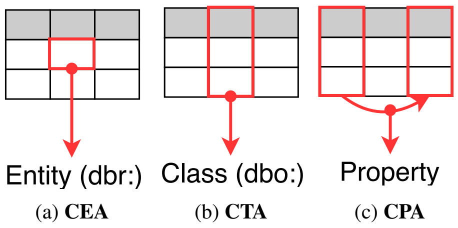


Fig. 1. Tabular Data Annotations with DBpedia. dbr: is an entity prefix, and dbo: is an type prefix

Fig. 2 depicts an example of semantic annotations for tabular data. The entity of dbr:Arthur_Drews is an annotation for a table cell of "A. Drews". The type annotations for the column "col1" are dbo:PopulatedPlace and dbo:Place. The property of dbo:deathYear is the annotation for the relation between the column "col0" and the column "col2".

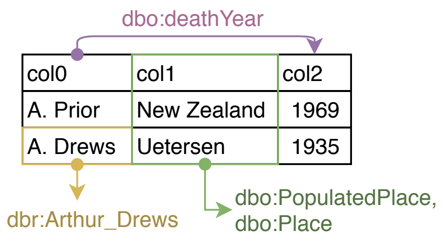


Fig. 2. An example of semantic annotation of tabular data

This paper proposes an annotation pipeline that combines multiple matching signals from different table elements to address schema heterogeneity and ambiguity. Our system is inspired by the graphical probability model-based approach [2] and the signal (or confidence) propagation as in the T2K system [3]; however, our system improves the annotation tasks' performance with the contributions summarized as follows.

- **Table Pre-processing**: We introduce a method for pre-processing tables to deal with data noisiness consisting of cell normalization, data type prediction for cells and columns, header prediction, and subject column prediction.
- **Entity Search**: Most previous systems used online entity search services to generate entity candidates (e.g., DBpedia lookup); however, it is hard to reproduce the same search results if the search index change due to the evolution of the knowledge graph. We introduce novel entity search modules[2] (keyword search, fuzzy search, and aggregation search) based on the dump of October 2016 version of DBpedia, that enable the entity search results to be reproduced for future studies.
- **Numerical Column Matching**: Data matching for numerical columns is challenging because the corresponding value in KG is rarely equal to a query value. Therefore, we adopt EmbNum+ [4] as the semantic labeling for numerical columns to find relevant properties. Then, we use DBpedia ontology to infer entity types which are the domains of the relevant properties.
- **Column-based Matching**: We introduce column-based matching between the table subject column and entity columns, and literal columns. The novel column-based signals could enhance the overall matching performance.
- **Public APIs and Graphical Interface:** We provide public tabular data annotation APIs, and graphical interface for demonstration[3], and instruction on replicating MTab4D experiment results[4]. Our implementation supports multilingual tables and could process many table formats such as Excel, CSV, TSV, or markdown tables. According to Wang et al., they only could generate the annotations using our system, while other annotation systems require high time complexity. [5].

Additionally, we also provide the contributions to the tabular data annotation community as follows.

- **Data Analysis:** This paper provides insightful analysis and extra resources (an adapted SemTab 2019 with the October 2016 version of DBpedia) on benchmarking semantic annotation with knowledge graphs [6].
- **Public Resources:** We also introduce APIs built on the October 2016 version of DBpedia (similar to SemTab 2019 setting) to standardize the evaluation of the tabular data annotation with a knowledge graph. Since the knowledge graph changes over time, it is hard to compare other systems that used a different version of the knowledge graph. Besides, the APIs aim to improve the reproducibility of sub-tasks of tabular annotation relating to entity search, entity information retrieval, numerical attribute retrieval, and evaluation retrieval.

[2]Entity Search: https://dbpedia.mtab.app/search
[3]Graphical Interface: https://dbpedia.mtab.app
[4]Repository: https://github.com/phucty/mtab4dbpedia

MTab4D is an extended version of our work MTab [7] (the best performance for the three matching tasks: the 1st rank in all (four) rounds and all (three) tasks); however, this paper advances the previous study as reproducibility and insightful analysis on the ground truth dataset. We introduce the new entity search and various public resources built on the dump data of the October 2016 version of DBpedia. These resources enable MTab4D to be reproduced for the future studies, while it is hard to replicate the MTab's performance relied on the public services (SemTab 2019).

The rest of this paper is organized as follows. In Section 2, we define the annotation tasks and describe MTab4D assumptions. Then, we present the overall framework and the details of each framework's module in Section 3. Section 4 outlines MTab4D public APIs and graphical interfaces. Section 5 provides the main differences between MTab4D and our previous studies MTab[7]. In Section 6, we report experimental settings and results. Section 7 discusses the related work on semantic annotation of table data and summarizes the participant approaches. Finally, we summarize the paper and discuss future directions and the lessons learned from the challenge in Section 8.

2. Definitions and Assumptions

In this section, we provide a formal definition for the three annotation tasks in Section 2.1. The assumptions on MTab4D are described in Section 2.2.

2.1. Problem Definitions

2.1.1. Knowledge Graph

Let $G = (E, T, P, \text{Triples})$ be the DBpedia knowledge graph, where E, T, P are the set of entities, the set of types (or classes), the set of properties (predicates), and the set of triples (graph edges) respectively. An entity is e, where $e \in E$; the type of a entity e is t_e, where $t_e \in T$; and a property is p where $p \in P$. A triple (<subject, predicate, object>) contains subject (an entity e), predicate (a property p), and object (an entity e, or a literal value such as number, string, time).

2.1.2. Tabular Data

Let S be a two-dimensional table consisting of an ordered set of N rows, and M columns. A table row is r_i, where $i = 1...N$; a table column is c_j, in which $j = 1...M$. A table cell is $S_{i,j}$ in the row r_i and the column c_j. A relation between two columns c_{j_1} and c_{j_2} is $\mathrm{R}_{c_{j_1},c_{j_2}}$, where $j_1, j_2 \in [1, M], j_1 \neq j_2$.

2.1.3. Matching Targets

Let $\mathrm{m}^S_{\mathrm{CEA}}$, $\mathrm{m}^S_{\mathrm{CTA}}$, and $\mathrm{m}^S_{\mathrm{CPA}}$ be the matching targets (indexes) of table cells, columns, and column pair relations, respectively.

2.1.4. Semantic Annotation Tasks

Given the DBpedia knowledge graph G, the table S, and matching targets $\mathrm{m}^S_{\mathrm{CEA}}$, $\mathrm{m}^S_{\mathrm{CTA}}$, $\mathrm{m}^S_{\mathrm{CPA}}$, the tables to KG matching problems could be formalized the three following tasks:

- Cell-Entity matching (**CEA**): matching the table cell $S_{i,j}$ (in the CEA matching targets, $S_{i,j} \in \mathrm{m}^S_{\mathrm{CEA}}$) into the entity e.

$$S_{i,j} \xrightarrow{\mathrm{CEA}} E \tag{1}$$

- Column-Type matching (**CTA**): matching the table column c_j (in the CTA matching targets, $c_j \in \mathrm{m}^S_{\mathrm{CTA}}$) into a class hierarchy.

$$c_j \xrightarrow{\mathrm{CTA}} T \tag{2}$$

- Column Pair Relation-Property matching (**CPA**): matching the relation between two columns $R_{c_{j_1},c_{j_2}}$ (in the CPA matching targets $R_{c_{j_1},c_{j_2}} \in \mathrm{m}^S_{\mathrm{CPA}}$) into the property p

$$R_{c_{j_1},c_{j_2}} \xrightarrow{\mathrm{CPA}} P \tag{3}$$

2.2. Assumptions

We build MTab4D system based on the following assumptions:

Assumption 1. *MTab4D is built based on a closed-world assumption.*

MTab4D annotates tabular data based on the knowledge graph information. Therefore, we assume that the knowledge graph (DBpedia) is complete and correct. The system could return incorrect answers if table elements are not available in the knowledge graph.

Assumption 2. *The tabular input data is a horizontal relational table type.*

A horizontal relational table contains semantic knowledge graph triples in <subject, predicate, object>. The table also has a subject column containing entity names, and the relation between the subject column and other table columns representing the predicate relation between the entities (subject) and attribute values (object).

Assumption 3. *Input tables are independent.*

We adopt this assumption since MTab4D treats input tables independently; therefore, we can use MTab4D for general purposes.

Assumption 4. *All the cell values of the same column have the same data type, and the entities related to cell values are of the same type.*

3. MTab4D Approach

In this section, we describe the MTab4D framework in Section 3.1. The details of each step are described in from Section 3.2 to Section 3.7.

3.1. Framework

We design our system (MTab4D) as the seven-steps pipeline as shown in Fig. 3. Step 1 pre-process the input table consisting of cell value normalization, cell and column data type prediction, header prediction, and subject column prediction. Step 2 is to generate entity candidates. Then, step 3 and step 4 generate type candidates and property candidates using the row-based aggregation from Step 2 respectively. Step 5 is to disambiguate entity candidates with confidence aggregation from step 2, step 3, and step 4. Step 6 and Step 7 are to disambiguate type and property candidates with results from Step 5, respectively.

The following are detailed explanations of each step of the framework.

3.2. Step 1: Pre-processing

We perform the four processes: cell normalization, data type prediction, header detection, and subject column prediction as follows.

