MantisTable: a Tool for Creating Semantic Annotations on Tabular Data

Marco Cremaschi[1][0000−0001−7840−6228], Anisa Rula[1,2][0000−0002−8046−7502], Alessandra Siano[1], and Flavio De Paoli[1][0000−0002−5047−7371]

[1] University of Milan - Bicocca, Viale Sarca 336, Milan, Italy
{marco.cremaschi,anisa.rula,flavio.depaoli}@unimib.it
a.siano2@campus.unimib.it
[2] Univeristy of Bonn {rula}@cs.uni-bonn.de

Abstract. This paper describes MantisTable, an open source Semantic Table Interpretation tool, which automatically annotates tables using a Knowledge Graph. MantisTable provides a graphical interface allowing users to analyse the results of the semantic table interpretation process and validate the final annotations. The tool also provides a guided mode for viewing and editing annotations by users. Thanks to MantisTable features, it is possible to create semantic annotations and favour the publication and exchange of tabular data.

Keywords: Semantic Web · Ontology · Linked Data · Knowledge Graph · Semantic Table Interpretation · Semantic Annotations

1 Introduction & Motivation

A vast amount of structured data represented in tables that contain relevant information are available on the Web. Despite the huge corpus of such tables on different topics, they set limitations on artificial intelligent tasks such as semantic search and query answering. Some approaches started to propose [6, 5, 2] extraction, annotation and transformation of tabular data into machine-readable formats. The problem of semantically annotating tables, also known as *Semantic Table Interpretation (STI)*, presents different challenges as demonstrated by the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching[3] whose aim is benchmarking systems dealing with the semantic annotation tables.

STI 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 Knowledge Graph (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 annotate tables such as *semantic classification* of columns, which classify a column either in literal or named entity. These steps are usually performed manually or semi-automatically

[3] https://www.cs.ox.ac.uk/isg/challenges/sem-tab/

which require more users involvement who are often not familiar with semantic modelling. They need support to understand and explore the footprint of annotation steps.

Although previous efforts partially tackled the STI problem in the past [6], there is still no full support by single tools. DataGraft[4], a cloud-based service that provides an interface to transform tabular data into RDF triples, performs semantic annotations manually. TableMiner+ is a tool that supports only Web tables for which the user must provide a URL. Moreover, a few approaches that state to provide fully automatic tools, their code is not available or accessible, and in most cases it cannot be executed for imprecise information on the configuration settings. Only a few are open source and, to the best of our knowledge, none of them fully support users with the comprehensive STI steps through an interactive interface.

On one hand, it is worth noticing that the task of annotating semantics of tables is more complicated than the annotation of textual documents, due to the lack of a context, usually deductible in the case of text documents. On the other hand, the development of tools in this domain is characterised by a certain complexity because of several elements of the table simultaneously considered. In this paper, we propose *MantisTable*[5], a web interface and an open source Semantic Table Interpretation tool that automatically annotates, manages and makes accessible to humans and machines the semantic of tables. This tool is independent of any particular context. Additional built-in guidance functionalities help to avoid common pitfalls and create correct annotations. The current implementation of MantisTable is released as open source under the Apache 2.0 License. Although an STI contains several steps, as will be explained in the next section, the key feature of our tool is the involvement of all the STI steps that run fully automatically.

2 Overview of MantisTable

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

Data Preparation aims to clean and uniform data inside the table. Transformations applied to tables are as follows: remove HTML tags and stop words, turn text into lowercase, solve acronyms and abbreviations, and normalise measurements units. The latter is performed by applying regular expressions, as described in [3].

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 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., geo coordinate, address, hex color code, URL). If the number of occurrences of the most frequent Regextype in a column exceeds a given threshold,

[4] https://datagraft.io/

[5] http://mantistable.disco.unimib.it

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 (e.g. the percentage of cells with unique content and distance from the first NE-column).

Concept and Datatype Annotation deals with mappings between columns headers and semantic elements (concepts or datatypes) in a KG. In the first step of Concept Annotation, we perform the entity-linking by searching the KG with the content of a cell, to get a set of candidate entities. We use the DICE similarity measure between the content of the cell and the candidate entities to disambiguate the content of the cell. In the second step of Concept Annotation, the abstract and all concepts for each winning entity are retrieved from DBpedia[6]. For each extracted concept, we count the occurrences in the abstract. For the Datatype Annotation, the results of the Column Analysis are taken into consideration. To associate a Datatype to each column, a Regextype is applied on the content of each column.

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. MantisTable considers the winning concept of the S-column as the subject of the relationship, and the remaining columns as objects. Further, the entities identified as subjects and objects are further searched in the KG. In order to identify the correct predicate, we compare the content of the column and the candidate predicates.

