Future Generation Computer Systems 112 (2020) 478–500

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A fully automated approach to a complete Semantic Table Interpretation

Marco Cremaschi *, Flavio De Paoli, Anisa Rula, Blerina Spahiu

University of Milan - Bicocca, Viale Sarca 336, 20126 Milan, Italy

ARTICLE INFO

Article history:
Received 10 February 2019
Received in revised form 7 May 2020
Accepted 14 May 2020
Available online 27 May 2020

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

ABSTRACT

In recent years, there has been an increasing interest in extracting and annotating tables on the Web. This activity allows the transformation of text data into machine-readable formats to enable the execution of various artificial intelligence tasks, e.g. semantic search and dataset extension. Semantic Table Interpretation is the process of annotating elements in a table. Current approaches are mainly based on lexical matching algorithms that rely on metadata associated with tables or custom Knowledge Graphs. Their main limitations are due to the lack of metadata, the little use of contextual semantics, and the incompleteness of the proposed methods that do not include all the necessary steps. In this paper, we propose a comprehensive approach and a tool that provides an unsupervised method to annotate independent tables, possibly without header row or other external information. The approach is based on the definition of a context created from the elements within the table in order to discriminate among matching entities found in shared Knowledge Graphs and create high-quality annotations. The approach has achieved excellent results in an international challenge, thus proving its effectiveness.

1. Introduction

A vast amount of information is provided as structured data on the Web in the form of tables. To size the phenomenon, an analysis reports 25M relational tables within 500M Web pages [1], and another estimates 150M HTML tables represented in English language [2]. More recent work highlighted the popularity of the tables by collecting 233M tables through the analysis of the Common Crawl[1] repository [3]. This significant increase can be linked to the uptake of the Open Data movement, whose purpose is to make a large number of tabular data sources freely available, addressing a wide range of domains, such as finance, mobility, tourism, sports, or cultural heritage [4]. Tables are essential to perform queries, but the implicit or visual structures employed in tables are not easily machine-readable. In order to allow computers to interpret, combine and reuse such data for several artificial-intelligence tasks (such as classification, clustering, filtering, and retrieval [5]), the semantics of data should become explicit. Therefore, an underlying requirement is identifying and annotating entities in cells, their types and the connections between entities.

Underlying the above requirement is the core idea of using Semantic Table Interpretation (STI) tasks with the intent to capture knowledge from tables. The input of STI is (i) *a well-formed and normalised* relational table (i.e. a table with headers and simple values, thus excluding nested and figure-like tables), as the one in Fig. 1, and (ii) a *Knowledge Graph* (KG), which describes real world entities in the domain of interest (i.e. a set of concepts, datatypes, predicates, instances, and the relations among them), as the example in Fig. 2. The output returned is a semantically annotated table, as shown in Fig. 3.

Moreover, the STI process is composed of the following main annotation steps: (i) *semantic classification of columns*, which takes into account the values of a column to mark it as *Literal column (L-column)* if values are datatypes (e.g. strings, numbers, dates such as 4808, 10/04/1983), or as *Named-Entity column (NE-column)* if values are concepts (e.g. Mountain, Mountain Range such as Mont Blanc, Mont Blanc massif); (ii) *detection of the subject column (S-column)*, which identifies the main column (the one all the others are referring to) among the *NE-columns* identified in the previous step (e.g. the Name column in Fig. 3); (iii) *concept and datatype annotation*, which associates *NE-columns* with a concept in the KG (e.g. the column Name is associated with Mountain in DBpedia[2]), and *L-columns* with a datatype in the KG (e.g. the column Coordinates is of type georss:point); and

* Corresponding author.
E-mail addresses: marco.cremaschi@unimib.it (M. Cremaschi), flavio.depaoli@unimib.it (F. De Paoli), anisa.rula@unimib.it (A. Rula), blerina.spahiu@unimib.it (B. Spahiu).

[1] http://commoncrawl.org/.

[2] http://dbpedia.org/resource/Mountain.

https://doi.org/10.1016/j.future.2020.05.019
0167-739X/

(iv) *predicate annotation*, which identifies the relations between the *S-column* and the other columns (e.g. Name dbo:elevation Height).

Each of the above steps is obtained by annotating column values referring to existing KGs. For example, in Fig. 3 if the majority of entities in the Name column is associated with dbo:Mountain, these entities are of type dbo:Mountain. Similarly, dbo:elevation can be identified as the predicate connecting entities in the Name column with datatypes of type xsd:integer of the Height column. Unfortunately, clear situations like the one in the example are not so common; therefore, we need to set up strategies and algorithms to address more complex scenarios.

The annotation steps of the STI involve several key challenges [6]: (i) *disambiguation*: the concepts of the entities described in a table are not known in advance, and those entities may correspond to more than one concept in the KG. For example, the entity Mont Blanc in Fig. 3 can refer to different entities associated with different concepts. As we might know, Mont Blanc is the name of a mountain, but, also, the name of a tunnel, a poem, and a dessert[3]; (ii) *homonym*: which is related to the presence of different entities with the same name and the same concept. In addition to the famous mountain located on the French-Italian border[4], there is also a mountain with the same name on the Moon[5]; (iii) *matching*: the name of the entity in the table may be syntactically different from the name in a KG. Johannisberg mountain refers to the Johannisberg (High Tauern) entity in DBpedia[6]; (iv) *missing context*: it is often easier to extract the context from textual documents than from tables due to the amount of content to be processed. On the other hand, disambiguating possible meanings of literals results to be more difficult.

In the last years, there has been several works on STI which can be mainly classified as supervised (they exploit already annotated tables for training) [7–10] or unsupervised (they do not require training data) [11–14]; and as automatic [7,9,15] or semi-automatic [10]. Moreover, some approaches [1,11–14,16,17] focus mainly on the analysis of the content of Web pages (containing the table) such as Web page title, table caption, or surrounding text, while others [7–9,15,18] 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) no use of contextual semantics: the contextual semantics, when available, extends the content of the cell by using table features such as header and row content (internal elements) or an entity with further information from the considered KG. Although exploiting the contextual semantics in the entity matching process can be of advantage for producing better results, most of the approaches focus only on lexical comparisons (e.g. TF-IDF, Jaccard) [6,19]. (ii) rely on external metadata: some approaches are based on external elements in terms of metadata such as table descriptions [13,14,16,17], that are not always available for real-world tables (for example, some tables do not have these elements, like SemTab2019 Challenge). (iii) adoption of personalised Knowledge Graphs (KGs); referring to specific domains or aimed at particular purposes, thus preventing the generalisation of the annotation approaches [12,15–18]. (iv) identification of only a subset of possible annotations (*NE-columns*, *L-columns*, and the relations between them) [1,7–9,12–14,16,18–21]. (v) lack of implementations and working tools [12,22–25].

To overcome such limitations we propose a comprehensive approach and a tool named MantisTable,[7] which provides an unsupervised and fully automated approach for annotating tables even without a header row or other external information using a KG.

Although there exist numerous approaches, there are only a few tools available to support STI (Section 5.2). Currently, in the state-of-the-art, there are only four tools with a graphical interface [22–25] (but some deprecated or not working) and one working only from the command line [12]. Considering the available and working tools, they all require manual or semi-manual intervention by the user who is often not familiar enough with the semantic modelling to complete the process. Due to the complexity of the STI and the simultaneous involvement of several elements of the table, the development of an STI tool is a complex task. MantisTable performs the table annotation process independently and the user, through an interactive interface, can explore each step and have a better understanding for each element involved in the final annotations. At the end of the process, the user is also supported in editing the annotations provided by the tool to enhance the results.

Our experiments with T2Dv2 Gold Standard[8] (from the general Web) and Limaye Gold Standard [1] (from Wikipedia pages) have shown that our method is effective and can outperform the state-of-the-art approaches as will be discussed in Section 4.3.

To guarantee replicability, we use a Docker build to collect all the necessary configurations and dependencies to run the tool. Moreover, the source code of MantisTable is open and freely available[9] to let interested researchers understand and improve the implemented STI process.

In summary, the main contributions of this paper are as follows: (i) a comprehensive approach which deals with all phases of an STI process (annotating column headers, value cells and identifying relationships between columns). Experiments have shown that the approach has achieved excellent results, which led to an award at the international Semantic Web Challenge on Tabular Data to Knowledge Graph Matching[10]; (ii) MantisTable is integrated into a user-friendly Web app that allows users to visualise and explore the results of each step and won the Best Demo Award during ESWC2019[11]; (iii) an additional tool, called STILTool, which helps the user to evaluate the results of STI by using different Gold Standards (i.e. T2Dv2, Limaye200 and the datasets of the above challenge); (iv) a systematic review of the state-of-the-art approaches and tools.

The rest of the paper is organised as follows: Section 2 describes the whole STI workflow in detail while in Section 3 MantisTable is described. Section 4 introduces the Gold Standards, the configuration parameters and finally discusses the evaluation results. An overview of the state-of-the-art in relevant scientific areas is provided in Section 5. Finally conclusions and pointers are presented in Section 6.

2. Semantic Table Interpretation: MantisTable workflow

MantisTable approach consists of the following phases:

0. **Data Preparation**, which aims to prepare the data inside the table;
1. **Column Analysis**, whose objective is the semantic classification that assigns types to columns (*NE-column* or *L-column*), and the detection of the subject column (*S-column*);

3 https://en.wikipedia.org/wiki/Mont_Blanc_(disambiguation).
4 http://dbpedia.org/resource/Mont_Blanc.
5 http://dbpedia.org/resource/Mont_Blanc_(Moon).
6 http://dbpedia.org/resource/Johannisberg_(High_Tauern).
7 http://zoo.disco.unimib.it/mantistable/.
8 http://webdatacommons.org/webtables/goldstandardV2.html.
9 https://bitbucket.org/disco_unimib/mantistable-tool.py/.
10 https://www.cs.ox.ac.uk/isg/challenges/sem-tab/.
11 https://2019.eswc-conferences.org/awards/.