3.2.1. Cell Normalization

We remove HTML tags and non-cell-values such as -, NaN, none, null, blank, unknown, ?, #. Additionally, we use the *ftfy* tool [8] to fix all noisy cells caused by incorrect encoding during file loading.

3.2.2. Data Type Prediction

The system firstly predicts a table cell's data type into either non-cell (empty cell), literal, or named-entity (NE) . We use the pre-trained SpaCy models [9] (OntoNotes 5 dataset) to identify named and numeric entities. A cell has a named-entity type when the SpaCy model recognizes a entity-name tag such as human names (PERSON), nationalities (NORP), building (FAC), companies (ORG), countries, cities (GPE), locations (LOC), objects, vehicles (PRODUCT), wars, sports events (EVENT), books, songs (WORK_OF_ART), law documents (LAW), named language (LANGUAGE). A cell has a literal type when the recognized SpaCy tag is a numeric tag such as date (DATE), time (TIME), percentage (PERCENT), amount of money (MONEY), measurements (QUANTITY), ordinal (ORDINAL), and other numerical values (CARDINAL). If there is no tag assigned, we associate the cell type as named-entity because the SpaCy model could miss recognized named-entity type.

Next, the system predicts a table column's data type into either non-match column (empty column) c^{nan}, a literal c^{lit}, or a named-entity column c^{ent}. The column data type is derived from the majority voting of all cell data types in this column.

3.2.3. Header Prediction

Let r^h be a table header. We use simple heuristics to predict table headers as follows.

- Table headers could be located in some of the first rows of a table.
- If the list of data types of the header candidate row differs from most data types of the remaining rows, the candidate is the table header. For example, the list of data types of header candidate (row) is [named-entity, named-entity, named-entity], while the list of the majority data type of remaining rows is [named-entity, literal, literal].
- We also found that the length of header text is empirically shorter or longer than the remaining data rows. If the length of values of the header candidate row is less than the 0.05 quantile or larger than the 0.95 quantiles of the length of the value of remaining rows, the candidates are the table header.

3.2.4. Subject Column Prediction

Let c^{core} be the subject column of a table. We adopt the heuristics proposed by Ritze et al. [3] as well as modify a simple heuristic to predict the subject column of a table as follows.

- A column is a subject column when its data type is a named-entity type.
- The average cell value length is from 3.5 to 200. We also add a restriction that only considers non-header cells since the length of table headers could differ from the remaining cells.

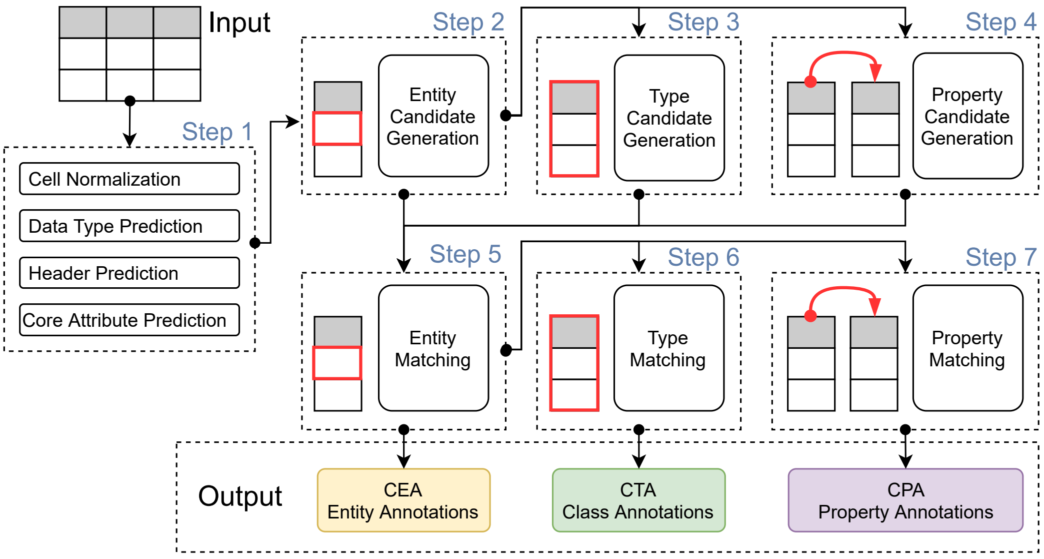


Fig. 3. MTab4D framework for tabular data annotations

– The subject column is determined based on the uniqueness score as an increased score for columns with many unique values and reduces the score for columns with many missing values. The subject column is the highest unique score column. If we have many columns that have the same score, the left-most column is chosen.

3.3. Step 2: Entity Candidate Generation

To generate entity candidates for the CEA matching targets, we perform entity searching on the MTab4D search engine[5]. Unlike previous work using online entity search services to generate entity candidates, the MTab4D entity search engine is built from DBpedia's October 2016 dump version for reproducibility. We extracted 5,226,192 entities (ignore disambiguate entities) and 35,331,799 entity labels in multilingual, including entity labels, aliases, other names, redirect entity labels, and disambiguation entity names.

We build the three entity search modules to address the ambiguity, noisiness of table cell values. Let q be a query as a table cell $S_{i,j}$; the entity search module retrieves the query from the MTab4D search engine to get a ranking list of relevant entities E_q and entities' ranking scores. We associate the ranking scores as entity candidate probabilities $Pr(e|S_{i,j})$. MTab4D search engine has three search modules: keyword search, fuzzy search, and aggregation search. The detail of search modules is described as follows.

3.3.1. Keyword Search

We build the keyword search to address table cells ambiguity and entity name variant. The ranking scores are calculated using the ranking functions of the BM25 algorithm and entity popularities. The entity popularities are pre-calculated using the PageRank algorithm on DBpedia. The ranking score of keyword search $\mathrm{s}_{\mathrm{keyword}}$ is calculated as follows.

$$\begin{aligned}\mathrm{s}_{\mathrm{keyword}}(q,e) = \alpha \cdot \mathrm{softmax}(s_{\mathrm{bm25}}(q,e)) \\ +(1-\alpha)\cdot r(e)\end{aligned} \quad (4)$$

The BM25 ranking score is $s_{\mathrm{bm25}}(q,e)$, and entity popularity is $r(e)$. We use the BM25 hyper-parameters as $b = 0.75, k_1 = 1.2$. We set $\alpha = 0.8$ as empirically putting more weighting for the ranking functions of the BM25 algorithm.

3.3.2. Fuzzy Search

Another challenge of entity search is that table cells might be noisy, containing many spelling errors, and expressed as abbreviations. We introduce the fuzzy search module using edit distance (Damerau–Levenshtein) and entity popularities. The ranking score of fuzzy search is calculated as follows.

$$\begin{aligned}\mathrm{s}_{\mathrm{fuzzy}}(q,e) = \alpha \cdot \frac{1}{(s_{\mathrm{edit}}(q,e)+1)} \\ +(1-\alpha)\cdot r(e)\end{aligned} \quad (5)$$

[5]Entity search: https://dbpedia.mtab.app/search

where $s_{\text{edit}}(q,e)$ is Damerau–Levenshtein distance between the table cell and entity label. We also use the same keyword search parameter α to empirically put more weight into the edit distance.

Since the edit distance calculation is expensive, we perform candidate filtering and hashing to reduce the number of operations on pairwise edit distance calculation. We remove entity candidates with their length larger or smaller d characters than the query's length (To be simple, we set d to six in all of our experiments). Because of the efficient reason, we only perform a fuzzy search with a maximum of six edit distances. We also perform candidate hashing with pre-calculating entity label deletion as SymSpell: Symmetric Delete algorithm [10].

3.3.3. *Aggregation Search*

This search module is designed to aggregate keyword search and fuzzy search results with a weighted fusion as the following equation.

$$s_{\text{agg}}(q,e) = \beta \cdot s_{\text{keyword}} + (1-\beta) \cdot s_{\text{fuzzy}} \tag{6}$$

We set the β parameter as 0.5 to provide an equal contribution to the keyword search and the fuzzy search.

3.4. *Step 3: Type Candidate Generation*

This step is to generate type candidates for the named-entity columns. The signal aggregation is described in Section 3.4.1. The type candidate signals including numerical column signals, entity search signals, named-entity recognition signals, and table header signals are described in Section 3.4.2, Section 3.4.3, Section 3.4.4, Section 3.4.5.

3.4.1. *Signal Aggregation*

If the table column c_j is a named-entity column (NE column) and it is also the subject column, the probabilities of type candidates are calculated from the entity search signals $Pr(t|E_{Q_{c_j}})$ (Section 3.4.3), named-entity recognition signals $Pr(t|\text{SpaCy}_{c_j})$ (Section 3.4.4), table header signals $Pr(t|r^h_{c_j})$ (Section 3.4.5), and numerical column signals $Pr(t|c^{\text{num}})$ (Section 3.4.2) as the following equation:

$$\begin{aligned} &Pr(t|c_j \text{ is a NE and a subject column}) = \\ &\quad w_1 \cdot Pr(t|E_{Q_{c_j}}) + w_2 \cdot Pr(t|\text{SpaCy}_{c_j}) \\ &\quad + w_3 \cdot Pr(t|r^h_{c_j}) + w_4 \cdot Pr(t|c^{\text{num}}) \end{aligned} \tag{7}$$

If the table column c_j is a named-entity column (NE column) and not the subject column, the probabilities of type candidates are also calculated similarly to the core attribute case but without the numerical column signals as the following equation:

$$\begin{aligned} &Pr(t|c_j \text{ is a NE and not a subject column}) = \\ &\quad w_1 \cdot Pr(t|E_{Q_{c_j}}) + w_2 \cdot Pr(t|\text{SpaCy}_{c_j}) \\ &\quad + w_3 \cdot Pr(t|r^h_{c_j}) \end{aligned} \tag{8}$$

Note that some probabilities of signals might be zero or too small, and aggregate the signals might add too much noise to the final aggregation. Therefore, if any signal probabilities less than γ[6], we omit the signals. MTab4D uses a equal contribution for the weighting of w_1, w_2, w_3, w_4, specifically $w_1 = w_2 = w_3 = w_4 = 1$. After aggregation, we also perform normalization for $Pr(t|c_j)$ to a range of [0,1] so that $Pr(T_{c_j}|c_j) = 1$.