Entity Linking deals with mappings between the content of cells and entities in the KG. The annotations obtained in the previous steps are used to create a query for the disambiguation of the cell contents. If more than one entity is returned for a cell, the one with a smaller edit distance (i.e., Wagner-Fischer distance) is taken.

3 Application Interface

MantisTable is a web application developed with NodeJs[7] and the Meteor framework[8]. The source code is released through a Git repository[9]. In order to scale and therefore improve efficiency, MantisTable has been installed in a Docker container to achieve parallelisation at the application level and facilitate the deployment on servers. The management of resources is performed by a load balancer (i.e., HAproxy[10]). The five phases of the STI have been modularly implemented allowing an easy replacement or extension by other developers.

MantisTable Loading and Storing. Tables are imported and stored in a MongoDB database. In MantisTable, a list of loaded tables is displayed on the

[6] For the demo we use DBpedia as one of the largest KG which covers different topics. Other KGs can be used.

[7] https://nodejs.org/

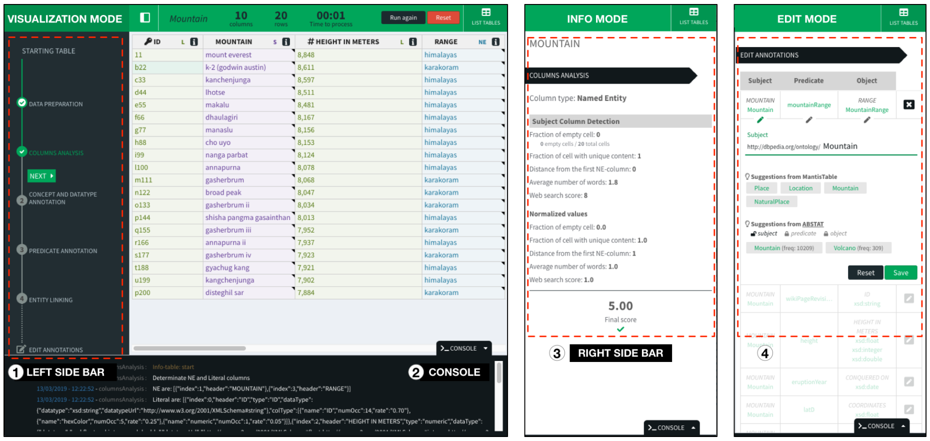
[8] https://www.meteor.com/

[9] https://bitbucket.org/disco_unimib/mantistable-tool/

[10] https://www.haproxy.org/

main page. For each table, a series of metadata is provided, such as name, date of loading, date of last modification, which tasks have already been performed and which are being executed thus making the user aware of the status of the annotation process. Finally, users can download the annotated tables at the end of the annotation process. Through the interface it is possible to add and load new tables (in JSON format), delete all tables of the Gold Standard (T2Dv2[11] and Limaye200 [1]) and process all tables in batch. It is also possible to update or delete every single table.

MantisTable Execution. After selecting a table, it is possible to manage the execution of the five phases described in Section 2. The user can either run all steps together or run them step-by-step to supervise the execution (Fig. 1 - info mode).



ID L	MOUNTAIN S	# HEIGHT IN METERS L	RANGE NE
11	mount everest	8,848	himalayas
b22	k-2 (godwin austin)	8,611	karakoram
c33	kanchenjunga	8,597	himalayas
d44	lhotse	8,511	himalayas
e55	makalu	8,481	himalayas
f66	dhaulagiri	8,167	himalayas
g77	manaslu	8,156	himalayas
h88	cho uyo	8,153	himalayas
i99	nanga parbat	8,124	himalayas
l100	annapurna	8,078	himalayas
m111	gasherbrum	8,068	karakoram
n122	broad peak	8,047	karakoram
o133	gasherbrum ii	8,034	karakoram
p144	shisha pangma gasainthan	8,013	himalayas
q155	gasherbrum iii	7,952	karakoram
r166	annapurna ii	7,937	himalayas
s177	gasherbrum iv	7,923	karakoram
t188	gyachug kang	7,921	himalayas
u199	kangchenjunga	7,902	himalayas
p200	disteghil sar	7,884	karakoram

```
nsAnalysis :   Info-table: start
nsAnalysis :   Determinate NE and Literal columns
13/03/2019 - 12:22:52 - columnsAnalysis :   NE are: [{"index":1,"header":"MOUNTAIN"},{"index":3,"header":"RANGE"}]
13/03/2019 - 12:22:52 - columnsAnalysis :   Literal are: [{"index":0,"header":"ID","type":"ID","dataType":
{"datatype":"xsd:string","datatypeUrl":"http://www.w3.org/2001/XMLSchema#string"},"colType":[{"name":"ID","numOcc":14,"rate":"0.70"},
{"name":"hexColor","numOcc":5,"rate":"0.25"},{"name":"numeric","numOcc":1,"rate":"0.05"}]},{"index":2,"header":"HEIGHT IN METERS","type":"numeric","dataType":
```