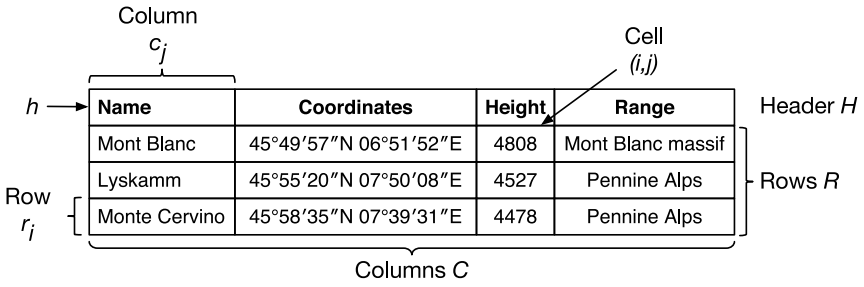

Fig. 1. Example of a well-formed relational table.

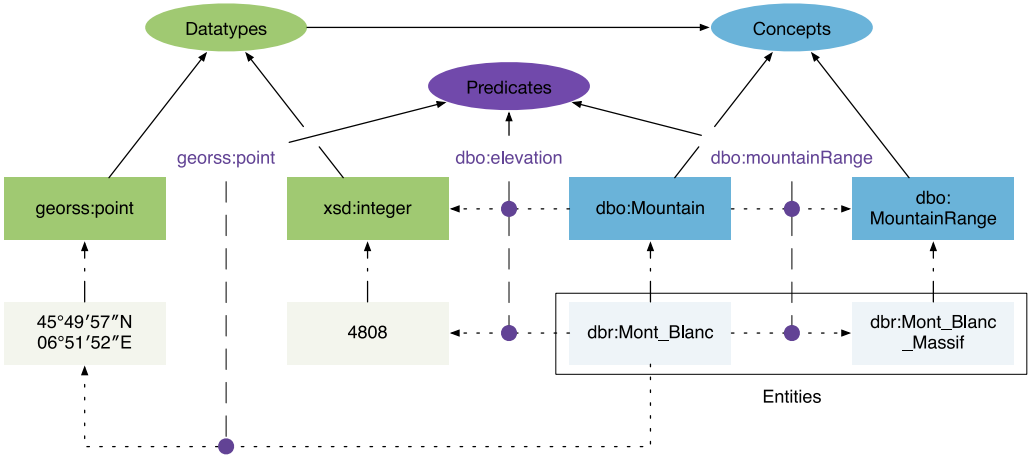

Fig. 2. A sample of Knowledge Graph.

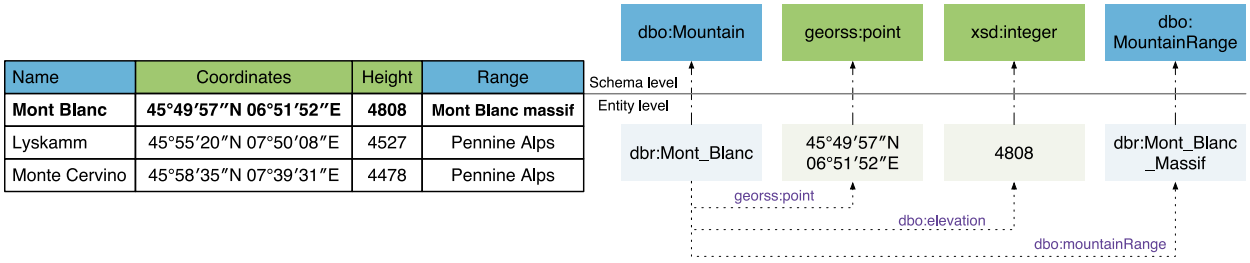

Fig. 3. Example of an annotated table.

2. **Concept Annotation and Datatype Annotation**, which deals with mappings between columns (or headers, if they are available) and semantic elements (concepts or datatypes) in a KG;
3. **Predicate Annotation**, whose objective is to find relations, using predicates, between the main column and the other columns to set the overall meaning of the table;
4. **Entity Linking**, which deals with mappings between cells and entities in a KG.

To describe each phase of the STI approach, consider Table 1, which lists the highest peaks in the world with additional information, such as heights and coordinates. The table has been extracted from the T2Dv2 Gold Standard,[12] and extended by adding new columns (i.e. COORDINATES, URL, DESCRIPTION, TEMPERATURE and a column with BOOLEANS) in order to demonstrate each phase of the approach.[13]

2.1. Data preparation

Before starting the annotation process, it is advisable to apply standard transformation rules to the values and the structure of the table so as to preserve only the relevant information. Examples of transformation are the following: deletion of HTML tags and some characters (i.e. " '), transformation of text into lowercase, deletion of text in brackets, explanation of acronyms and abbreviations, and normalisation of units of measurement. To decrypt acronyms and abbreviations, the Oxford English Dictionary[14] is used. The normalisation of units of measurement is performed by applying regular expressions, as described in [12]. MantisTable extends the original set of regular expressions to cover a complete set of units, which includes 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. For each type of unit of measurement, a set of conversion rules has been defined, like the

[12] T2Dv2 table index: 14311244 0 7604843865524657408 - http://webdatacommons.org/webtables/goldstandardV2.html.

[13] https://bitbucket.org/disco_unimib/mantistable-tool/src/master/app/private/tables/test/mountains/.

[14] http://public.oed.com/how-to-use-the-oed/abbreviations/.

Table 1
The table reports a list of the highest peaks in the world from T2Dv2 Gold Standard, extended with other columns to consider a variety of situations (grey columns).

ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD	BOOL	URL	DESCRIPTION	T
11	Mount Everest	8.848 km	Himalayas	May 29, 1953	27.98785, 86.92502	1	https://en.wikipedia.org /wiki/Mount_Everest	Mount Everest, known in Nepali as Sagarmāthā and in Tibetan as Chomolungma, is Earth's highest mountain above sea level [..]	-35C
b22	K-2	8,611 m	Karakoram	July 31, 1954	35.87998, 76.51510	0	https://en.wikipedia.org /wiki/K2	K2, also known as Mount Godwin-Austen or Chhogori, is the second highest mountain in the world, after Mount Everest (8,848 metres), at 8,611 metres (28,251 ft).	-30C
c33	Kanchenjunga	8.597 km	Himalayas	Wednesday, 25 May 1955	27.70249, 88.14753	true	https://en.wikipedia.org /wiki/Kangchenjunga	Kangchenjunga, also spelled Kanchenjunga, is the third highest mountain in the world, and lies partly in Nepal and partly in Sikkim, India.	-29C
D44	Lhotse	8,511 m	Himalayas	-429926400	27°57'45.4"N 86 56'1.4"E	no	en.wikipedia.org /wiki/Lhotse	Lhotse is the fourth highest mountain in the world at 8,516 metres (27,940 ft), after Mount Everest, K2, and Kangchenjunga.	-28C
...									

Table 2
Table 1 after the data preparation phase.

ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD	BOOL	URL	DESCRIPTION	T
11	mount everest	8.848 km	himalayas	may 29, 1953	27.98785, 86.92502	1	https://en.wikipedia.org/wiki/mount_everest	mount everest, known in nepali as sagarmatha and in tibetan as chomolungma, is earth's highest mountain above sea level [..]	−35c
b22	k-2	8,611 m	karakoram	july 31, 1954	35.87998, 76.51510	0	https://en.wikipedia.org/wiki/k2	k2, also known as mount godwin-austen or chhogori, is the second highest mountain in the world, after mount everest (8848 m), at 8611 m (28,251 ft).	−30c
c33	kanchenjunga	8.597 km	himalayas	wednesday, 25 may 1955	27.70249, 88.14753	true	https://en.wikipedia.org/wiki/kangchenjunga	kangchenjunga, also spelled kanchenjunga, is the third highest mountain in the world, and lies partly in nepal and partly in sikkim, india.	−29c
d44	lhotse	8511 m	himalayas	−429926400	27° 57'45.4"n 86 56'1.4"e	no	https://en.wikipedia.org/wiki/lhotse	lhotse is the fourth highest mountain in the world at 8516 metres (27,940 ft), after mount everest, k2, and kangchenjunga.	−28c

ones presented in Listing 1, which shows the equivalent ways to express measures of area.

Table 2 shows the transformation of Table 1 after the completion of the Data Preparation phase. It is worth noticing that in this simulation the data preparation phase failed in the normalisation of some rows (e.g. the CONQUERED ON column). Despite this, the tool will be able to deal with these kinds of values.

2.2. Column analysis

The first step of column analysis considers the semantic classification of columns into either Literal columns (*L-column*) for datatype values (e.g. strings, numbers, dates, such as 4808, 10/04/1983), or Named-Entity columns (*NE-column*) for concept values (e.g. Mountain, Mountain Range, such as Mont Blanc, Mont Blanc massif). The first activity requires the identification of good *L-columns* candidates. Our method exploits Regular Expressions (RegEx) to check if the content of the cells of a column can be classified as empty, date, number and long text as described in [11,12,14,15,26]. In addition, the set of RegEx is extended with 14 new *Regextypes*: *geo coordinates, iso8601 date, street address, hex colour, url, image file, credit card, email address, ip address, isbn (International Standard Book Number), boolean, id, currency and iata (International Air Transport Association) codes*. If the number of occurrences of the most frequent *Regextypes* detected exceeds a given threshold (see Section 4.3), the column will be annotated as *L-column* and the most frequent *Regextype* will be assigned to the column under analysis; otherwise the column will be annotated as *NE-column*. For example, in the mountain table (Table 2), this step assigns the *L-column* to the COORD column, with values of *Regextypes* geo coordinates, and the *NE-column* to the PEAK column.