3.4.2. *Numerical Column Signals*

Let $Pr(t|c_j^{num})$ be type candidates of the subject column given a numerical column, where c_j^{num} is the literal column so that column values are numerical. We use EmbNum+ [4] to find relevant numerical properties $P^{\text{num}}_{c_j}$ in DBpedia[7]. The confidence score of a property p is calculated as the following equation.

$$s_p^{c_j} = \text{size\_of}(P^{\text{num}}_{c_j}) - rank_p \tag{9}$$

where $rank_p$ is the ranking index of p in $P^{\text{num}}_{c_j}$. The scores are also normalized to a range of [0,1] and associated as the probability of property candidates of a numerical column $Pr(p|c_j^{\text{num}})$.

Next, we use the property candidates to infer the classes (types) for the subject column. The inferred classes are the domain class (dbo:domain) of the property candidates. For example in Fig. 4, the property candidates of the two numerical columns are "dbo:oclc" and "dbo:finalPublicationYear". The inferred type candidates of the subject column given the two numerical columns are "dbo:WrittenWork" and "dbo:PeriodcalLiterature" (the domain types of the property candidates).

Let $T_{c_j^{\text{num}}}$ be the set of inferred types from numerical columns; the types confidence probabilities are calcu-

[6]In MTab4D, we select $\gamma = 0.5$ empirically

[7]We used the 200 most frequently numerical attributes (numerical properties e.g., height, weight) of DBpedia as the database

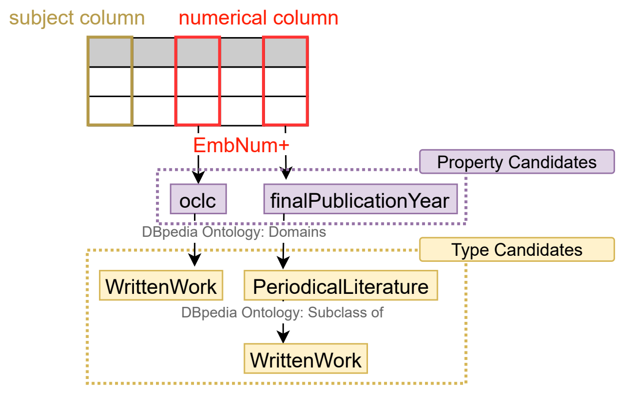


Fig. 4. Property lookup with EmbNum+

lated as the following equation.

$$Pr(t|c^{\text{num}}) = max(Pr(t|c_j^{\text{num}}) \quad (10)$$

where

$$Pr(t|c_j^{\text{num}}) = Pr(p|c_j^{\text{num}}) \quad (11)$$

where t is the domain type of the property p. We normalize the type candidate probabilities $Pr(t|c^{\text{num}})$ to [0,1] by dividing each element with the total sum.

3.4.3. *Entity Search Signals*

Let Q_{c_j} be the list of cell values (queries), $E_{Q_{c_j}}$ be the list of entity candidates derived from the entity search modules, the type candidate signals $Pr(t|E_{Q_{c_j}})$ (from the entity types of all cells in the column c_j) are calculated as the following equation:

$$Pr(t|E_{Q_{c_j}}) = \sum_{q \in Q_{c_j}} Pr(t_e), e \in E_q \quad (12)$$

where

$$Pr(t_e) = Pr(e|E_q) \quad (13)$$

We normalize the signals to a range [0, 1] so that $Pr(T_{c_j}|E_{Q_{c_j}}) = 1$ (divide each element with the total sum), where T_{c_j} is the type candidates in the column c_j.

3.4.4. *Named-Entity Recognition Signals*

$Pr(t|\text{SpaCy}_{c_j})$: The type candidates aggregated from SpaCy named-entity recognition (Section 3.2.2) for each cell in the column c_j. The mappings between SpaCy named entities and DBpedia entity types are described in Table 1. The type candidate signals given the mapping between SpaCy named entities and DBpedia classes are calculated as the following equation:

Table 1
Mappings between SpaCy named entities and DBpedia entity types

SpaCy Tags	**DBpedia Entity Types**
PERSON	dbo:Person
NORP	dbo:Country, dbo:Religious, dbo:PoliticalParty
FAC	dbo:PopulatedPlace, dbo:Building, dbo:RouteOfTransportation, dbo:Airport
ORG	dbo:Organization
LOC	dbo:PopulatedPlace
GPE	dbo:PopulatedPlace
PRODUCT	dbo:Device, dbo:Food
EVENT	dbo:Event
WORK_OF_ART	dbo:Work
LAW	dbo:LawFirm
LANGUAGE	dbo:Language

$$Pr(t|\text{SpaCy}_{c_j}) = \sum_{S_{i,j} \in c_j} Pr(t|\text{SpaCy}(S_{i,j})) \quad (14)$$

where

$$Pr(t|\text{SpaCy}(S_{i,j})) = \begin{cases} 1, \text{if } \exists \text{ TypeMap}((\text{SpaCy}(S_{i,j})) \\ 0, \text{others} \end{cases} \quad (15)$$

where the $\text{SpaCy}(S_{i,j})$ is the SpaCy named-entity recognition function, $\text{TypeMap}((\text{SpaCy}(S_{i,j}))$ is the mapping between SpaCy named entities to DBpedia classes. We normalize the signals to range [0, 1] so that $Pr(T_{c_j}|\text{SpaCy}_{c_j}) = 1$. The normalization is done by dividing each element with the total sum.

3.4.5. *Table Header Signals*

The table header signals is $Pr(t|r^h_{c_j})$ calculated as the type candidates given a lexical similarity (normalized Damerau–Levenshtein distance) between the table header $r^h_{c_j}$ and DBpedia classes.

3.5. *Step 4: Property Candidate Generation*

This step is to generate property candidates $Pr(p|r_{c_{j_1},c_{j_2}})$ of the relation $R_{c_{j_1},c_{j_2}}$ between two columns c_{j_1} and c_{j_2}. We assume that the input table is a horizontal relational type so that this step focuses on the relation of the subject column to an entity column and the subject column to a literal column. To be simple, we associate the first subject column is $c^{\text{core}}_{j_1}$, an entity column is $c^{ent}_{j_2}$, and a literal column is $c^{lit}_{j_2}$.

3.5.1. Subject Column - Named-Entity Column

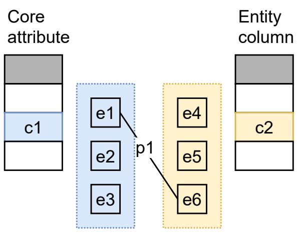


Fig. 5. Illustration of property candidate generation between the subject column and an entity column

Let $Pr(p|c^{\text{core}}_{j_1}, c^{ent}_{j_2})$ be the property candidates of the relation between the subject column $c^{\text{core}}_{j_1}$ and a named-entity column $c^{ent}_{j_2}$. A table cell of the subject column $c^{\text{core}}_{j_1}$ and row r_i is S_{i,j_1}. A table cell in the same row of a named-entity column $c^{ent}_{j_2}$ is S_{i,j_2}. We assume that there is a relation between entity candidates of S_{i,j_1} and S_{i,j_2} that equivalence to a DBpedia property; therefore, we query how many links (relations or properties) between entity candidates of S_{i,j_1} and S_{i,j_2}. The confidence scores of property candidates are calculated as $\text{score}^{c_{i,j_1}, c_{i,j_2}}_p = 1$ if there is any relation between entity candidates of two columns. We aggregate the scores of all rows of the two columns to get the property candidate scores as the following equation.

$$Pr(p|c^{\text{core}}_{j_1}, c^{ent}_{j_2}) = \sum_{i \in [1,N]} \text{score}^{c_{i,j_1}, c_{i,j_2}}_p \tag{16}$$

Then, we normalize the scores to a range of [0,1] by dividing each element with the total sum and associate the scores as the probabilities of property candidates between the subject column and a named-entity column.

Fig. 5 illustrates property candidate generation between the subject column and an entity column. The list of e_1, e_2, e_3 is the entity candidates of the table cell c_1, while e_4, e_5, e_6 are the entity candidates of cell c_2. The property candidate p_1 is derived from finding properties between the two lists of entity candidates. The final property candidates are aggregated from all rows of the two columns.

