MantisTable: an Automatic Approach for the Semantic Table Interpretation

Marco Cremaschi, Roberto Avogadro, and David Chieregato

University of Milan - Bicocca, Viale Sarca 336, 20126 Milan, Italy
marco.cremaschi@unimib.it
{r.avogadro,d.chieregato}@campus.unimib.it

1 Presentation of the system

1.1 State, purpose, general statement

On the Web we can find a vast amount of structured data, represented in tables that contain relevant information. Despite the huge corpus of such tables on different topics, they set limitations on artificial intelligence tasks, such as semantic search and query answering. This is the reason why some approaches started to propose extraction, annotation and transformation of tabular data into machine-readable formats. In particular, in the last years, there has been a ton of works on the annotation of tabular data, also known as *Semantic Table Interpretation (STI)*, which can be mainly classified as supervised (they exploit already annotated tables for training) [5,10,7,2] or unsupervised (they do not require training data) [13,8,1,6]; and as automatic [5,7,4] and semi-automatic [2]. Moreover, some approaches [3,13,8,1,6,12,11] focus mainly on the analysis of Web tables' context such as Web page title, table caption, or surrounding text, while others [5,10,7,4,9] address independent tables which can only rely on their own data. We identify some limits of the state-of-the-art approaches as follows: i) they adopt lexical comparisons for matching which ignore the contextual semantics; ii) they rely on metadata like column names and sometimes even external information like table descriptions, both of which are often unavailable in real world applications; iii) they use personalised Knowledge Graph (KG); iv) they perform only a few steps of STI. To overcome such limitations, we propose a comprehensive approach and a tool named *MantisTable*[1], which provides an unsupervised method to annotate independent tables, possibly without a header row or other external information. MantisTable takes a *well-formed and normalised* relational table (i.e. a table with headers and simple values, thus excluding nested and figure-like tables), and a KG which describes real-world entities in the domain of interest (i.e. a set of concepts, datatypes, predicates, entities, and the relations among them) in input, and returns a semantically annotated table in output. This process comprises different steps to semantically

[1] mantistable.disco.unimib.it

annotate tables, such as *semantic classification* of columns, which classifies a column either as a literal or as a named entity. Besides the approach, we propose *MantisTable tool*, a web interface and an open source Semantic Table Interpretation tool that automatically annotates, manages and makes the semantic of tables accessible to humans and machines. This tool is independent of any particular context. Additional built-in guidance functionalities help to avoid common pitfalls and to create correct annotations. Although a STI contains several steps, as will be explained in the next section, the key feature of our approach is the involvement of all the STI steps that run fully automatically. This approach and tool were developed by one PhD student and two master's students.

1.2 Specific techniques used

The *MantisTable* approach implements STI steps through five phases:

0. **Data Preparation**, which aims to prepare the data inside the table;
1. **Column Analysis**, whose tasks are the semantic classification, that assigns types to columns (NE-column or L-column), and the detection of the subject column (S-column);
2. **Entity Linking**, which deals with mappings between cells and entities in a KG;
3. **Predicate Annotation**, whose task is to find relations, in the form of predicates, between the main column and the other columns to set the overall meaning of the table;
4. **Concept and Datatype Annotation**, which deals with mappings between columns and semantic elements (concepts or datatypes) in a KG.

To describe each phase of the STI approach we consider Table 1[2], which lists video games with additional information, such as publisher, release date, etc.

Table 1. Table with a list of video games extracted from Round 1.

Title	Publisher	EU Release Date	AU Release Date	PEGI	ACB
Donkey Kong Country	Nintendo	2006-12-08	2006-12-07	7	G
Super Castlevania IV	Konami	2006-12-29	2006-12-29	3	PG
DoReMi Fantasy: Milon's DokiDoki Adventure (900 Wii Points)	Hudson Soft	2008-09-05	2008-09-05	3	G
...					