The second step of Column Analysis considers the subject column detection that takes into account the *NE-columns*. The subject column (*S-column*) is the main column in the table: it contains entities for which the other columns provide additional information. To identify the *S-column*, our method considers the following list of features (Table 3): (i) fraction of empty cells (emc), (ii) fraction of cells with unique content (uc), (iii) distance from the first *NE-column* (df), and (iv) average number of words (aw). The first three are taken and adapted from [11], while the fourth is taken and adapted from [17]. In contrast to [11], MantisTable does not consider features that are expensive (in computational, economic, and maintenance terms), such as the Web search (ws) and context match score (cm). The former requires the use of a search engine, while the latter analyses the external contexts (e.g. webpage title, table caption and surrounding paragraphs). As shown in Section 4.3, the use of these features does not bring improvements in the quality of the annotations. The work in [17] inspired us to consider the average number of words (aw) feature, which calculates the average number of words within the cells of each column. It has been empirically demonstrated that the best candidate is the column with the highest average number of words per cell. In Section 4, we present an analysis of the performance of the Subject Column Detection using different configurations and features.

The four features, after the normalisation, are combined to compute the $subcol(c_j)$ score for each *NE-column* using Formula (1), which is an adaptation (due to the different features considered) of the one presented in [11].

$$subcol(c_j) = \frac{2uc_{norm}(c_j) + aw_{norm}(c_j) - emc_{norm}(c_j)}{\sqrt{df(c_j) + 1}} \qquad range : [0, 3] \qquad (1)$$

Listing 1: Conversion table for units of measurement of area.

```
area | squareMetre | squaremetre, m2, m², μ2, μ², τ.μ.
area | squareMillimetre | squaremillimetre, mm2, μμ2, μμ² | 1.0E-6
area | squareCentimetre | squarecentimetre, cm2, cm² | 0.0001
area | hectare | hectare, ha | 10000.0
area | squareMile | squaremile, sqmi, mi2, mi² | 2589988.110336
```

Table 3
Features for subject column detection.

Feature	Notation
Fraction of empty cells	emc
Fraction of cells with unique content	uc
Distance from the first NE-column	df
Average number of words in each cell	aw

Table 4
Values of the features of the *S-column* detection for the mountain table.

Feature	PEAK column	RANGE column
emc	0	0
uc	1	0.1
df	0	2
aw	1.6	1
subcol	3	0.5

The column with the highest score will be selected as the *S-column* for the considered table. Table 4 shows the values of the features related to the mountain table (Table 1), and the *subcol* scores that identify the PEAK column as the *S-column* of the table (Table 5). In order to promote columns with unique values, as a typical characteristic of the object columns established in [17], more weight is given to the *uc*.

2.3. Concept annotation and datatype annotation

The main purpose of the concept and datatype annotation phase is to annotate the *NE-columns* by searching a KG to extract the related concepts, and annotate *L-columns* with datatypes.

a. Concept Annotation: in the first step of this phase, the approach performs the entity-linking on *NE-columns* by searching the KG with the content of a cell $tx(i, j)$. The first *NE-column* considered is the *S-column*, since it should contain a high number of distinct values and unambiguous entities (see the description of feature unique content *uc* in Section 2.2) which facilitates the identification of a valid candidate concept for the annotation of the column. The approach selects at most k cells of the column to keep the size of the problem bounded and ensures a reasonable response time (the value of k will be discussed in Section 4.3). The content of the k cells is used to search the KG and get a set of candidate entities. The content of the cell *tx(i,j)* and the candidate entities $E_{i,j} \subseteq E$ are used to disambiguate the content of the cell by considering the degree of similarity.

Given a candidate entity $e_{i,j} \in E_{i,j}$, the similarity depends on two components: entity context (*econtext*) and entity name (*ename*). *econtext* is a score that represents the similarity between the description of the entity $e_{i,j}$ in the KG and the cell context $x_{i,j} \in X_{i,j}$ of cell *tx(i,j)*. *ename* is a score that represents the similarity between the name of the entity $e_{i,j}$ in the KG and the words in cell *tx(i,j)*. Inspired by the formula of *econtext* provided in [11], which is given as the intersection between the bowsets of different contexts, we propose a new and straightforward intuition; if an entity occurs many times in its context then it can be considered a relevant candidate. Instead of the token intersection of the *ename* formula in [11], we use the edit distance because tokens in the table cells may contain some misspellings that do not affect the edit distance score.

All the terms of the following formulae are normalised using the classic "dividing by maximum" normalisation. *econtext* is calculated by Formula (2), where the bowset function returns the set of unique words from a bag-of-words (bow) after morphological normalisation and stop-word removal have been applied:

$$econtext(e_{i,j}) = |bowset(edescription(e_{i,j})) \cap bowset(rcontext(i))| + |bowset(edescription(e_{i,j})) \cap bowset(hcontext(j))| \quad (2)$$

$$range: [0, 1]$$

In this case, the similarity calculation considers the number of tokens in common between the description (textual information describing the entity, e.g. abstract) of the entity with the content of the header and with the row. The description of the entity $e_{i,j}$ is $edescription(e_{i,j})$, which is retrieved from the reference KG. If the reference KG does not support descriptions, we try to extract this information from other KGs, for example by exploiting the links provided by the sameAs predicate (e.g. we can exploit YAGO sameAs predicate to find similar entities in DBpedia). In DBpedia such a description is given by the dbo:abstract predicate, while other KGs use different predicates (e.g. the description predicate in Wikidata and Freebase, or the comment predicate in OpenCyc). For cases where it is not possible to retrieve this information from KGs (lack of the sameAs predicate, or absence of any descriptive predicate), a Web search can be used instead of proposing a substantial improvement of the method adopted in [11]. However, as described above, it is a solution to be avoided as the cost per search result increases over time and APIs of the search engines are changing frequently.

The cell context $x_{i,j}$ is represented by row context $rcontext_{i,j}$ and header context $hcontext_{i,j}$, which are defined as follows:

- $rcontext_{i,j}$ is the concatenation of all the words in the cells in the same row i for every column j, without considering the content of cell *tx(i,j)*;
- $hcontext_{i,j}$ is the concatenation of all the words in the header of column j plus the concatenation of all the synonyms extracted from shared vocabularies (e.g. from Wordnet[15] and Oxford dictionary).

In contrast to [11], which considers all 7 contextual features for calculating the score *econtext* (column header, row content, column content, Web page title, table caption and/or title, paragraphs and semantic markups), we consider only some of them. If we classify the above features as in-table (the information in table rows and columns) and out-table (the information in the hosting Web page) context features, the approach considers only the in-table context features since the out-table context features are cumbersome and do not bring any advantage as will be discussed in Section 4.3. Moreover, while [11] uses Dice to measure similarity, MantisTable does not consider the frequency of tokens because this characteristic often rewards incorrect candidate entities, as we verified experimentally.

[15] https://wordnet.princeton.edu/.

Table 5
The example table after Column Analysis phase.

L *id*	S	L *number*	NE	L *date*	L *geo coordinate*	L *bool*	L *url*	L *long text*	L *temp*
ID	PEAK	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESCRIPTION	T
11	mount everest	8848 m	himalayas	may 29, 1953	27.98785, 86.92502	1	en.wikipedia.org /wiki/Mount_Everest	mount everest, known in nepali [..]	-35C
b22	k-2	8611 m	karakoram	july 31, 1954	35.87998, 76.51510	0	en.wikipedia.org /wiki/K2	k2, also known as [..]	-30C
c33	kanchenjunga	8597 m	himalayas	may 25, 1955	27.70249, 88.14753	true	en.wikipedia.org /wiki/Kangchenjunga	kangchenjunga also spelled kanchenjunga [..]	-29C
d44	lhotse	8511 m	himalayas	-429926400	27°57'45.4"N 86°56'1.4"E	no	en.wikipedia.org /wiki/Lhotse	lhotse is the fourth [..]	-28C

Table 6
Class frequencies for each extracted entity.

PEAK	Entity	rdf:type	Type	Frequency
Mount everest	http://dbpedia.org/resource/Mount_Everest	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [..]	Place Mountain NaturalPlace Location	2 3 1 1
k-2	http://dbpedia.org/resource/K-2_(Kansas_highway)	dbo:Place, dbo:Location, dbo:ArchitecturalStructure, dbo:Infrastructure, dbo:Road, dbo:RouteOfTransportation, schema:Place [..]	Place Location ArchitecturalStructure Infrastructure Road RoadOfTransportation	2 1 1 1 1 1
kangchenjunga	http://dbpedia.org/resource/Kangchenjunga	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [..]	Place Mountain NaturalPlace Location	2 3 1 1
lhotse	http://dbpedia.org/resource/Lhotse	dbo:Place, dbo:Location, dbo:Mountain, dbo:NaturalPlace, schema:Mountain, schema:Place, umbel:Mountain [..]	Place Mountain NaturalPlace Location	2 3 1 1

ename is calculated by computing the edit distance (Levenshtein distance) between the labels (in different languages) of candidate entity $e_{i,j} \in E_{i,j}$ and the content of the cell *tx(i,j)*:

$$ename(e_{i,j}) = editDistance(tx(i,j), e_{i,j}) \qquad range : [0, 1] \tag{3}$$

The final objective is to identify the entity with the highest confidence score (*econf*), which will then be used for annotating the cell. Before computing the *econf*, all scores have to be normalised by dividing each score by the maximum value obtained for that score. The confidence score *econf* is computed as follows:

$$\begin{aligned} econf(e_{i,j}) = {} & bonusnorm(e_{i,j}) + econtextnorm(e_{i,j}) \\ & - ename(e_{i,j}) * 2 \qquad range : [-2, 2] \end{aligned} \tag{4}$$

In this case, the *ename* is given more weight because we want to reward entities that have a lower edit distance, and therefore are more likely to be correct. Empirical results demonstrate the validity of this heuristic. Another difference with [11] is the use of Dice similarity with different weights for scores. The new score $bonusnorm(e_{i,j})$ in Formula (4) is used to rank entities that are most related to the content of the cell. Ranking is performed by considering the presence of words from the text in the cell within the entity labels and the entity abstract (Formula (5)). It is computed as the intersection between each candidate entity's label and the cell content considering every token independently.