3.5.2. Subject Column - Literal Column

Let $Pr(p|c^{\text{core}}_{j_1}, c^{lit}_{j_2})$ be the property candidates of the relation between the subject column $c^{\text{core}}_{j_1}$ and a literal column $c^{lit}_{j_2}$. A cell of the subject column is S_{i,j_1}, and the cell of the literal column is S_{i,j_2}. We perform value-based matching to calculate the similarities between entity attribute values of the S_{i,j_1} entity candidates and the cell value S_{i,j_2}. Given an entity candidate e of S_{i,j_1} that has pairs of property(p_e)-value(v_e), we compare the similarity between S_{i,j_2} with all pair values v_e based on their data types (textual type or numerical type). We select the pairs of table cell values and entity values that have their similarities larger than β. The similarities are calculated as the following:

- Textual values: We use the inverse of Damerau–Levenshtein distance as the similarity between v_e and S_{i,j_2} as $\text{score}(v_e, S_{i,j_2})$ as the following equation.

$$\text{score}(v_e, S_{i,j_2}) = \frac{1}{s_{\text{edit}}(v_e, S_{i,j_2}) + 1} \tag{17}$$

  where $s_{\text{edit}}(v_e, S_{i,j_2})$ is the Damerau–Levenshtein distance.
- Numerical values: we adapt the relative change as the numerical similarity as the following equation.

$$\text{score}(v_e, S_{i,j_2}) = \begin{cases} 1 - \frac{|S_{i,j_2} - v_e|}{max(|S_{i,j_2}|, |v_e|)}, \\ \quad \text{if } max(|S_{i,j_2}|, |v_e|) \neq 0 \\ 1, \text{if } max(|S_{i,j_2}|, |v_e|) = 0 \\ \quad \text{and } |S_{i,j_2} - v_e| = 0 \end{cases} \tag{18}$$

We aggregate all relevant ratios based on properties, normalize the similarities to [0,1], and associate this with property candidates' probabilities.

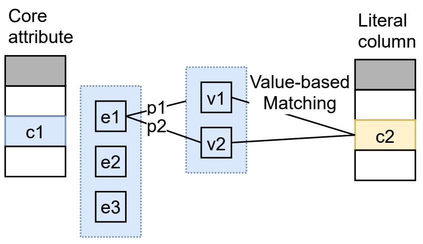


Fig. 6. Illustration of property candidate generation between the subject column and a literal column

Fig. 6 illustrates property candidate generation between the subject column and a literal column. The list of e_1, e_2, e_3 is the entity candidates of the table cell c_1, the pairs of property-value $[p_1, v_1], [p_2, v_2]$ are the triples of the entity candidate e_1. We calculate the similarities between the entity attribute values v_1, v_2 and the literal cell $c2$ and aggregate the similarity based on the properties of the entity values.

3.6. Step 5: Entity Matching

This step is to re-calculate the entity candidates based on the prior signals from the previous steps. Given a table cell $S_{i,j}$, we consider the signals from:

- $Pr(e|E_{S_{i,j}})$: The entity candidate probabilities from entity search as described in Section 3.3.
- $Pr(e|c_j)$: The probabilities of entity candidates given their type's probabilities are as in Section 3.4. We use different equations depend whether the column is the subject column or not. It could be calculated as the following equations.

$$Pr(e|c_j) = \begin{cases} Pr(t_e|c_j \text{ is a NE and not a subject column}) \\ Pr(t_e|c_j \text{ is a NE and a subject column}) \end{cases} \tag{19}$$

  where t_e is a type of the entity e.
- $Pr(e|S_{i,j})$: We calculate the similarity between the entity candidate label and the table cell using the normalized Damerau–Levenshtein distance. Then, we associate the score as the probabilities of entity candidates given the cell value $S_{i,j}$.
- $Pr(e|r_i)$: This signals are calculated from the probabilities of entity candidates in the subject column given cell values in a row r_i. We do the same procedure of value-based matching as Step 4 to compare all entity attribute values with row values but similarities are ordered by the entities. Then, we compute the mean probability for all cell values.

The entity candidate probability of the table cell of $S_{i,j}$ is calculated as the following equation:

$$Pr(e|S_{i,j}) = \begin{cases} w_5 \cdot Pr(e|E_{S_{i,j}}) + w_6 \cdot Pr(e|c_j) \\ +w_7 \cdot Pr(e|S_{i,j}) + w_8 \cdot Pr(e|r_i), \text{ if } S_{i,j} \in c_j^{\text{core}} \\ \\ w_5 \cdot Pr(e|E_{S_{i,j}}) + w_6 \cdot Pr(e|c_j) \\ +w_7 \cdot Pr(e|S_{i,j}), \text{ if } S_{i,j} \notin c_j^{\text{core}} \end{cases} \tag{20}$$

where w_5, w_6, w_7, w_8 are parameters. We also use a equal contribution for the parameters as $w_5 = w_6 = w_7 = w_8 = 1$. We select the highest probability of entity candidates as the annotation for a table cell.

3.7. Step 6, 7: Type and Property Matching

We aggregate the highest probabilities of entity candidates in Step 5 for each cell $S_{i,j}$, then infer types and properties with the majority voting.

4. MTab4D APIs, and Graphical Interface

This section describes our implementations as described in Section 3: MTab4D APIs and MTab4D graphical interface.

4.1. MTab4D APIs

We provide the five APIs as the following.

- Entity Search: The user could use this API to search relevant entities from the October 2016 version of DBpedia. There are three search modules (Section 3.3), such as keyword search, fuzzy search, and aggregation search.
- Entity Information Retrieval: The user could retrieve entity information from the October 2016 version of DBpedia. The responded object include DBpedia title, mapping to Wikidata, Wikipedia, label, aliases, types, entity popularity (PageRank score), entity triples, and literal triples.
- Table Annotation: The user could send a table to the API and get the annotation results including structural, and semantic annotations. The user could provide the target of annotations for CEA, CTA, or CPA tasks. However, MTab4D also could automatically predict the targets based on data types. The CEA targets are the table cells whose data types are strings. The CTA targets are columns so that the column data types are strings. The CPA targets are the relation between the core attribute and the remaining table columns.
- Numerical Labeling: The user could do numerical labeling from numerical columns and get a ranking list of relevant properties as EmbNum+ [4].
- SemTab 2019 Evaluation: The user could submit the annotation results of CEA, CTA, and CPA tasks to calculate the evaluation metrics from the original and adapted SemTab 2019.

4.2. MTab4D Graphical Interface

We provide two interfaces as entity search and table annotation.

4.2.1. Entity Search Interface

The user can enter a query in the entity search interface then search with the three MTab4D entity search modules (e.g., keyword search, fuzzy search, and aggregation search). Figure 7 illustrates an example of the fuzzy search with the keyword of “Senaticweb". It takes only 0.06 second to get the relevant “Semantic Web" entity.

4.2.2. Table Annotation Interface

The user can copy and paste table content expressed in any language from tabular data files (Excel, CSV, TSV) or tables on the Web in the table annotation interface. Then, the user could tap the “Annotate" button to get the annotation results.

Figure 8 illustrates an example of table annotation on the “v15_1" table in Round 4 of SemTab 2019. MTab4D takes 0.78 seconds to annotate the input table as shown in the left figure. The figure on the right is the annotation results. The table header is in the first row, and the core attribute is in the first column. Entity annotations are in red and located below the table cell value. The type annotation is in green and located in the “Type” column. Finally, the relations between the core attribute and other columns are in blue and located in the property column.

5. MTab4D vs MTab [7]

MTab4D is an extended version of our work MTab [7] (the best performance for the three matching tasks: the 1st rank in all (four) rounds and all (three) tasks). This system advances the previous study in the three directions:

- Refactoring Implementation: We refactor the codes over most of the components of MTab [7] to optimize MTab4D efficiency. MTab4D is publicly available at[4] under an open source license.
- Reproducibility: MTab’s entity search modules are built by aggregated many online entity search services from DBpedia, Wikipedia, and Wikidata [7]. As a result, it is hard to reproduce the entity search results since the search index changes over time. We build new search modules based on the dump of October 2016 version of DBpedia to enable reproducibility. Moreover, we also provide the new adapted SemTab 2019 data with the DBpedia October 2016 version [6]. These resources enable a consistent environment setup for a fair comparison between annotation systems the future studies.
- Public services: We also focus on building public services so that we refactor the implementation, optimize system efficiency, and support multilingual tables, and could be able to process various table formats such as Excel, CSV, TSV, or markdown tables. We also provide a graphical interface that enables the user to do table annotation by pasting the table of content from table files or web site. According to Wang et al. [5], they only could generate the annotations using our system, while others require high time complexity.