Data Preparation aims to clean and uniform data inside the table. Transformations applied to tables are as follows: deletion of HTML tags and some characters (i.e. " '), transformation of text into lowercase, deletion of text in brackets, resolution of acronyms and abbreviations, and normalisation of units of measurement. To decipher acronyms and abbreviations, the Oxford English Dictionary[3] is used. The normalisation of units of measurement is performed by applying regular expressions, as described in [8]. MantisTable extends the original set of regular expressions to cover a complete set of units, which includes

[2] Round 1 table index: 11833461_1_3811022039809817402

[3] public.oed.com/how-to-use-the-oed/abbreviations/

area, currency, density, electric current, energy, flow rate, force, frequency, fuel efficiency, information unit, length, linear mass density, mass, numbers, population density, power, pressure, speed, temperature, time, torque, voltage and volume.

Column Analysis whose tasks are the *semantic classification* that assigns types to columns that are named entity (NE-column) or literal column (L-column), and the *detection of the subject column* (S-column). The first step of the Column Analysis phase is to identify good L-column candidates. To accomplish this task, we consider 16 regular expressions that identify several Regextypes (e.g. numbers, geo coordinate, address, hex color code, URL). If the number of occurrences of the most frequent Regextype in a column exceeds a given threshold, that column is annotated as L-column, otherwise, it is annotated as NE-column. The second step deals with the *subject column detection* that takes into account the identified NE-columns. We can define the S-column as the main column of the table based on different statistic features, like Average Number of Words (aw) in each cell, Fraction of Empty Cells (emc) in the column, Fraction of Cells with Unique content (uc) and Distance from the First NE-column (df). These features are combined to compute the $subcol(c_j)$ score for each NE-column as follows:

$$subcol(c_j) = \frac{2uc_{norm}(c_j) + aw_{norm}(c_j) - emc_{norm}(c_j)}{\sqrt{df(c_j) + 1}} \quad (1)$$

The column with the highest score will be selected as the S-column for the considered table. The values of the features for the S-column detection related to the video games table (Table 1) are shown in Table 2. In this case the Title column is the S-column of the table (Table 3).

Table 2. Values of the features of the S-column detection for the video games table.

Feature	Title column	Publisher column
emc	0	0
uc	1	0.21
df	1	2
aw	1	0.37
final	3	0.57

Table 3. Table 1 after the Column Analysis phase.

S	NE	L	L	L	NA
Title	**Publisher**	**EU Release Date**	**AU Release Date**	**PEGI**	**ACB**
donkey kong country	nintendo	2006-12-08	2006-12-07	7	g
...					

Entity Linking deals with mappings between the content of cells and entities in the KG. The original version of MantisTable used a naive approach: we employed a SPARQL query considering only the entities for which, within the values of `rdf:type`, it was possible to find the label of the winning class. If more than one entity was returned for a cell, the one with a smaller edit distance (i.e. Wagner-Fischer distance) was taken. Between Round 3 and Round 4 we developed a more effective approach, which is described below. In the new approach we consider the table entities row by row. To discover the mappings for each cell in a row we take a set of *eligible* entities from the KG. We define

as *eligible* all the entities which label contains the cell's content or contains the cell's tokens as shown in Listing 1.1. By converting this information to a graph representation we find all the paths that, starting from a eligible entity of the cell's subject, enter into a eligible entity of the cell's object of the row. To increase the accuracy we also try to match the literal cells with the literals in the subject cell's candidate entities: that is, we seek for a match between literals in the table and eligible entities by using a simple matching algorithm. This algorithm distinguishes between three main datatypes: dates, numbers and strings. While dates and strings are matched exactly, numeric matching is done using an approximated matching algorithm: a pair of numbers makes a match if the distance (the absolute difference) between the two is less than a threshold. To pick the winning entities we apply to each path a score defined in the following way:

$$PS_r(p_i) = ||p_i|| + (1 - \sum_{j}^{||p_i||} editDistance(tx(j,r), e_{j,r})) \tag{2}$$

where PS_r is the path score of row r, p_i is an eligible path found, $||p_i||$ is the length of the path defined as the number of vertices contained in the path and $editDistance(tx(j,r), e_{j,r})$ is the edit distance (Levenshtein) between cell content and eligible entity. Then eventually we take the path with the maximum score.