$$\begin{aligned} bonus(e_{i,j}) = {} & |bowset(tx(i,j)) \cap bowset(e_{i,j})| \\ & + |bowset(tx(i,j)) \cap bowset(edescription(e_{i,j}))| \end{aligned}$$

$$range : [0, 1] \tag{5}$$

Related to the mountain table (Table 1) for each cell *tx(i,j)* a set of candidate entities $E_{i,j}$ is extracted from the DBpedia with the query shown in Listing 2. MantisTable actually uses DBpedia as it contains real, large scale data, and is challenging enough to assess the abilities of an STI system. In any case, the only DBpedia specific predicate used in the workflow is the dbo:abstract predicate, which can be replaced as discussed above. The query searches the KG considering both the entire content of the cell and the individual words. In addition, MantisTable searches for the descriptions associated with the entities according to the synonyms of the header. Values in the header are assumed to be nouns, thus the respective synonyms are extracted from WordNet, which is a semantic-lexical database of the English language, and from the thesaurus of Oxford dictionary. In addition, we improve the method in [11] by searching for the descriptions associated with the entities according to the synonyms of the header. For instance, considering the example in Table 1, the approach searches the KG with the synonyms of the header of the *S-column* PEAK, which results in *summit, mountain, bluff, ridge*.[16] The maximum number of results per query is set to 10; this value has been defined empirically after several tests in which the correct result was mostly found within the first 5 results.

In this example the row and header context are:

- $rcontext_{i,j}$: 11, 8848, himalayas, may 29, 1953, 27.98785 86.92502, en.wikipedia.org/wiki/Mount Everest ...;
- $hcontext_{i\,j}$: the synonyms of the header PEAK are summit, mountain, bluff, ridge, ben, beg, jebel, mount.

Listing 3 shows the candidate entities with the associated bonus for the cell with value "Mount Everest". It is evident that candidates with lower bonus score are less correlated with the analysed cell.

[16] https://www.lexico.com/en/synonym/peak.

Listing 2: SPARQL query to retrieve a set of candidate entities for a text in a cell.

```
SELECT DISTINCT (str(?s) as ?s) (str(?edescription) as ?edescription)
WHERE {
{
    ?s dbo:abstract ?edescription .
    ?s a ?type .
    ?s rdfs:label ?label .
    ?label <bif:contains> 'mount AND everest' .
    ?edescription <bif:contains> '("peak" OR "summit" OR "mountain" OR [synonyms])' .
  }
  FILTER NOT EXISTS { ?s dbo:wikiPageRedirects ?r2 } .
  FILTER (!strstarts(str(?s), 'http://dbpedia.org/resource/Category:')) .
  FILTER (!strstarts(str(?s), 'http://dbpedia.org/property/')) .
  FILTER (!strstarts(str(?s), 'http://dbpedia.org/ontology/')) .
  FILTER (strstarts(str(?type), 'http://dbpedia.org/ontology/')) .
  FILTER (lang(?edescription) = 'en') .
}
ORDER BY ASC(strlen(?label))
LIMIT 10
```

Listing 3: Candidate entities with relative bonus score for the "Mont Everest" cell.

```
South_Summit_(Mount_Everest), Bonus Score: 2
Mount_Everest, Bonus Score: 4
Mount_Everest_Boarding_School, Bonus Score: 4
1922_British_Mount_Everest_expedition, Bonus Score: 4
Into_Thin_Air:_Death_on_Everest, Bonus Score: 3
The_Man_Who_Skied_Down_Everest, Bonus Score: 3
Mount_Everest_Nepal, Bonus Score: 4
[..]
```

Table 7
Global frequency values for the PEAK column.

Type	Global frequency	# cells
Place	28	14
Mountain	36	13
NaturalPlace	13	13
~~ArchitecturalStructure~~	1	1
~~Infrastructure~~	1	1
~~Road~~	1	1
~~RouteOfTransportation~~	1	1

At this stage we can compute *econtext*, *ename* and *econf* scores by applying the Formulae (2)–(4) which are reported in Listing 4 for the cell with value "Mount Everest". In this case, the winning entity is dbr:Mount Everest.[17]

The second step of the Concept Annotation collects the identified winning entities $e_{i,j}$ into a set of concepts $CO_{i,j} \subseteq CO$, from which the concept to be associated with the column j will be extracted.

In particular, for each winning entity, all the rdf:type[18] values are extracted from DBpedia. Then, for each extracted type, the frequency (considering also different ontologies) and the number of cells in which the type appears are calculated. Table 6 reports the frequencies referred to the example in Table 1 and the column PEAK.

To avoid possible incorrect links, we decided to select a set of types whose frequency is close to the maximum frequency of all types of the candidate list, up to a certain threshold. In other terms, we select the set of type candidates, which satisfy two conditions: (i) fall within the range defined by the maximum global frequency as upper bound and δ as lower bound; (ii) if the type belongs to a number of cells greater than a threshold β (the value of δ and β will be discussed in Section 4.3). Therefore, the final result from the phase is the one shown in Table 7.

From the ontology, the hierarchy of concepts in the example above is *Place* > NaturalPlace > Mountain and Place > Location. The frequency of NaturalPlace is summed to the frequency of Mountain. The most specific concept from the two branches of the hierarchy is used to annotate the column. Fig. 4 shows the workflow of Concept Annotation.

b. Datatype Annotation: Datatype annotation relies on the results of the Column Type Analysis (Section 2.2) that delivered an association between every *L-column* and a specific *Regextype*.

Table 8 reports a mapping between *Regextypes* and Datatypes. In the case of one-to-one relation, the corresponding datatype is used to annotate the *L-column*, otherwise a further analysis step is required to identify the correct datatype, as described in the next Section 2.4. Table 9 shows the example in Table 1 with final columns annotations.

2.4. Predicate annotation

For Predicate Annotation, the MantisTable approach assumes the winning concept of the *S-column* as the subject of relationships among columns, and annotations of the other columns as objects. To identify candidate predicates, we exploit two complementary techniques based on exploratory queries and summary profiles. The former searches the KG for the subject and the object with two distinct methods for *NE-columns* and *L-columns* (Section 2.4.1), while the latter uses a distributed tool that integrates profiles for RDF data (Section 2.4.2). In this way, the efficiency of the approach additionally improves by returning faster searches.

2.4.1. Predicate annotation using exploratory queries

a. Predicate Annotation for NE-column: We proceed by obtaining all candidate predicates (predicates in the Resource Description Format (RDF) triples) between the instances of the *S-column* concept and the instances of the *NE-column* c_m. Given the concept co_j of the *S-column* c_j and the instances of the *NE-column* c_m, we select predicates from triples whose subjects are of type co_j and the objects are instances of the *NE-column* c_m. In addition, we execute a query where, given the concept co_m of the *NE-column* c_m and the instances of the *S-column* c_j, we select predicates form triples whose subjects are all instances of the *S-column* c_j and the object are of type co_m. Unlike [11], we do not use the entities associated with the cells, but the content of the cells

17 http://dbpedia.org/resource/Mount_Everest.
18 http://www.w3.org/1999/02/22-rdf-syntax-ns%23type.

Listing 4: List of entities with entity context score, entity name score and confidence score.

```
"dbr:Mount_Everest"
  score econtext 1
  score ename 0
  score bonus 1
  score econf 2.5 # winning entity
"dbr:Mount_Everest_Nepal"
  score econtext 0.06
  score ename 0.31
  score bonus 1
  score econf 0.79
"dbr:Mount_Everest_webcam"
  score econtext 0.11
  score ename 0.35
  score bonus 1
  score econf 0.61
"dbr:Joint_Himalayan_Committee"
  score econtext 0.10
  score ename 0.8
  score bonus 0.5
  score econf -0.78
[..]
```

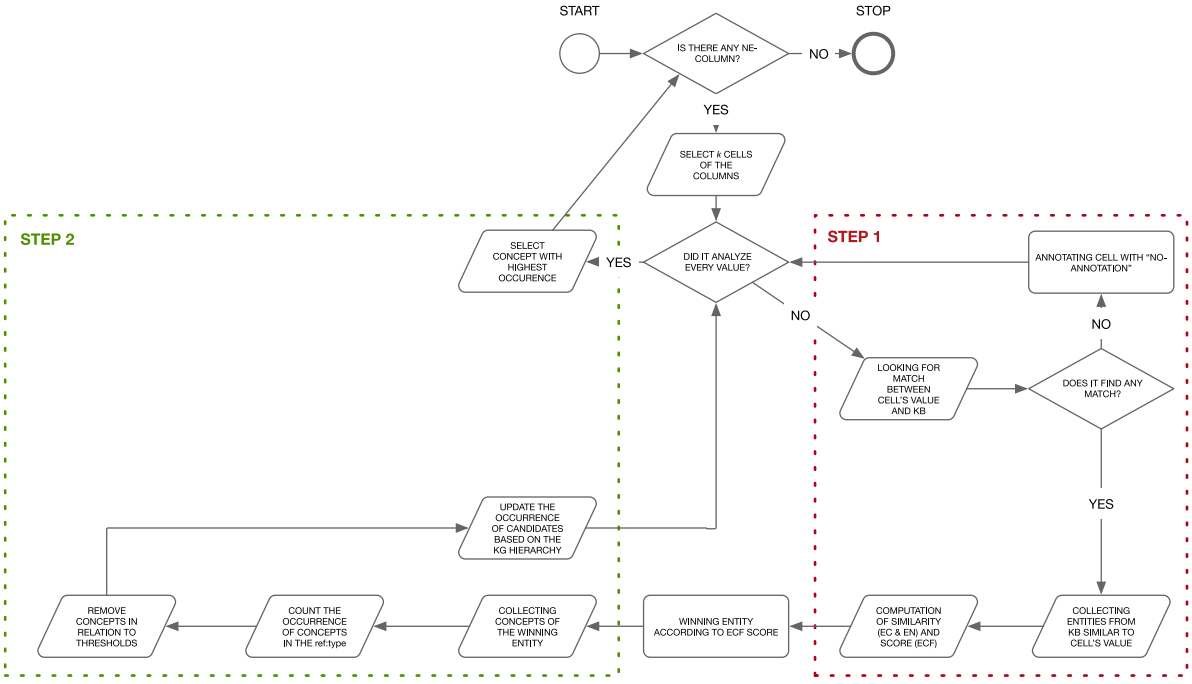

Fig. 4. Concept annotation workflow.