6. Evaluation

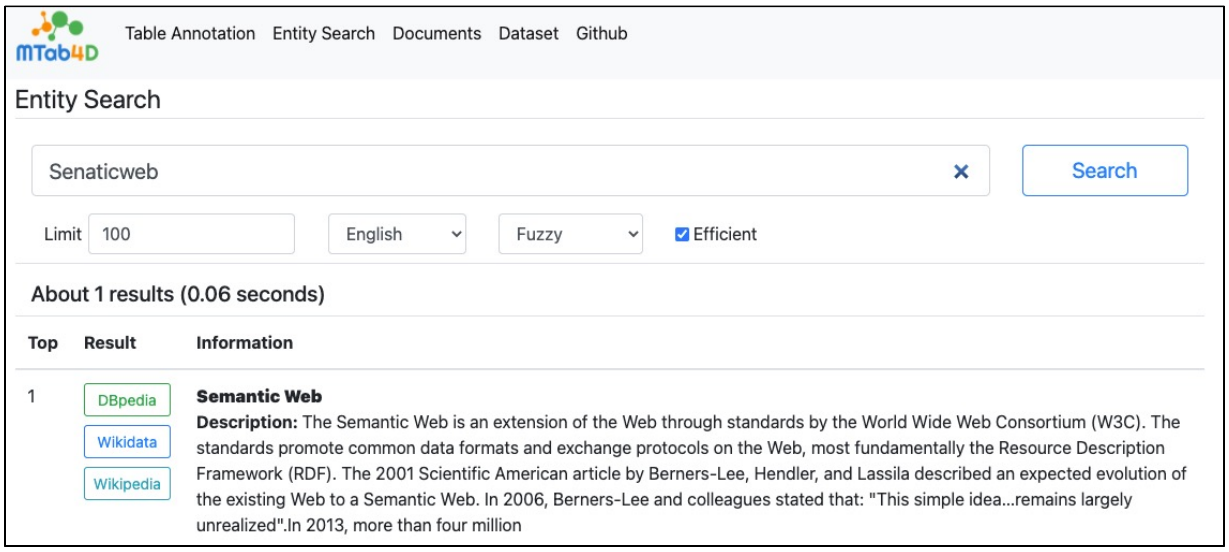
In this section, we first report the detail about benchmark datasets in Section 6.1, evaluation metrics in Section 6.3, and experimental setting in Section 6.4. The overall results are reported in Section 6.5.

6.1. Datasets

We use the two datasets as the original SemTab 2019 (four rounds) and the adapted SemTab 2019 with the 2016 October of DBpedia [6]. Table 2 reports the statistic about the two datasets. The numbers shown are the number of tables in each dataset; target cells (in CEA), columns (in CTA), and column pairs (in CPA) of the original and adapted versions of the SemTab 2019 dataset.

6.1.1. Original SemTab 2019 Dataset

The SemTab 2019 challenge has four rounds; each round came with a different set of tables and the targets of matching for each annotation task [1]. In detail, round 1 data is a subset of the T2Dv2 dataset, a standard dataset in tabular data annotation. Round 2 is the biggest and most complex one since it combines Wikipedia tables and automatically generated tables from DBpedia. Round 3 and round 4 datasets also are automatically generated from DBpedia, but the easily matched cells are removed in Round 4. To generate the tabular data, firstly, a list of classes and properties are gathered, then for each class, the generator selects groups of properties and using them to create "realistic" tables using SPARQL queries. Finally, the "realistic" tables are added noise into the surface textual of table cells or remove "easy" matches cells.



Entity Search

Senaticweb

Limit 100 | English | Fuzzy | ☑ Efficient

About 1 results (0.06 seconds)

Top	Result	Information
1	DBpedia Wikidata Wikipedia	**Semantic Web** **Description:** The Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF). The 2001 Scientific American article by Berners-Lee, Hendler, and Lassila described an expected evolution of the existing Web to a Semantic Web. In 2006, Berners-Lee and colleagues stated that: "This simple idea...remains largely unrealized".In 2013, more than four million

Fig. 7. MTab4D entity search interface

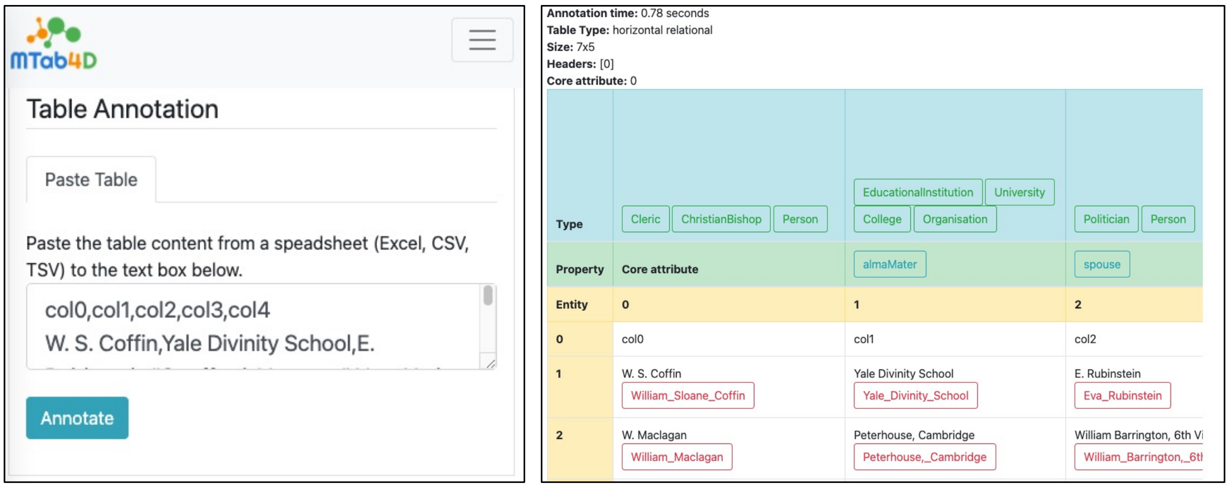


Table Annotation

Paste the table content from a speadsheet (Excel, CSV, TSV) to the text box below.

```
col0,col1,col2,col3,col4
W. S. Coffin,Yale Divinity School,E.
```

Annotation time: 0.78 seconds
Table Type: horizontal relational
Size: 7x5
Headers: [0]
Core attribute: 0

Type	Cleric ChristianBishop Person	EducationalInstitution University College Organisation	Politician Person
Property	**Core attribute**	almaMater	spouse
Entity	**0**	**1**	**2**
0	col0	col1	col2
1	W. S. Coffin William_Sloane_Coffin	Yale Divinity School Yale_Divinity_School	E. Rubinstein Eva_Rubinstein
2	W. Maclagan William_Maclagan	Peterhouse, Cambridge Peterhouse,_Cambridge	William Barrington, 6th Vi William_Barrington,_6tl

Fig. 8. MTab4D table annotation interface

Table 2
Statistics on the original and adapted SemTab 2019 datasets

Round	#Table	CEA		CPA		CTA	
		Orginal	Adapted	Orginal	Adapted	Orginal	Adapted
1	70	8418	8406	116	116	120	120
2	11925	463796	457567	6762	6762	14780	14333
3	2162	406827	406820	7575	7575	5762	5673
4	818	107352	107351	2747	2747	1732	1717

6.2. *Adapted SemTab 2019 Dataset*

This section presents an adapted SemTab 2019 with the 2016 October of DBpedia for reproducibility [6]. It is challenging to compare annotation systems while they do not use the same experimental setting (the same version of DBpedia). Knowledge Graphs change over time so that the schema or instances from a DBpedia version have many differences from another version. As a result, a annotation system could yield a difference performance when it is benchmarked on difference DBpedia version.

To enable reproducibility, we perform many adaptations on the original version of SemTab 2019 dataset to the October 2019 of DBpedia (Section 6.2.1), as well as introduce many open resources (Section 6.2.2).

6.2.1. Ground Truth

We process the original SemTab 2019 dataset as follows:

– We make the matching targets and ground truth answers consistent by removing the matching targets that are not available in the ground truth and the original matching targets.
– We remove invalid entities, types, and properties that are not available in the October 2016 version of DBpedia.
– We add missing redirect, equivalent entities, types, and properties.
– We remove prefix to avoid redirect issues. For example, the expected prefix of the entity “Lake Alan Henry" is “http://dbpedia.org/**resource**/Lake_Alan_Henry" while the CEA ground truth of the original SemTab 2019 Round 1 has “http://dbpedia.org/**page**/Lake_Alan_Henry". Removing the prefix also have a data storage-efficient (The adapted SemTab 2019 saves 51.3% space than the original dataset).

6.2.2. Public Resources

We also prepare open resources to enable them to be reproduced for future studies.

– Schema: We prepare the CSV files as DBpedia class hierarchy, properties, and their equivalents.
– Data: We also published an entity JSON list dump (all information about entities) of the October 2016 version of DBpedia. An user also can quickly get the information of each entity using our opened API[4] without process entire all DBpedia entities.
– Entity Search: We provide a public API of entity search based on entity label and aliases (multilingual) of DBpedia 2016 October. The search results will be expected to be the same using our API while using other online entity searches (e.g., DBpedia entity search or Wikipedia search) could yield different answers.
– Other resources: We also provide other public APIs[4] of tabular data annotations, numerical attribute labeling, annotation evaluation for the original and adapted SemTab 2019 datasets [6].

6.2.3. Analysis on the Original SemTab 2019 Dataset

CEA Task: Table 3 depicts the number of inconsistencies between the SemTab 2019 ground truth data and the October 2016 version of DBpedia.

Index Inconsistencies (IIndex) describes the number of invalid table cell indexes of CEA targets. Encoding Inconsistencies (IEncoding) describes the number of encoding errors of DBpedia URIs. Many samples are inconsistent with URI encoded and decoded representation. For example, an entity URI of dbr:Angélica_Rivera could be encoded as "dbr:Ang%C3%A9lica_Rivera" and decoded as "dbr:Angélica_Rivera". The ground truth of CEA contains a mixture between encoded URI and decoded URI. The encoding URI (percent-encoding) is not encouraged[8]. Invalid Inconsistencies (IInvalid) is the number of invalid entities that are not in the October 2016 version of DBpedia, and Redirect Inconsistencies (IRedirect) is the number of targets missing redirect entities.