Listing 1.1. SPARQL query to retrieve a set of eligible entities for a cell's content.

```
CONSTRUCT {
    ?s ?p ?o.
} WHERE {
    {
        {
            ?s ?p ?o.
            ?s rdfs:label ?label.
            ?label <bif:contains> '(<cell value>)'.
        } UNION {
            ?s ?p ?o.
            ?s rdfs:label ?label.
            ?label <bif:contains> '(<token> AND <token> [AND <token>])'.
        } UNION {
            OPTIONAL {
                ?altName rdfs:label ?lab.
                ?lab <bif:contains> '(<token> AND <token> [AND <token>])'.
                ?altName dbo:wikiPageRedirects ?s.
                ?s ?p ?o.
            }.
        }
    } UNION {
        ?o ?p ?s.
        ?s rdfs:label ?label.
        ?label <bif:contains> '(<token> AND <token> [AND <token>])'.
    }
    [...]
}
```

Predicate Annotation, whose task is to find relations in the form of predicates, between the Subject column and the other columns, to set the overall meaning of the table. MantisTable approach considers the set of predicates found in the Entity Linking phase and takes the predicate with maximum frequency.

Concept and Datatype Annotation deals with mappings between columns headers and semantic elements (concepts or datatypes) in a KG. Our approach reuse the information extracted by the linking phase (linked entities).

For each winning entity (E_{ij}) we pick the associated concepts (`rdf:type`), then for each column we build a dictionary C_j where each `rdf:type` have associated the number of rows of the table containing that concept divided by the total number of entities found for that column. We use a threshold set at 40% of the maximum score, the concepts with a score lower then this threshold are discarded. Table 4 show an example of this scoring.

Table 4. Example for C_j

Concept	Score
Person	0.7
Politician	0.5
Athlete	0.3
GolfPlayer	0.3
Place	0.3
PopulatedPlace	0.3
Settlement	0.3
City	0.3

With the concept list obtained in this way, we build a concept graph representing the hierarchy between them as shown in Figure 1. To identify the winning connected component within this graph, we take the maximum result of the following formula (connected component score) applied to each component:

$$CCScore(CC_i) = \sum_{n \in CC_i} conceptScore(n) \tag{3}$$

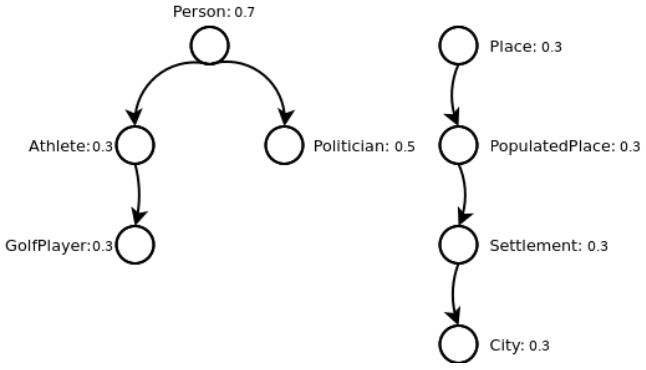

Fig. 1. Example concepts graph's paths

where CC_i is the ith connected component and *conceptScore* is a function that returns the corresponding concept score.

After this step, in order to complete the column annotation task we pick the path of the winning connected component which maximizes the score. We define the path's score in the following way, where p is a path of concepts defined as $c_1, c_2, \cdots, c_n$.

$$pathScore(p) = \sum_{i}^{n} conceptScore(p_i) \tag{4}$$

This method is being used to reinforce the identification of winning concepts in spite of the presence of noisy data or contingent errors in the knowledge base.