MOUNTAIN

COLUMNS ANALYSIS

Column type: **Named Entity**

Subject Column Detection

Fraction of empty cell: **0**
0 empty cells / **20** total cells
Fraction of cell with unique content: **1**
Distance from the first NE-column: **0**
Average number of words: **1.8**
Web search score: **8**

Normalized values

Fraction of empty cell: **0.0**
Fraction of cell with unique content: **1.0**
Distance from the first NE-column: **1**
Average number of words: **1.0**
Web search score: **1.0**

5.00
Final score ✓

EDIT ANNOTATIONS

Subject	Predicate	Object	
MOUNTAIN Mountain	mountainRange	RANGE MountainRange	☒

Subject
http://dbpedia.org/ontology/ Mountain

Suggestions from MantisTable
Place Location Mountain NaturalPlace

Suggestions from ABSTAT
subject *predicate* *object*
Mountain (freq: 10209) Volcano (freq: 309)

Reset Save

MOUNTAIN Mountain	wikiPageRevisi...	ID xsd:string	
MOUNTAIN Mountain	height	HEIGHT IN METERS xsd:float xsd:integer xsd:double	
MOUNTAIN Mountain	eruptionYear	CONQUERED ON xsd:date	
MOUNTAIN Mountain	latD	COORDINATES xsd:float	

Fig. 1. MantisTable interface overview: Visualization Mode (1. left side bar, 2. console), Info Mode (3. right side bar), Edit Mode (4. edit form)

MantisTable Exploration. It is possible to navigate all the executed steps by clicking on each phase and analyse the results in the visualization mode (Fig. 1 - 1. left side bar). For all phases, additional information about the execution is shown in the console located under the table (Fig. 1 - 2. console). By clicking on a header or body cell, information about the current phase is reported in the info mode (Fig. 1 - 3. right sidebar).

Fig. 1 shows the table after the *Column Analysis*. Different colours are used to immediately distinguish the different columns types: S-column, NE-columns, and L-columns. The L-column headers show representative icons of the Regex-type that has been assigned to them. By clicking the header cells contextualised information is displayed, e.g. features for the identification of S-column are displayed for NE-columns, and the distribution of candidate regular expressions are shown for L-column.

[11] http://webdatacommons.org/webtables/goldstandardV2.html

After the *Concept and Datatype Annotation* phase, clicking on a header cell opens the right sidebar to display the selected concepts (for NE-columns) and datatypes (for L-column). Clicking on a body cell, two tabs are opened to display the candidate entities (with the similarity scores computed to select the winning one), and candidate concepts associated with the entities (with score). To enhance the awareness of the user, the abstract from DBpedia related to the winning concept is also reported. With the same approach, the user can browse the predicate annotations by first selecting that phase on the left, and then view the annotation relating to the relationship between the S-column and the columns on the right sidebar.

MantisTable Editing. Even if MantisTable implements a fully automated annotation process, it is important to allow users to understand what has been achieved and give them the opportunity to modify and enhance the results. The former has been achieved with the exploration features sketched above, the latter has been accomplished by providing a widget to edit the annotations (Fig. 1 - 4. edit mode). The annotation validation and editing require that the user has previous knowledge about the structure of the KG. Therefore, to support the user we integrate ABSTAT[12] [4], a Resource Description Format (RDF) data set profiling tool. It takes a dataset and the relative ontology (in OWL format) in input, and produces a summary and some statistics about the dataset. In addition, ABSTAT provides access to summaries via APIs that allow to easily extract information about the KG such as the frequency of a particular concept in the KG or the frequency of a particular type of pattern. In essence, ABSTAT provides information on how concepts and properties are used within DBpedia, thus supporting the user to make better choices.

4 Acknowledgments

This research has been supported in part by EU H2020 projects EW-Shopp - Grant n. 732590. Special thanks to Roberto Avogadro, David Chieregato, Carlo Mattioli and Blerina Spahiu for their support during the development of the project.

References

1. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. VLDB **3**, 1338–1347 (2010)
2. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic Labeling: A Domain-Independent Approach, pp. 446–462. Springer (2016)
3. 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 (2015)

[12] http://backend.abstat.disco.unimib.it

4. Spahiu, B., Porrini, R., Palmonari, M., Rula, A., Maurino, A.: Abstat: ontology-driven linked data summaries with pattern minimalization. In: ISWC. pp. 381–395. Springer (2016)
5. 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 (2011)
6. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semantic Web **8**(6), 921–957 (2017)