The Round 2 dataset contains many inconsistencies, including the four types of inconsistencies (7.8% inconsistencies) because this dataset combine a subset of Wikipedia tables (with a different KG target: the October 2015 version of DBpedia) and automatically generated tables (from the October 2016 version of DBpedia). Round 1 dataset is the second place of inconsistencies (7%) since this is the subset of T2D dataset (with a KG target as the 2014 version of DBpedia). Round 3 dataset has 3% inconsistencies, and Round 4 dataset is the cleanest in four rounds (2% inconsistencies).

Table 3
Analysis on the CEA task of SemTab 2019 with the October 2016 version of DBpedia

Round	IIndex	IEncoding	IInvalid	IRedirect
1	0	418	22	203
2	250	21912	7662	6130
3	0	7427	763	4140
4	0	1487	22	682

CTA Task: Table 4 depicts the number of inconsistencies of CTA with the October 2016 version of DBpedia. Index errors (IIndex) describes the number of invalid table column indexes of CTA targets. Missing equivalent classes (IHierarchy) is the number of inconsistencies of hierarchy classes in the CTA ground truth. There is

[8]DBpedia URI encoding: https://wiki.dbpedia.org/uri-encoding

2% index errors in the round 2 dataset. Although, the CTA ground truth is derived from the October 2016 version of DBpedia, there are some inconsistencies such as dbo:Region class is not a ancestor of dbo:City in the October 2016 version of DBpedia while it is ancestor in the ground truth data. The class hierarchy also misses equivalent classes such as dbo:PenaltyShootOut class is the equivalent with dbo:Event. It could be semantic incorrect about the fact of <dbo:PenaltyShootOut owl:equivalentClass dbo:Event>, however we adopt the fact to follow DBpedia correctness and completeness assumption.

Round 2 dataset has 2% index errors where the target matching is not available in the input table. Round 1 has no errors, while round 2, round 3, round 4 have 5%, 6%, 1% IHierarchy error rate, respectively.

Table 4
Analysis on CTA task of SemTab 2019 with the October 2016 version of DBpedia

Round	IIndex	IHierarchy
1	0	0
2	323	697
3	0	353
4	0	14

CPA Task: We found that the ground truth of CPA task miss many equivalent properties. For example, the properties of dbo:team has its equivalent property as dbo:club. Some of the equivalent properties in the October 2016 version of DBpedia are (dbo:team, dbo:club), (dbo:composer, dbo:musicBy, dbo:jureLanguage), and (dbo:area, dbo:landArea, dbo:waterArea).

Table 4 depicts statistic on missing equivalent properties (IIEquivalent) the October 2016 version of DBpedia. Round 1 has 4% of ground truth missing equivalent properties. Round 2, 3, 4 have approximate of 28% missing equivalent properties.

Table 5
Analysis on CPA task of SemTab 2019 with the October 2016 version of DBpedia

Round	IEquivalent
1	5
2	1911
3	2030
4	828

6.3. Evaluation Metrics

There are four different metrics used to evaluate tabular data annotation:

F1-score is a harmonic mean of precision and recall. It is used as the primary score to measure the the performance of entity annotations (**CEA** - all rounds), property annotations (**CPA** - all rounds), and type annotation (**CTA** - round 1). The F1 metric is calculated as follows.

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{21}$$

where Precision, and Recall are calculated as follows.

$$\text{Precision} = \frac{\text{\# correct annotations}}{\text{\# annotations}} \tag{22}$$

$$\text{Recall} = \frac{\text{\# correct annotations}}{\text{\# target annotations}} \tag{23}$$

Precision scores is used as the secondary score in entity annotations (**CEA** - all rounds), property annotations (**CPA** - all rounds), and type annotation (**CTA** - round 1).

Regarding the type annotation **CTA** task, there are two metrics designed to measure the hierarchy of class annotations (Average Hierarchical - AH) and perfect class annotations (Average Perfect - AP). The AH score is used as the primary score, while the AP score is used as the secondary score for rounds 2, 3, 4 of **CTA** task.

Let the list of target columns be T, a column annotation be a, and the number of perfect annotations be p_a, the number of OK annotations denotes as o_a, and the number of the wrong annotation denotes as w_a. The equations of the AH score and AP score are described as follows.

$$\text{AH} = \frac{\sum_{a \in T} p_a + 0.5 * o_a - w_a}{|T|} \tag{24}$$

$$\text{AP} = \frac{\sum_{a \in T} p_a}{\sum_{a \in T} p_a + o_a + w_a} \tag{25}$$

6.4. Experimental settings

MTab4D is built based on the October 2016 version of DBpedia with three versions (a, b, f) depending on the use of the entity search module. MTab4Db is the system that uses the keyword search, MTab4Df is used the fuzzy search, and MTab4Da is used the aggregation search.

We compare MTab4D with the others reported results from SemTab 2019 dataset (original version). Unlike our participated system MTab, MTab4D focuses on reproducibility, where we use the entity search built from dump data of DBpedia.

We also conduct experiments MTab4D on the adapted version of SemTab 2019, where we remove the inconsistencies between ground truth and the October 2016 version of DBpedia.

6.5. Experimental Results

In this section, we first report the results of MTab4D with other SemTab 2019 participants on the orginal data version in Section 6.5.1. The full results are reported in the challenge websites[9]. Then, we present the results of MTab4D on the adapted version of SemTab 2019 in Section 6.5.2.

6.5.1. Original SemTab 2019 Dataset

Table 6 depicts the CEA results in the F1 score and Precision of MTab4D compared to the other systems on the original version of SemTab 2019. Because of the high data inconsistencies in round 1 and round 2, MTab4D could not provide comparable results with the original system MTab. However, MTab4Db, with a keyword search, got slightly higher performance than the MTab system in Round 3 and Round 4. In Round 1 and Round 2, MTab4Df using fuzzy search achieves higher performance than MTab4D using keyword search and aggregation search. It could be explained that the datasets have a higher noisy level of table cells, such as incorrect encoding parsing, entity labels variance, or abbreviation. In rounds 3 and 4, MTab4Db using keyword search achieves higher performance since the table cells are more likely similar to entity labels of DBpedia.

Table 7 depicts the CTA results in AH score and AP score of MTab4D compared to the other systems on the original version of SemTab 2019. Because of the CTA inconsistencies of the ground truth, MTab4D results are not comparable with our system MTab in SemTab 2019. MTab4Df achieves slightly higher performance than MTab4D using other entity search modules.

Table 8 depicts the CPA results in F1 score and precision of MTab4D compared to the other systems on the original version of SemTab 2019. MTab4D results are slightly lower than our system MTab in SemTab 2019 because of lacking equivalent properties of the CPA ground truth. The results of using different search modules are similar; as a result, we conclude that there is no effect of using different entity search modules in MTab4D.

6.5.2. Adapted SemTab 2019 Dataset

This section reports a comparison of the MTab4D performance on the original and adapted versions of the SemTab 2019 datasets.

Table 9 depicts the MTab4D results in the F1 score of the CEA task on the original and adapted version of the SemTab 2019 dataset. MTab4D is built on the October 2016 version of DBpedia consistently achieve better performance on the adapted version dataset than the original one. Round 1, 2 results in the adapted version have more improvement than Round 3, 4 because Round 1, 2 have more inconsistencies in the original dataset.

Table 11 depicts the MTab4D results in the F1 score of the CPA task on the original and adapted version of the SemTab 2019 dataset. The performance of MTab significantly improves on the adapted version, adding the equivalent properties into the ground truth data.

Due to the incompleteness of DBpedia, there are many indirect equivalent properties in DBpedia. For example, dbo:deathCause and dbo:causeOfDeath have the same equivalent property of wikidata:P509 (cause of death). The problem of knowledge graph completion is not the main focus of this work, but we can expect the improvement of property annotations when the completeness of DBpedia is improved.

Table 11 depicts the MTab4D results in the AH score of the CTA task on the original and adapted version of the SemTab 2019 dataset. The performance of MTab consistently improves on the adapted version.

7. Related Work

In this section, we review the other systems participating in SemTab 2019. Also, we discuss related works on the tabular data annotation task.

[9]Results: http://www.cs.ox.ac.uk/isg/challenges/sem-tab/2019/results.html

Table 6
CEA results in F1 score and Precision for the four rounds of the original version of SemTab 2019

	F1 score				Precision			
Round	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
MTab*	**1.000**	**0.911**	0.97	0.983	**1.00**	**0.911**	0.97	0.983
CSV2KG*	0.448	0.883	0.962	0.907	0.627	0.893	0.964	0.912
Tabularisi*	0.884	0.826	0.857	0.803	0.908	0.852	0.866	0.813
MantisTable*	1.000	0.614	0.633	0.973	1.00	0.673	0.679	0.983
LOD4ALL*	0.852	0.757	0.828	0.648	0.874	0.767	0.833	0.654
ADOG*	0.657	0.742	0.912	0.835	0.673	0.745	0.913	0.838
DAGOBAH*	0.897	0.713	0.725	0.578	0.941	0.816	0.745	0.599
MTab4Db	0.839	0.887	**0.984**	**0.984**	0.839	0.888	**0.984**	**0.984**
MTab4Df	0.867	0.892	0.983	0.983	0.873	0.9	0.983	0.983
MTab4Da	0.839	0.885	0.983	0.983	0.839	0.886	0.983	0.983

* Results are taken from SemTab 2019 [11]

Table 7
CTA results in F1 and Precision for the round 1 and AH score, and AP score of the original version of SemTab 2019