Without loss of generality let's consider the example in Table 4 and Figure 1. To calculate the score for the *Person* and *Place* connected components we calculate CCScore as in Equation 3.

$$\begin{aligned} CCScore(PersonCC) = conceptScore(Person) + conceptScore(Athlete) + \\ conceptScore(Politician) + conceptScore(GolfPlayer) \end{aligned}$$

$$\begin{aligned} CCScore(PlaceCC) = conceptScore(Place) + conceptScore(PopulatedPlace) + \\ conceptScore(Settlement) + conceptScore(City) \end{aligned}$$

therefore

$$\begin{aligned} CCScore(PersonCC) = 0.7 + 0.3 + 0.3 + 0.5 = 1.8 \\ CCScore(PlaceCC) = 0.3 + 0.3 + 0.3 + 0.3 = 1.2 \end{aligned}$$

so the winning connected component will be *PersonCC*.

Now let's see how path's score on *PersonCC* is computed. Let's pick some paths on this connected component defined as $p_1 = Person, Politician$ and $p_2 = Person, \cdots, GolfPlayer$, then the score is computed in the following way:

$$score(p_1) = conceptScore(Person) + conceptScore(Politician)$$

$$\begin{aligned} score(p_2) = conceptScore(Person) + conceptScore(Athlete) + \\ conceptScore(GolfPlayer) \end{aligned}$$

therefore

$$\begin{aligned} score(p_1) = 0.7 + 0.5 = 1.2 \\ score(p_2) = 0.7 + 0.3 + 0.3 = 1.3 \end{aligned}$$

so our method choose the p_2 path as the winning concept annotation.

2 Link to the system

As described above, the MantisTable approach has been integrated into a web application developed with Python and the Django. A MongoDB database acts as table and KG repository. The code is freely available through a Git repository[4]. In order to achieve the scalability of the application, and therefore improve efficiency, MantisTable has been installed in a Docker container to achieve parallelisms at the application level and to facilitate the deployment on servers. The management of resources is performed by using Task Queues (i.e. Celery Workers[5]). The five phases of the STI have been modularly implemented, allowing an easy replacement or extension by other developers.

2.1 Adaptations made for the evaluation

To participate in the challenge we made some changes as follows: i) MantisTable was originally developed to support the JSON format for loading tables and for exporting results. In order to take part in the challenge, we developed a new parser for managing the CSV tables. During this phase, we encountered several problems in the management of different characters encoding; ii) since target columns (columns to be annotated) were provided during the challenge, we disabled most of the Column Analysis phase (i.e. subject column detection); iii) our solution was made to identify just one correct Class per column, so we made a new export script with different criteria for the selection of concepts, also considering the hierarchy of these in the KG.

[4] bitbucket.org/disco_unimib/mantistable-tool.py
[5] docs.celeryproject.org/en/latest/userguide/workers.html

3 Results

In this section, the MantisTable approach's results in Round 1, Round 2, Round 3 and Round 4 of the challenge will be discussed. F1 and F2 in the following tables are referred to F1-Score and Precision for CEA and CPA Tasks in every Round and CTA in Round1 while they are referred to AH-Score and AP-score for CTA task in the Rounds from 2 to 4.

In the first round MantisTable achieved the results in Table 5. The results are good, in particular in the CEA task. In the second round MantisTable achieved the results in Table 6.

Table 5. Results of Round 1.

TASK	F1	F2
CTA	0.929	0.929
CEA	1.0	1.0
CPA	0.965	0.991

Table 6. Results of Round 2.

TASK	F1	F2
CTA	1.049	0.247
CEA	0.614	0.673
CPA	0.460	0.544

Table 7. Results of Round 3.

TASK	F1	F2
CTA	1.648	0.269
CEA	0.633	0.679
CPA	0.518	0.595

Table 8. Results of Round 4.