Table 8
Datatype and *Regextype* mapping.

Regextype	Datatype	Description
geo coordinates	xsd:float	32-bit floating point
address	xsd:string	string
hex colour	xsd:string	string
numeric	xsd:double	64-bit floating point
	xsd:float	32-bit floating point
	xsd:integer	integer value
	xsd:nonPositiveIntegerbyte	integer containing only non-positive values (..,−2,−1,0)
	xsd:negativeInteger	integer containing only negative values (..,−2,−1)
	xsd:nonNegativeInteger	integer containing only non-negative values (0,1,2,..)
	xsd:positiveInteger	integer containing only positive values (1,2,..)
boolean	xsd:boolean	boolean (true or false)
URL	xsd:anyURI	Uniform Resource Identifier
image	xsd:string	string
credit card	xsd:string	string
email	xsd:string	string
IP	xsd:string	string
ISBN	xsd:string	string

and the concepts associated with the columns under analysis. The above procedure is important because some entities may not use all the predicates related to their concepts. For example, there are no predicates for dbo:MountBlanc and dbo:MountBlanc massif,

Table 9
Table 1 with column annotations.

-	dbo:Mountain	xsd:integer	dbo:Mountain_Range	xsd:date	georss:point	-	dbo:wikiPageDisambiguates	-	-
ID	MOUNTAIN	HEIGHT	RANGE	CONQUERED ON	COORD.	B	URL	DESC.	T
11	mount everest	8848 m	himalayas	may 29, 1953	27.98785, 86.92502	1	en.wikipedia.org /wiki/Mount_Everest	[..]	-35C

while there exist three predicates between dbo:Mountain and dbo:MountainRange that are: dbo:mountainRange, dbo:parentMountainPeak and dbo:locatedInArea. In order to have a larger set of candidate predicates, we use the concept related to those entities in our queries.

The result is a set $P_m \subseteq P$ predicates. In order to identify the correct predicate $p_m \in P_m$ for *NE-column* c_m, we compare the content of the column and the candidate predicates. Given a candidate predicate p_m, the predicate confidence score (*pconf*) depends on two components: the predicate context (*pcontext*) and predicate frequency (*pfreq*). *pcontext* is a score that represents the similarity between the representation of the predicate from the KG with the representation in the *NE-column*. *pcontext* is calculated by comparing a candidate predicate p_m with the context $x_m \in X_m$. As already introduced in Section 2.3, we consider the in-table context (header context), and the out-table context (Web search). The approach is similar to the one proposed in [11], with the difference that they consider, for the out-table context, paragraphs and semantic markups in the page hosting the table, which are not always available (as in the Challenge and in the T2Dv2 Gold Standards). Unlike the Column Annotation, for this step we consider the Web search. Even though the Web search can be expensive, for the Predicate Annotation, experiments have shown that by using few terms (cell content for the *NE-column*) good results can be achieved (see Section 4.3). We calculate the overlap between the representation of the candidate predicate p_m and the representation of each context x_m using the Dice similarity, computed as follows:

$$pcontext(p_m) = \sum_{x_m \in X_m} \frac{\sum_{w \in bowset} (p_m) \cap bowset(x_m)\ (bow(p_m)[w] + bow(x_m)[w])}{sum(bow(p_m)) + sum(bow(x_m))}$$
$$range : [0, 2] \quad (6)$$

$bow(p_m)$ is the bow representation of a predicate name obtained by splitting on uppercase (e.g. mountainRange generates the couple mountain, range), and including some synonyms (e.g. mountain, range, mountain chain). The denominator returns the sum of all word frequencies in bow. The selection of the winning candidate is computed using *pconf*:

$$pconf(p_m) = pcontext(p_m) + pfreq(p_m) \quad range : [0, 3] \quad (7)$$

where $pfreq(p_m)$ is the frequency of the predicate pm in the query of Listing 5.

Listing 5 shows a query returning a set of candidate predicates P_m from the KG. Two distinct parts in the query can be identified. In the first part, we look for all the properties having entities of type dbo:Mountain as their subject and some values extracted from the *NE-column* (e.g. Himalayas, Karakoram). In the second part, we look for all properties having some values extracted from the *S-column* as their subject (e.g. Mount Everest, Kangchenjunga) and some entities of type dbo:Mountain Range as objects.

The Web search exploits the *websearch* feature which is the aggregation (union of the snippets) in a single bowset of the results from a search engine fed with the content of the cells of the *NE-columns*.

In this example the header context and Web search for the cell *m,j* are:

- $hcontext_{m,j}$: the synonyms of the header RANGE are limit, sierra, line, span
- $websearch_{m,j}$: the bowset for the searching term karakoram is the set wikipedia the karakoram or karakorum is a large mountain range spanning the border of pakistan a significant part 28 50 of the karakoram range is glaciated compared to the himalaya 8 12 and alp 2 2 mountain glacier may serve a karakoram range mountain asia britannica com pakistan the himalayan and karakoram range the himalaya which has long been a physical and cultural divide between south and central asia ...

The query shown in Listing 5 returns the set of candidate predicates marked in blue as in Listing 6 while the scores are computed with Formulae (6) and (7). In this case, the winning predicate is dbo:mountainRange.[19]

b. Predicate Annotation for L-column: as for the Predicate Annotation for *NE-column*, we execute a query where the subject is co_j from the *S-column* while the objects are values of the target *L-column*. In this case, since we have values instead of entities, in addition to the concept co_j, we also consider the synonyms of the header of the target *L-column* (if present) to increase the number of predicate candidates.

The query shown in Listing 7 returns the set of candidate predicates marked in blue as in Listing 8 while the scores are computed with Formulae (6) and (7).

c. Predicate Annotation for numerical values: the approach for Predicate Annotation of *L-columns* discussed so far often produces unsatisfactory results in the case of columns containing numerical values because of the way they are stored in a KG (see Section 4.3 for further discussion). To improve the quality of the annotations, we extend the method proposed in [4], which applies a hierarchical clustering algorithm on a reference KG to build a Background Knowledge Graph (BKG). The BKG contains information about the set of possible values for each pair predicate-concept. Consider the pair <age and People> from the BKG, the set of values for this pair might be in the range [0-110] considering that the maximum age for all the People in the KG is 110 years old. Other pair examples from a BKG are <temperature and City> and <longitude/latitude and City>. A k-nearest neighbour takes as input the set of numerical data extracted from an *L-column* and assigns a pair predicate-concept. If a set of numerical values is associated with the predicate height, the set may represent either the height of buildings, of persons, as well as the elevation of mountains. Given two numerical sets, one in input (from the table to be annotated), and one inside the BKG, the approach in [4] analyses and compares the distribution of numerical values through the distance of Kolmogorov–Smirnov (KS) statistic D [27]. In summary, if two numerical sets are equally distributed, i.e. the two sets hold the same numeric values, then the statistic D converges to 0. The application of the method proposed in [4] returns top-k candidate pairs (predicate-concept). We extended [4] by ranking the top-k using the similarity of the concept inside the candidate pairs with respect to the concept co_j annotated with the *S-column*.

[19] http://dbpedia.org/ontology/mountainRange.

Listing 5: SPARQL query to retrieve a set of candidate properties for a NE-column.

```
SELECT DISTINCT ?p (str(?plabel) as ?plabel) (count(?p) as ?count)
WHERE {
  {
    VALUES ?o {
      <http://dbpedia.org/resource/Himalayas>
      <http://dbpedia.org/resource/Karakoram>
      [..]
    } {
      ?s ?p ?o .
      ?p rdfs:label ?plabel .
      ?s a <http://dbpedia.org/ontology/Mountain> .
      FILTER (lang(?plabel)="en")
    }
  }
  UNION {
    VALUES ?s {
      <http://dbpedia.org/resource/Mount_Everest>
      <http://dbpedia.org/resource/Kangchenjunga>
      [..]
    } {
      ?s ?p ?o .
      ?p rdfs:label ?plabel .
      ?o a <http://dbpedia.org/ontology/MountainRange> .
      FILTER (lang(?plabel)="en")
    }
  }
}
```

Listing 6: List of predicates with predicate context, predicate frequency and predicate confidence score.

```
"dbo:locatedInArea"
  frequency 13
  score pcontext 0.0868
  score pfreq 0.0050
  score pconf 0.0918
 "dbo:mountainRange"
  frequency 1904
  score pcontext 0.9898
  score pfreq 0.9294
  score pconf 1.9193 # winning predicate
"dbo:regionCode"
  frequency 13
  score pcontext 0.0050
  score pfreq 0.0193
  score pconf 0.0244
```

Listing 7: SPARQL query to retrieve a set of candidate properties for a L-column.

```
SELECT DISTINCT ?p (str(?plabel) as ?plabel) (count(?p) as ?count)
WHERE {
  {
    VALUES ?o {
      8.848 8.611 8.597 8.511 8.481 8.167 8.156 8.153 8.124 8.078
    } {
      ?s ?p ?o .
      ?p rdfs:label ?plabel .
      ?s a <http://dbpedia.org/ontology/Mountain> .
      FILTER (lang(?plabel)="en")
    }
  }
  UNION {
    ?s ?p ?o .
    ?p rdfs:label ?plabel .
    ?plabel bif:contains '("height" OR "meters" OR [synonyms])' .
    FILTER (lang(?plabel)="en") .
    FILTER isLiteral(?o)
  }
}
LIMIT 50
```