	AH score				AP score			
Round	**1(F1)**	**2**	**3**	**4**	**1(Precision)**	**2**	**3**	**4**
MTab*	**1.000**	**1.414**	**1.956**	**2.012**	**1.000**	**0.276**	**0.261**	**0.300**
CSV2KG*	0.833	1.376	1.864	1.846	0.833	0.257	0.247	0.274
Tabularisi*	0.825	1.099	1.702	1.716	0.825	0.261	0.277	0.325
MantisTable*	0.929	1.049	1.648	1.682	0.933	0.247	0.269	0.322
LOD4ALL*	0.850	0.893	1.442	1.071	0.850	0.234	0.260	0.386
ADOG*	0.829	0.713	1.409	1.538	0.851	0.208	0.238	0.296
DAGOBAH*	0.644	0.641	0.745	0.684	0.580	0.247	0.161	0.206
MTab4Db	-	0.947	1.837	1.922	-	0.216	0.247	0.289
MTab4Df	-	0.994	1.839	1.927	-	0.225	0.249	0.289
MTab4Da	-	0.962	1.838	1.922	-	0.217	0.247	0.289

* Results are taken from SemTab 2019 [11]

Table 8
CPA results in F1 score and Precision for the four rounds of the original version of SemTab 2019

	F1 score				Precision			
Round	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
MTab*	**0.987**	**0.881**	**0.844**	**0.832**	**0.975**	**0.929**	**0.845**	**0.832**
CSV2KG*	-	0.877	0.841	0.830	-	0.926	0.843	0.835
Tabularisi*	0.606	0.79	0.827	0.823	0.638	0.792	0.83	0.825
MantisTable*	0.965	0.46	0.518	0.787	0.991	0.544	0.595	0.841
LOD4ALL*	-	0.555	0.545	0.439	-	0.941	0.853	0.904
ADOG*	-	0.459	0.558	0.75	-	0.708	0.763	0.767
DAGOBAH*	0.415	0.533	0.519	0.398	0.347	0.919	0.826	0.874
MTab4Db	-	0.838	0.842	0.829	-	0.841	**0.847**	**0.837**
MTab4Df	-	0.838	0.842	0.829	-	0.841	**0.847**	**0.837**
MTab4Da	-	0.838	0.842	0.829	-	0.841	**0.847**	**0.837**

* Results are taken from SemTab 2019 [11]

7.1. SemTab 2019 Systems

This section describes the six other systems frequently participants for all rounds of SemTab 2019 challenges.

Table 9

CEA Results in F1 score of MTab4D for the original and adapted version of SemTab 2019 dataset

	Round 1		Round 2		Round 3		Round 4	
Method	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**
MTab4Db	0.839	0.856 (+2.03%)	0.887	0.918 (+3.49%)	0.984	0.992 (+0.81%)	0.984	0.988 (+0.41%)
MTab4Df	0.867	0.886 (+2.19%)	0.892	0.924 (+3.59%)	0.983	0.992 (+0.92%)	0.983	0.987 (+0.41%)
MTab4Da	0.839	0.859 (+2.38%)	0.885	0.916 (+3.50%)	0.983	0.992 (+0.92%)	0.983	0.986 (+0.31%)

Table 10

CPA Results in F1 score of MTab4D for the original and adapted version of SemTab 2019 dataset

	Round 1		Round 2		Round 3		Round 4	
Method	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**
MTab4Db	-	-	0.838	0.981 (+17.06%)	0.842	0.973 (+15.56%)	0.829	0.98 (+18.21%)
MTab4Df	-	-	0.838	0.981 (+17.06%)	0.842	0.973 (+15.56%)	0.829	0.98 (+18.21%)
MTab4Da	-	-	0.838	0.981 (+17.06%)	0.842	0.973 (+15.56%)	0.829	0.98 (+18.21%)

Table 11

CTA Results in AH score of MTab4D for the original and adapted version of SemTab 2019 dataset

	Round 1		Round 2		Round 3		Round 4	
Method	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**	**Original**	**Adapted**
MTab4Db	1.292	1.35 (+4.49%)	0.947	0.98 (+3.48%)	1.837	1.926 (+4.84%)	1.922	1.923 (+0.05%)
MTab4Df	1.337	1.396 (+4.41%)	0.994	1.028 (+3.42%)	1.839	1.928 (+4.84%)	1.927	1.926 (-0.05%)
MTab4Da	1.3	1.358 (+4.46%)	0.962	0.998 (+3.74%)	1.838	1.927 (+4.84%)	1.922	1.923 (+0.05%)

Table 12

Comparison of entity candidate generation methods of SemTab 2019 participants

	MTab4D	CSV2KG	TabularISI	MantisTable	LOD4ALL	ADOG	DAGOBAH
URI heuristic*	x	✓	x	x	✓	x	x
DBpedia SPARQL	✓	x	x	✓	x	x	✓
DBpedia Lookup	✓	✓	x	x	x	x	x
DBpedia Spotlight	x	✓	x	x	x	x	x
Wikidata SPARQL	✓	x	✓	x	x	x	✓
Wikipedia (CirrusSearch)	x	x	x	x	x	x	✓
Wikipedia (Multilingual)	✓	x	x	x	x	x	x
DBpedia Elastic Search	x	x	✓	x	x	✓	x
Wikidata Elastic Search	x	x	✓	x	x	x	✓
LOD4ALL Elastic Search	x	x	x	x	✓	x	x

The participants generate entity candidates by looking up table cell values or search values in the local index with Elastic Search in DBpedia, Wikidata. Table 12 reports the lookup services used in the participant systems because of lacking specification of the use of information retrieval techniques, hyper-parameters, database index sources. Then, the type candidates and property candidates are estimated using the entity candidates. Finally, the systems perform entity disambiguation to return the CEA results. The CTA, and CPA annotations are generated with the CEA annotations using majority voting.

CSV2KG (IDLAB) first searches on DBpedia lookup and DBpedia Spotlight to generate entity candidates [12]. The type candidates and property annotations are estimated using majority voting approaches based on entity candidates. Then, the entity annotations are estimated using the information of property candidates. Finally, type annotations are estimated using entity annotations.

Tabular ISI approach first generates entity candidates with Wikidata API and Elastic Search on entity labels of Wikidata, DBpedia. Second, the authors use the heuristics TF-IDF approach and machine learning (neural network ranking) model to select the best candidate for the entity annotation task [13]. The type annotations are estimated with the results from entity annotations with hierarchy searching on common classes. The property annotations are estimated by finding the relation between entity candidates of the primary and secondary columns or values matching the primary and secondary columns' values.

Mantis Table performs column analysis, including predicting name entity columns, literal columns, and subject column, then mapping between columns into concepts in DBpedia [14][10]. The relationships between the subject column and other columns are estimated based on predicate context and predicate frequency of column value and candidate predicates. Finally, entity linking is performed using the results from previous steps for cell value disambiguation. The property annotations are estimated by getting the maximum frequency of property candidates in the entity linking phase. To estimate type annotations, the authors calculate the hierarchical path score of entity types from entity annotations. Then type annotations are estimated on the maximum of the path score.

DAGOBAH performs entity linking with a lookup on Wikidata and DBpedia as well as voting mechanisms [15]. The authors used Wikidata entity embedding to estimate the entity type candidates, assuming that the same column's entities should be closed in the embedding spaces as they share semantic meanings.

LOD4ALL uses a combination of direct search (SPARQL ASK on dbr:"query"), keyword search (Abbreviation of Human name), and Elastic Search to find entity candidates [16]. The entity candidates will be used to estimate type annotations with majority voting. Then, the system determines the entity annotations with the type constraints. Finally, the property annotations are estimated by a column-based majority voting with entity annotations of each table row.

ADOG focuses on entity annotation with Elastic Search on an integrated ontology (DBpedia sub-graph) using a NoSQL database named ArangoDB [17]. The system estimates entity annotation using Levenshtein distance, and the results of type and property annotations are estimated from entity annotations.

In summary, some participants adopt the online lookup services of DBpedia, Wikidata, Wikipedia. As a result, it is hard to reproduce the experimental result while the source index changes over time. Some participants built offline lookup services but lacked specification on the information retrieval techniques, hyper-parameter settings, or index sources. It is also hard to reproduce their results. In MTab4D, we address reproducibility problems by building a database index from a dump data of the October 2016 version of DBpedia and releasing many public APIs about entity search modules that enable reproducibility for future study.

[10]Web interface: http://zoo.disco.unimib.it/mantistable/

Moreover, tabular data contains many numerical attributes that help us use semantic labeling results for numerical attributes. In MTab4D, we aggregate signal from the result of semantic labeling for numerical attributes (columns) using EmbNum+ [4] (deep metric for distribution similarity calculation). Additionally, we also use novel signals from the relations of column pairs to enhance overall matching performance.

7.2. Other Tabular Data Annotation Tasks

The tabular data annotation tasks could be categorized as structure or semantic annotation.

The structural annotation contains many tasks as table type prediction [18], data type prediction, table header annotation, subject column prediction, and holistic matching across tables [19]. In SemTab 2019, most tables are represented as a horizontal relational type; headers are located at the first row of tables, and the subject column is in the first table column.

There are many previous studies on table semantic annotation, including schema-level matching, e.g., tables to classes [3], columns to properties [3, 4, 20, 21] or classes [22], and data-level matching, e.g., rows [3, 23] or cells to entities [2, 22]. SemTab 2019 also has schema annotation as the CTA task, data annotation as the CEA task, and a novel CPA task as column relation annotation.