TASK	F1	F2
CTA	1.682	0.322
CEA	0.973	0.983
CPA	0.787	0.841

In Round 2 we particularly focused on the CTA task because it was crucial to our initial algorithm for the other steps; we used the results of the Concept Annotation to filter the results in the CEA and CPA tasks. About the CEA task, it is possible that during the parsing and the cleaning of data some rows were misplaced in different indexes. We didn't have much time to look into it but we are sure that we could have done a better job in this task using other resources for disambiguation (e.g. DBpedia or Wikidata). Most of the considerations we did in Round 2 are no longer valid with the new approach.

In the third round MantisTable achieved the results in Table 7. The results are better than the previous round both in the precision and in the F1-Score but the CEA and CPA tasks still have low precision due to too many false-positives: regarding the CEA task our approach did not have enough contextual information to discriminate between eligible entities, while for the CPA task our approach suffered for the entity linking.

In the fourth round MantisTable achieved the results in Table 8. To overcome the lack of contextual information needed by the linking phase, we changed our approach as described in the previous sections gaining a considerable precision boost for both the CEA and CPA tasks compared to previous rounds.

The results of CEA and CPA tasks with the new method are shown in tables 9 and 10.

Table 9. Results of Round 2 (recomputed).

TASK	**F1**	**F2**
CEA	0.799	0.897
CPA	0.663	0.702

Table 10. Results of Round 3 (recomputed).

TASK	**F1**	**F2**
CEA	0.94	0.977
CPA	0.723	0.759

4 Conclusions and General comments

Unlike the state of the art approaches, MantisTable i) provides a comprehensive solution to support all annotations steps; ii) provides an unsupervised method to annotate independent tables; iii) generates context for disambiguation; iv) provides a tool to support STI workflow and a tool to support the evaluation by providing validation indicators which are both publicly available. In relation to the results obtained in Round 4, we are going to change the workflow of our tool to better implement the methods for the CEA and CPA tasks. We are also going to develop a way to process tables via remote API calls to allow easier third party integration.

References

1. Deng, D., Jiang, Y., Li, G., Li, J., Yu, C.: Scalable column concept determination for web tables using large knowledge bases. Proc. VLDB Endow. **6**(13), 1606–1617 (Aug 2013)
2. Knoblock, C.A., Szekely, P., Ambite, J.L., Goel, A., Gupta, S., Lerman, K., Muslea, M., Taheriyan, M., Mallick, P.: Semi-automatically Mapping Structured Sources into the Semantic Web, pp. 375–390. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
3. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endow. **3**(1-2), 1338–1347 (Sep 2010)
4. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) The Semantic Web – ISWC 2013. pp. 363–378. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
5. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic Labeling: A Domain-Independent Approach, pp. 446–462. Springer International Publishing, Cham (2016)
6. Quercini, G., Reynaud, C.: Entity discovery and annotation in tables. In: Proceedings of the 16th International Conference on Extending Database Technology. pp. 693–704. EDBT '13, ACM, New York, NY, USA (2013)
7. Ramnandan, S., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning Semantic Labels to Data Sources, pp. 403–417. Springer International Publishing, Cham (2015)
8. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics. pp. 10:1–10:6. WIMS '15, ACM, New York, NY, USA (2015)
9. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a web of semantic data for interpreting tables. In: Proceedings of the Second Web Science Conference. vol. 5 (2010)
10. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. Web Semantics: Science, Services and Agents on the World Wide Web **37–38**, 152 – 169 (2016)
11. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. Proc. VLDB Endow. **4**(9), 528–538 (Jun 2011)
12. Wang, J., Wang, H., Wang, Z., Zhu, K.Q.: Understanding tables on the web. In: Proceedings of the 31st International Conference on Conceptual Modeling. pp. 141–155. ER'12, Springer-Verlag, Berlin, Heidelberg (2012)
13. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semantic Web **8**(6), 921–957 (2017)