Listing 8: List of predicates with predicate context, predicate frequency and predicate confidence score.

```
"dbo:elevation"
  frequency 238
  score pcontext 0.9674
  score pfreq 1.0635
  score pconf 1.321 # winning predicate
"dbo:heightMetric"
  frequency 1
  score pcontext 0.0040
  score pfreq 0.8634
  score pconf 0.2918
"dbo:heightDatum"
  frequency 5
  score pcontext 0.0203
  score pfreq 0.8634
  score pconf 0.3081
```

For calculating the similarity we use the following criteria:

1. *Exact Match.* The similarity is set to 1 if the candidate is the same co_j concept of the *S-column*. In such case, the searching is completed and the candidate can be selected as the annotation for the column. For example, if we consider the column HEIGHT in Table 9, the top-k predicate-concept candidates pair are reported in Listing 9. Since the co_j concept of the *S-column* MOUNTAIN is dbo:Mountain, and such concept is in the set of top-k, then the associated predicate for the column HEIGHT is dbo:elevation.
2. *Similarity match.* It is possible that the concept co_j of the *S-column* is not among the top-k candidates, but instead an equivalent concept is present. The set of equivalent concepts (defined by the predicate owl:equivalentClass) can be retrieved from DBpedia (e.g. dbo:Location and dbo:Place). As for the exact match, if at least one of the identified equivalent concepts is present among the top-k results, the associated predicate can be selected to annotate the column.
3. *Common ancestor match.* If the concept co_j of the *S-column*, or equivalent concepts, does not match with any of the candidate concepts, then super-classes of those concepts can be considered with the aim to exclude the candidates that do not belong to the same branch of the hierarchy of the concepts. Super-classes can be retrieved from DBpedia by exploiting the predicate rdfs:subClassOf. For example, let us consider another table extracted from T2Dv2 Gold Standard,[20] containing information about Countries and Capital cities. In this table the *S-column* is annotated with the concept dbo:Country and such concept is not present in the set of the top-k as shown in Listing 10. In this case, we retrieve from DBpedia the hierarchy of concepts for dbo:Country. If a more general concept with respect to the *S-column* annotation is present in the pairs we select it for the annotation of the *L-column*. Numerical values referring to different concepts and predicates in different domains can share the same distribution, for example, numerical values related to the capacity of stadiums <populationTotal and Location> and <numberOfStudents and Agent>. Therefore, starting from the set of top-k candidates, the SPARQL query in Listing 11 counts the number of the common super-concept to co_j. In this case, the only common ancestor is the root concept (i.e. owl:Thing of DBpedia), thus the candidate and its predicate are not considered. Hence, the ordering of the predicates is maintained, and, at the end of the comparisons, the predicate with the highest confidence score (*pconf*) is selected among the top-k results.
4. *Predicate confidence score.* In case the application of the first three criteria does not identify a predicate, then the one with the highest predicate confidence score is selected (see Listing 8).

After the Predicate Annotation phase of the STI the example in Table 1 is annotated as in Fig. 5. Sometimes it is not possible to find a relation between some columns and the *S-column*. In particular, this phase is difficult for those columns that contain too generic values (as the column B of type xsd:boolean). Through this example, we could highlight such limitations.

2.4.2. Predicate annotation using summaries profiles

The queries shown in Listings 5 and 7 may take too long to get an answer. This is due to the high number of predicates that might relate an *S-column* with the values of an *L-column*. Sometimes, queries cannot be executed because of timeout issues at the endpoint server. In order to overcome such limitations and to increase the efficiency of the approach, we have integrated ABSTAT,[21] a distributed tool for calculating and exploring summaries for RDF data. ABSTAT takes as input a dataset and the related ontology (in OWL[22] format), and produces as output a summary and statistics about the dataset. ABSTAT's summary is a collection of patterns of the form <subjectType, pred, objectType>, which represent the occurrence of triples <sub, pred, obj> in the data, such that subjectType is the most specific type of the subject and objectType is the most specific type of the object. With the term type we refer to either an ontology class (e.g. foaf:Person) or a datatype (e.g. xsd:DateTime). Statistics includes frequencies for classes and predicates that are present in patterns. Fig. 6 shows an example of patterns extracted by ABSTAT. The first line is a pattern <dbo:Mountain dbo:mountainRange dbo:MountainRange> that states that there are instances of type Mountain linked to instances of type MountainRange using the predicate mountainRange. For each pattern several statistics are returned. The frequency of the pattern shows how many times this pattern occurs in the dataset. The number of instances shows how many instances have this pattern including those for which the types Mountain and MountainRange and the predicate mountainRange can be inferred. Max (Min, Avg) subjs-obj cardinality is the maximal (minimal, average) number of distinct entities of type Mountain linked to a single entity of type MountainRange through the predicate mountainRange. Max (Min, Avg) subj-objs is the maximal (minimal, average) number of distinct entities of type MountainRange linked to a single entity of type Mountain through the predicate.

For the annotation of the predicates, MantisTable exploits the ABSTAT's APIs to extract the top n predicates that link two concepts. The top n are selected according to frequency for each pattern. More weight is given to the patterns that occur often in the dataset. The right predicate for the annotation is selected by calculating *pconf* (Formula (7)).

The advantage of using ABSTAT profiles is three-fold. First, ABSTAT summary patterns are indexed to support a very efficient and fast exploration process. Second, ABSTAT makes APIs available to users so they can make use of "query templates" and construct queries very easily. This is an ability that has been proven to be very efficient and helpful to explore and write queries about the data faster and more accurately [28]. Third, for every concept and predicate, ABSTAT extracts frequencies about their use inside the KG. Frequency is a crucial statistic for the identification of top-k predicates that relate concepts in a KG.

2.5. Entity linking

Entity Linking is the last phase of the approach. The content of a cell *tx(i,j)* is used to create a query to retrieve the best candidate entities for that cell. An example for the column MOUNTAIN in Table 1 is reported in Listing 12.

The annotations obtained in the previous phases can be used to filter the set of results and identify the best candidate entity. However, it has been experimentally noticed that the use of *NE-column* annotations is not always effective. For instance, not all entities of the *NE-column* have the same rdf:type. Therefore,

[20] T2Dv2 table index: 14311244 0 7604843865524657408 - http://webdatacommons.org/webtables/goldstandardV2.html.

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

[22] https://www.w3.org/OWL/.

Listing 9: List of predicate-concept pairs for column HEIGHT of Table 1.

```
http://dbpedia.org/ontology/Location (width)
http://dbpedia.org/ontology/Device (length)
http://dbpedia.org/ontology/Weapon (length)
http://dbpedia.org/ontology/Mountain (elevation)
http://dbpedia.org/ontology/Place (populationDensity)
[..]
```

Listing 10: List of concepts-predicate pairs for column “area sq.km.”.

```
http://dbpedia.org/ontology/Town (areaWater)
http://dbpedia.org/ontology/City (areaWater)
http://dbpedia.org/ontology/Location (areaWater)
http://dbpedia.org/ontology/Settlement (areaWater)
http://dbpedia.org/ontology/Place (area)
[..]
```

Listing 11: SPARQL query to retrieve a set of super-concept of a concept.

```
SELECT ?numentity (count(distinct ?entity) as ?count)
WHERE {
        VALUES ?numentity { <top-k candates> }
        <subjectConcept t> rdfs:subClassOf* ?entity .
        ?numentity rdfs:subClassOf* ?entity2 .
        FILTER (?entity = ?entity2).
  }
```

Listing 12: An example of SPARQL query to retrieve candidate entities for cell content “mount everest”.

```
SELECT (str(?s) as ?s) ?type
WHERE {
  ?s rdfs:label ?l .
  ?l <bif:contains> '("mount everest" OR ("mount" AND "everest"))' .
  ?s rdf:type ?type .
  FILTER (lang(?l) = "en")
}
```

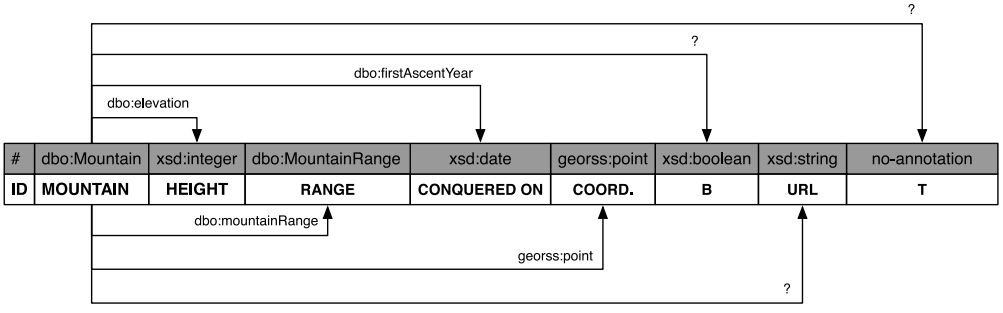

Fig. 5. The example table with Predicate Annotations.

	subject type (occurrences)	predicate (occurrences)	object type (occurrences)	frequency	instances	Max subjs-obj	Avg subjs-obj	Min subjs-obj	Max subj-objs	Avg subj-objs	Min subj-objs
filter	dbo:Mountain	predicate	dbo:MountainRange								
	dbo:Mountain (16123)	OP dbo:mountainRange (13104)	dbo:MountainRange (2457)	9972	9972	1653	11	1	3	1	1
	dbo:Mountain (16123)	OP dbo:locatedInArea (46400)	dbo:MountainRange (2457)	50	50	9	2	1	1	1	1
	dbo:Mountain (16123)	OP dbo:parentMountainPeak (2463)	dbo:MountainRange (2457)	17	17	15	6	1	1	1	1
	dbo:Mountain (16123)	OP owl:differentFrom (50072)	dbo:MountainRange (2457)	7	7	2	1	1	1	1	1
	dbo:Mountain (16123)	OP rdfs:seeAlso (156792)	dbo:MountainRange (2457)	3	3	1	1	1	1	1	1

Fig. 6. ABSTAT patterns filtered for the subType Mountain.

when the query returns more than one entity, the edit distance (i.e. Levenshtein distance) is computed between the normalised table cell content and the normalised candidate entity label (the value of rdfs:label). Only the entity with the smallest edit distance is considered for the annotation. Listing 13 shows the results for the cell "mount everest" of column MOUNTAIN.