7.3. DBpedia Version

Due to the different environmental settings, such as the DBpedia version, it is hard to compare the annotations directly. Table 13 reports the DBpedia versions used in table annotation tasks. Quercini et al. [24] use a snippet of DBpedia in 2013. T2K [3] conducts experiments on the T2D dataset built on the 2014 version of DBpedia. Efthymiou et al. [23] introduce Wikipedia tables, and an adapted version of Limaye gold standard [2] built on the October 2015 version of DBpedia. The recent work (MantisTable [25]) builds the annotation system based on the 2017 version of DBpedia. In this work, we follow the SemTab 2019, to build the system based on the October 2016 version of DBpedia.

Table 13
Studies used DBpedia as the target knowledge graph

Study	DBpedia Version
Quercini et al. [24]	2013
T2K [3]	2014
Efthymiou et al. [23]	October 2015
MantisTable [25]	2017
MTab4D	October 2016

8. Conclusion

This paper presents MTab4D, a table annotation system that combines multiple matching signals from different table elements to address schema heterogeneity, data ambiguity, and noisiness. This paper also provides insightful analysis and extra resources on benchmarking semantic annotation with knowledge graphs. Additionally, we also introduce MTab4D APIs and graphical interfaces for reproducibility. Experimental results on the original and adapted datasets of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2019) show that our system achieves an impressive performance for the three matching tasks.

8.1. Future Work

MTab4D could be improved in many dimensions, such as effectiveness, efficiency, and generality. Regarding efficiency, MTab4D could be modified in a parallel processing fashion since the lookup steps, and the probability estimations in Step 2, 3, and 4 are independent. Regarding effectiveness, MTab4D performance could be improved by relaxing our assumptions:

- Assumption 1: The closed-world assumption might not hold in practice. Improving the completeness and correctness of knowledge graphs might improve MTab4D performance.
- Assumption 2: Classifying table types before matching could help to improve MTab4D performance.
- Assumption 3: In reality, some tables could have a shared schema. For example, tables on the Web could be divided into many web pages; therefore, we can expect an improving matching performance by stitching tables on the same web page (or domain) [19], [26]. Therefore, performing holistic matching could help improve MTab4D performance.

8.2. Lessons Learned

This session discusses the lessons learned from SemTab 2019 challenge

Benchmarking Value From our perspective, SemTab 2019 plays a vital role in benchmarking tabular data annotation tasks. Due to the differences in benchmark settings, tabular datasets, and target matching knowledge bases in the literature, there is a need for a general benchmark for tabular data annotation tasks to promote a fair comparison of annotation systems. This challenge reflects the practical performance of matching techniques and the importance of features for tabular matching.

DBpedia as the Target Knowledge Graph The choice of DBpedia as the target matching reflects the low update knowledge graph. In real-world practice, many knowledge graphs change rapidly, such as Wikidata. We will have a different set of challenges in matching the fast-evolving knowledge graphs.

Acknowledgements

The research was partially supported by the Cross-ministerial Strategic Innovation Promotion Program (SIP) Second Phase, "Big-data and AI-enabled Cyberspace Technologies" by the New Energy and Industrial Technology Development Organization (NEDO).

We would like to thank the SemTab 2019 challenge organizers for their organizing the successful challenge. We also thank IBM Research and SIRIUS for their sponsorship of the challenge.

References

[1] O. Hassanzadeh, V. Efthymiou, J. Chen, E. Jiménez-Ruiz and K. Srinivas, SemTab2019: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching - 2019 Data Sets, Zenodo, 2019. doi:10.5281/zenodo.3518539.

[2] G. Limaye, S. Sarawagi and S. Chakrabarti, Annotating and Searching Web Tables Using Entities, Types and Relationships, *PVLDB* **3**(1) (2010), 1338–1347. doi:10.14778/1920841.1921005. http://www.vldb.org/pvldb/vldb2010/pvldb_vol3/R118.pdf.

[3] D. Ritze, O. Lehmberg and C. Bizer, Matching HTML Tables to DBpedia, in: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS 2015, Larnaca, Cyprus, July 13-15, 2015*, 2015, pp. 10:1–10:6. doi:10.1145/2797115.2797118.

[4] P. Nguyen, K. Nguyen, R. Ichise and H. Takeda, EmbNum+: Effective, Efficient, and Robust Semantic Labeling for Numerical Values, *New Generation Computing* (2019). doi:10.1007/s00354-019-00076-w.

[5] D. Wang, P. Shiralkar, C. Lockard, B. Huang, X.L. Dong and M. Jiang, TCN: Table Convolutional Network for Web Table Interpretation, in: *WWW 2021*, pp. 4020–4032.

[6] P. Nguyen, N. Kertkeidkachorn, R. Ichise and H. Takeda, Semantic Annotation for Tabular Data with DBpedia: Adapted SemTab 2019 with DBpedia 2016-10, Zenodo, 2021, This data is redistributed from - SemTab 2019: https://doi.org/10.5281/zenodo.3518530 - Wikipedia https://www.wikipedia.org/ - DBpedia http://dbpedia.org/ - T2Dv2 Gold Standard for Matching Web Tables to DBpedia http://webdatacommo ns.org/webtables/goldstandardV2.html Please refer to the licenses from these sources. doi:10.5281/zenodo.4922769.

[7] P. Nguyen, N. Kertkeidkachorn, R. Ichise and H. Takeda, MTab: Matching Tabular Data to Knowledge Graph using Probability Models, *CoRR* **abs/1910.00246** (2019). http://arxiv.org/abs/1910.00246.

[8] R. Speer, ftfy, 2019, Version 5.5. https://github.com/LuminosoInsight/python-ftfy.

[9] M. Honnibal and I. Montani, spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing, 2017, To appear. https://spacy.io/.

[10] W. Garbe, SymSpell: Symmetric Delete algorithm, GitHub, 2012.

[11] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen and K. Srinivas, Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: *The Semantic Web - 17th International Conference, ESWC 2020*, Lecture Notes in Computer Science, Vol. 12123, Springer, 2020, pp. 514–530. https://doi.org/10.1007/978-3-030-49461-2_30.

[12] G. Vandewiele, B. Steenwinckel, F.D. Turck and F. Ongenae, CVS2KG: Transforming Tabular Data into Semantic Knowledge, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/IDLab.pdf.

[13] A. Thawani, M. Hu, E. Hu, H. Zafar, N.T. Divvala, A. Singh, E. Qasemi, P. Szekely and J. Pujara, Entity Linking to Knowledge Graphs to Infer Column Types and Properties, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/Tabularisi.pdf.

[14] M. Cremaschi, R. Avogadro and D. Chieregato, MantisTable: an Automatic Approach for the Semantic Table Interpretation, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/MantisTable.pdf.

[15] Y. Chabot, T. Labbe, J. Liu and R. Troncy, DAGOBAH: An End-to-End Context-Free Tabular Data Semantic Annotation System, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/DAGOBAH.pdf.

[16] H. Morikawa, F. Nishino and N. Igata, Semantic Table Interpretation using LOD4ALL, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/LOD4ALL.pdf.

[17] D. Oliveira and M. d'Aquin, ADOG - Anotating Data with Ontologies and Graphs, 2019, Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, ISWC. http://www.cs.ox.ac.uk/isg/challenges/sem-tab/papers/ADOG.pdf.

[18] K. Nishida, K. Sadamitsu, R. Higashinaka and Y. Matsuo, Understanding the Semantic Structures of Tables with a Hybrid Deep Neural Network Architecture, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, 2017, pp. 168–174. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14396.

[19] O. Lehmberg and C. Bizer, Stitching Web Tables for Improving Matching Quality, *PVLDB* **10**(11) (2017), 1502–1513. doi:10.14778/3137628.3137657. http://www.vldb.org/pvldb/vol10/p1502-lehmberg.pdf.

[20] M. Pham, S. Alse, C.A. Knoblock and P.A. Szekely, Semantic Labeling: A Domain-Independent Approach, in: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*, 2016, pp. 446–462. doi:10.1007/978-3-319-46523-4_27.

[21] J. Chen, E. Jiménez-Ruiz, I. Horrocks and C.A. Sutton, ColNet: Embedding the Semantics of Web Tables for Column Type Prediction, in: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, January 27 - February 1, 2019, Honolulu, Hawaii, USA*, 2019, pp. 29–36. doi:10.1609/aaai.v33i01.330129.

[22] Z. Zhang, Effective and efficient Semantic Table Interpretation using TableMiner$^+$, *Semantic Web* **8**(6) (2017), 921–957. doi:10.3233/SW-160242.

[23] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro and V. Christophides, Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings, in: *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, 2017, pp. 260–277. doi:10.1007/978-3-319-68288-4_16.

[24] G. Quercini and C. Reynaud, Entity discovery and annotation in tables, in: *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, G. Guerrini and N.W. Paton, eds, ACM, 2013, pp. 693–704. doi:10.1145/2452376.2452457.

[25] M. Cremaschi, F.D. Paoli, A. Rula and B. Spahiu, A fully automated approach to a complete Semantic Table Interpretation, *Future Gener. Comput. Syst.* **112** (2020), 478–500. https://doi.org/10.1016/j.future.2020.05.019.

[26] D. Ritze, Web-Scale Web Table to Knowledge Base Matching, PhD thesis, University of Mannheim, Germany, 2017. https://ub-madoc.bib.uni-mannheim.de/43123.