3. MantisTable tool

This section describes MantisTable, the tool that implements the STI workflow described in Section 2. MantisTable was developed to overcome the lack of working tools in the STI state-of-the-art: it provides an easy way to manage and annotate tables and export the results in different formats.

It is an open-source Web application available through a Git repository.[23] It is developed with the Django framework[24] in Python, and exploits a MongoDB[25] database as table and KG repository. MantisTable has been encapsulated in a Docker container to facilitate the deployment and scalability by replication. The management of messages is performed by using Task Queues (i.e. Celery Workers[26]). Data management is supported by the MongoDB Sharding[27] method to split and distribute data across multiple machines, and the use of a unique MongoDB Shard Key.

Each of the five phases of the STI have been implemented as an independent component to facilitate evolution, replacement and extensions of single parts. Fig. 7 shows the modular architecture of MantisTable, which is organised in three layers. The first one is the View Layer that provides a graphic user interface to serve different types of tasks such as storing and loading tables, executing the STI phases and exploring the annotated tables. Moreover this layer allows the exploration of the results that can be further edited to enhance the annotation quality. The Controller Layer creates the abstractions between the View layer and the Model layer, and implements all the STI phases. Finally, the Model Layer manages mainly data access components to communicate with external data sources such as DB and DBpedia connectors.

In the MantisTable interface, a list of loaded tables is displayed on the main page. The user can select a table by using the associated information (table name, date of loading, date of last modification, already completed phases and phases still pending) and control the execution of STI phases through the menu at the top. More specific information is displayed on individual tables and in the panel on the right (Fig. 8).

MantisTable annotations can be exported in several formats (RDF/XML,[28] N3,[29] NTriples,[30] N-Quads,[31] Turtle[32] and JSON-LD[33]) and include the following data:

- Concept Annotation: "Table ID", "Column ID" and "DBpedia classes";
- Datatype Annotation: "Table ID", "Column ID" and "datatype";
- Predicate Annotation: "Table ID", "Head Column ID", "Tail Column ID", and "DBpedia predicate";
- Entity Linking: "Table ID", "Column ID", "Row ID" and "DBpedia entity".

4. Experiments

This section describes the evaluation of our approach. The aim of the experiments is to evaluate the correctness of the overall approach and, separately, of each single phase. We also compare our results with state-of-the-art approaches. Finally, we describe STIL Tool that is used to support the evaluation of the annotations.

4.1. Datasets

For the first part of the experiments, we use two Gold Standards: Version 2 of the T2D[34] and Limaye200 [1].

- The T2Dv2 Gold Standard consists of manually annotated row-to-instance, attribute-to-property and table-to-concept correspondences between 779 Web tables and the DBpedia Knowledge Base Version 2014.[35] The tables originate from the English-language subset of the Web Data Commons Web Tables Corpus.[36] As described in [29], during the extraction, the tables have been classified in layout, entity, relational, matrix and other tables. Concerning our goal, we select the relational tables because they contain information about an entity. In particular, from the 779 tables, we select 234 which share at least one instance with DBpedia. The tables cover different topics including places, jobs and people. Altogether, the Gold Standard contains 33 concepts, 25119 entities and 618 predicate correspondences. The correspondences were created manually. Inside the T2Dv2, the tables are structured in JSONformat; the JSON also contains additional text, which represents the external context of the respective table. The correspondences between the data in the table and the KG (the semantic annotations) are contained in 3 CSV files, where the different types row-to-istance, attribute-to-property and table-to-concept are respectively reported. The first type annotates each row of a table to an entity within the KG; the second provides an annotation for the relations between *S-columns* and other columns using predicates; the last one associates the whole table with a unique concept.
- The Limaye200 [11] is a dataset consisting of 200 Wikipedia tables extracted from LimayeAll [1]. For each table the types of columns (i.e. *NE-column*, *L-column* and *S-column*) are identified. The *NE-columns* annotation phase uses the concepts in Freebase. As previously described, MantisTable uses DBpedia as KG. In order to evaluate our annotation for each entity associated with a cell, the corresponding mapping between Freebase and DBpedia was extracted. The concepts are then used as an annotation for the column. In this case the tables are structured as XML.

Limaye200, compared to T2Dv2, is much smaller in terms of the total number of rows, while the number of total columns for both Gold Standards is comparable. The "Structuredness" field in Table 10 indicates how many blank cells are present within tables. In particular, the Structuredness is a weighted sum of each table's Structuredness, where the weight of each table is based on its sum of columns and rows, normalised by the total sum of columns and rows [30]. Considering the two Gold Standards, the

23 https://bitbucket.org/disco_unimib/mantistable-tool.py.
24 https://www.djangoproject.com/.
25 https://www.mongodb.com/.
26 https://docs.celeryproject.org/en/latest/userguide/workers.html
27 https://docs.celeryproject.org/en/latest/userguide/workers.html
28 https://www.w3.org/TR/rdf-syntax-grammar/.
29 https://www.w3.org/TeamSubmission/n3/.
30 https://www.w3.org/TR/n-triples/.
31 https://www.w3.org/TR/n-quads/.
32 https://www.w3.org/TR/turtle/.
33 https://json-ld.org/.
34 http://webdatacommons.org/webtables/goldstandardV2.html.
35 http://wiki.dbpedia.org/data-set-2014
36 http://commoncrawl.org/.

Listing 13: The value of the edit distance for cell content “mount everest”.

```
milton everest 5
my everest 4
mont everest 1
mount everest committee 10
```

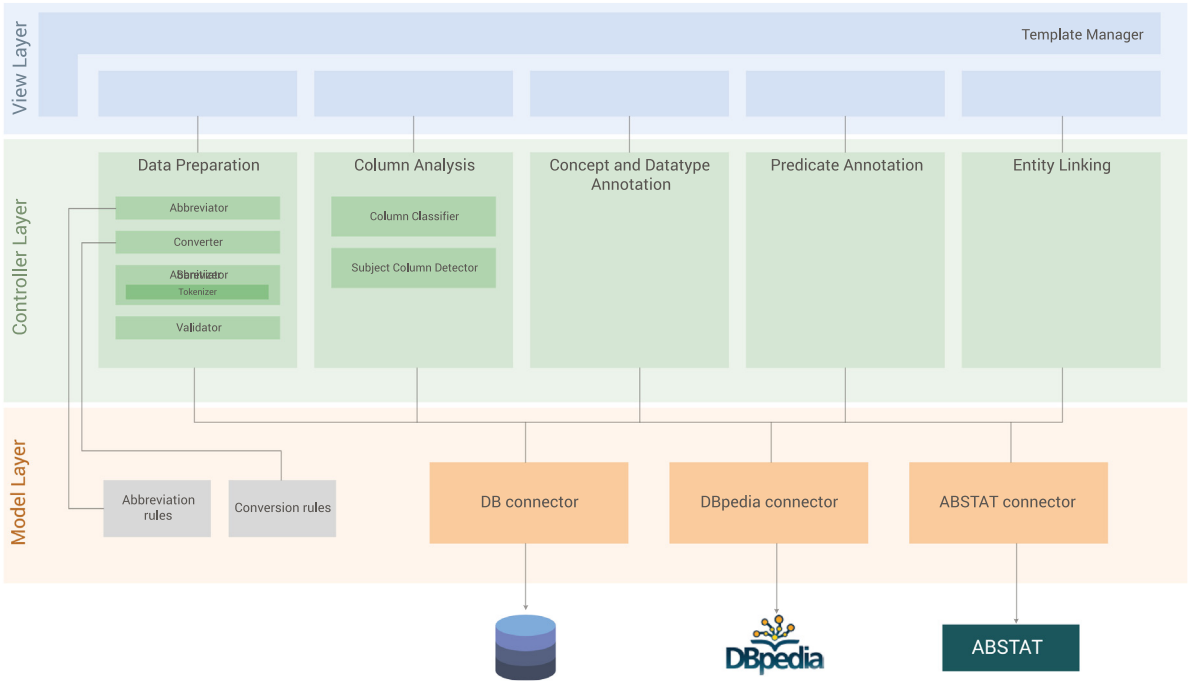

Fig. 7. Architecture of MantisTable tool.

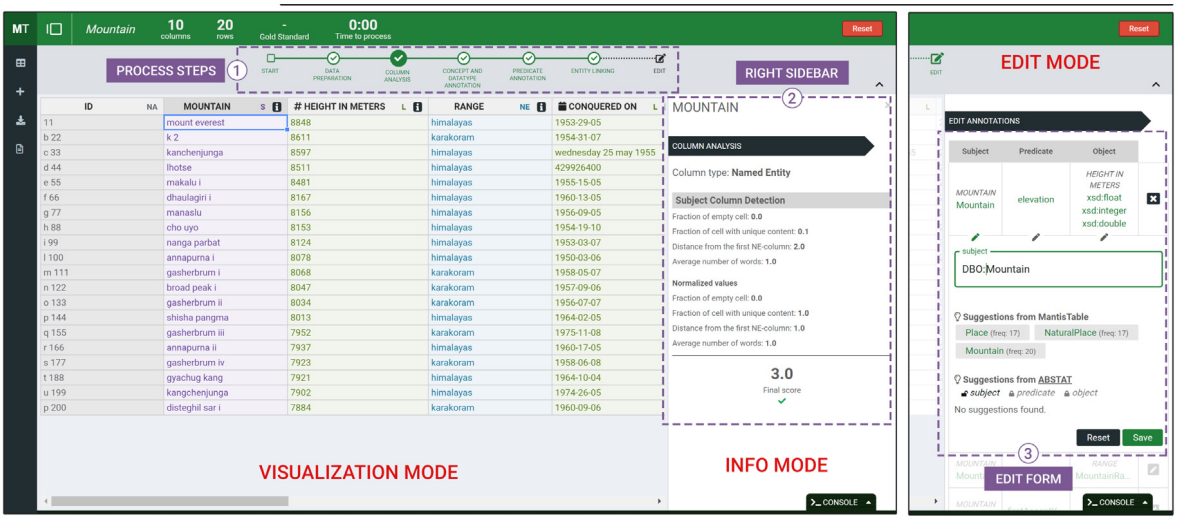

Fig. 8. MantisTable interface: table view.

Table 10
Characteristics of the Gold Standards.

	Table	Columns				Rows				Structuredness	Columns			Concepts	Pred.
		total	min	max	avg	total	min	max	avg		S	NE	L		
T2Dv2	234	1157	1	13	4	27,966	5	585	119	0.92	231	–	–	39	154
Limaye200	200	919	2	11	4	4036	3	102	20	0.97	200	504	216	84	–

total number of annotated tables is 434. The characteristics are summarised in Table 10.

In addition, we extend the experiments and run MantisTable on additional tables proposed by the challenge "Tabular Data to Knowledge Graph Matching".

4.2. Evaluation measures

To measure the effectiveness of the annotation process, we adopt the metrics proposed in the challenge.[37] Precision P of the mapping between the table data and the KG is calculated using the following formula:

$$P = \frac{|PA|}{|SA|} \tag{8}$$

where a perfect annotation PA refers to the annotation returned by the STI process which corresponds to the annotation of the Gold Standard. A submitted annotation SA refers to the annotation returned by the STI process.

Recall R is calculated as follows:

$$R = \frac{|PA|}{|GA|} \tag{9}$$

where the number of ground truth annotations GAs correspond to the number of annotations in the Gold Standard. Finally, we combine the predefined measures through the F-measure (F1), which represents the harmonic mean between precision and recall.

4.3. Experimental settings

Approaches and implementation choices. MantisTable is evaluated against two state-of-the-art approaches and a baseline as follows:

1. the two state-of-the-art approaches are [29] and [11]. The approach described in [29] applies the T2Dv2, while [11] uses Limaye200;
2. a further comparison is made with the results obtained from a Baseline semantic annotation method defined below.

The approach described in [11] could not be directly tested, as no runnable code was available at the time of conducting the evaluation,[38] as reported also in [19]. The data shown in the comparison tables are as a result of a new implementation of this approach following the described approach in the related paper.[39]

Baseline method. The Baseline method runs the Subject Column Detection phase as defined for MantisTable, while it uses a different method for the Concept Annotation, Datatype Annotation and the Predicate Annotation phase. For the Concept Annotation, in the Baseline we use the content of the cells to look for the corresponding entities within the KG, using a SPARQL query. The retrieved entity is the one that has the value of the rdfs:label equivalent to the exact value of the cell. Subsequently, for the annotation of the *NE-columns* with a semantic concept, the candidate with the highest number of occurrences is selected. The Predicate Annotation follows a similar procedure: the candidate relationships are retrieved by looking for the predicate which exists between the entities identified in the *S-column*, and the value/entity in the same row of the other columns. The most frequent predicate is chosen for the final annotation.

Table 11
Results of the subject column detection step. The precision is given as #correct (TP)/#total.

Method	T2Dv2
Zhang et al. [11]	0.87
MantisTable	0.98
MantisTable without fraction of empty cells feature	0.9444
MantisTable without fraction of cells with unique content feature	0.9401
MantisTable without distance from the first NE-column	0.9444
MantisTable without average number of words in each cell	0.93

Configuration of the hardware. All experiments have been performed on a Linux machine with 2 cores (2.3 Ghz) and 8 GB RAM. As KG the local replica of the online version of DBpedia 2017 is used.[40]

Parameters settings. In our approach, we consider different parameters. In order to define the final values for these parameters several experiments on T2Dv2 GoldStandard have been performed. Fig. 9 shows the variation of the precision when the value of k (the number of cells considered in the Concept Annotation) changes. As it is shown, the maximum Precision is obtained for a value of k equal to 30. The second analysed parameter analysed is γ, which represents the threshold relating to the number of occurrences of a *Regextype* in order to annotate a column as *L-column* in the Column Analysis phase. Experiments have shown that when at least 50% ($\gamma = 0.5$) of the values of the cells belong to the same *Regextype* then the Precision and Recall reach the maximum possible value as shown in Fig. 10. The parameters δ (the threshold related to the maximum global frequency of candidate concepts), and β (the threshold related to the number of cells that can be associated with that particular concept), are used for filtering candidate concepts during the Concept Annotation and Datatype Annotation phase. Experiments have shown that the best results are obtained with $\delta = 0.5$ and $\beta = 0.4$ as reported in Fig. 11.

The last parameter to discuss is the *econf* threshold (Formula (4)), which refers to the number of words in the name of the candidate entity. Considering the way our score is calculated, *econf* will be lower if the entity name has only one word and will be higher if the entity name is composed of two or more words. At the same time, entities with more than 2 words will get a global *econf* score similar to the one with 2 words because the context will get a higher weight. In order to determine the best *econf* threshold for entities with more than two words we analysed the worst performing table without using any threshold. The T2Dv2 table considered is about Swimmers[41] and only one candidate entity per column has a corresponding DBpedia entity. With this analysis we determined that cells containing name and surname should both simultaneously find a match with the corresponding entities of DBpedia in order to be a good candidate for the annotation. Therefore, the threshold we considered for entities composed of 2 or more words is 2.4. The best table to determine a valid threshold for one word entities in T2Dv2 is a column containing countries (they contain one word cell for most of the cells in the columns).[42] In this way we determined that all the candidate entities with an *econf* score lower than 1.5 are not correctly assigned, thus, these cells should not be considered in the final annotation.

37 https://www.aicrowd.com/challenges/iswc-2019-column-type-annotation-cta-challenge.

38 As of July 2019.

39 https://bitbucket.org/disco_unimib/tableminer-imp/.

40 https://dbpedia.org/sparql.

41 T2Dv2 table index: 1438042986423 95 20150728002306-00329-ip-10-236-191-2_805336391_10 - http://webdatacommons.org/webtables/goldstandardV2.html.

42 T2Dv2 table index: 24859353 0 7027810986004269522 - http://webdatacommons.org/webtables/goldstandardV2.html.

Table 12
Problem on subject column detection step.

Table name	GS	Example of the row	Topic	Comments
12125836_0_1134348206297032434	T2Dv2	AM 900; KALI AM; Spanish News/Talk; 747 E Green St, Pasadena 91101; (626) 844–8882	Radio	In this table there exist different columns with unique values (e.g. radio names and the respective frequency for that radio channel). Thus it is difficult to discriminate which is the *S-column* among columns with unique values
6310680_0_5150772059999313798	T2Dv2	Afghanistan; Afghani; AFA; 4	Currency	Possible T2Dv2 error
80184932_0_240003884724905487	T2Dv2	Diversified Industrials; 3M Co.; Solutions videos, demo videos	Companies and Industries	The two NE-columns are very similar to each other

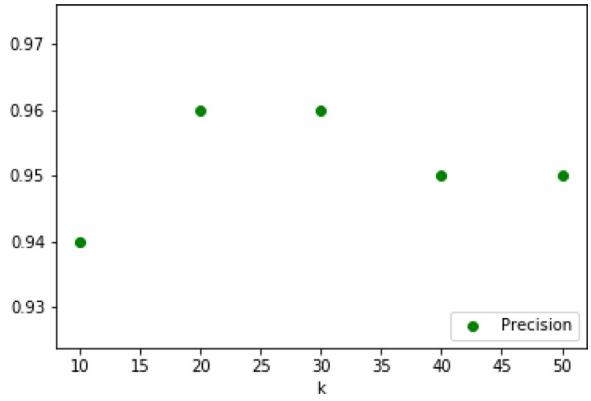

Fig. 9. Precision as the value of k changes.

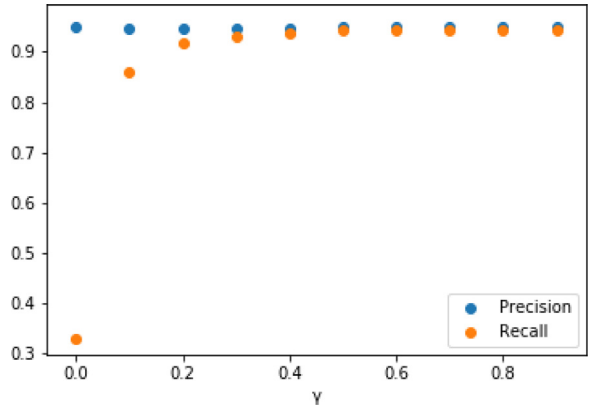

Fig. 10. Precision and Recall as the value of γ changes.

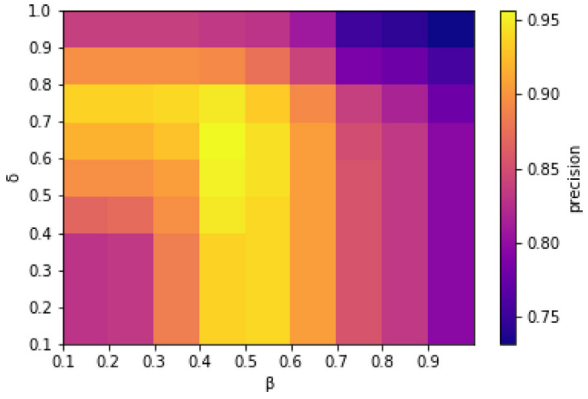

Fig. 11. Precision of the approach when the δ and β value change.

4.4. Evaluation results and discussion

Subject Column Detection. Table 11 shows the evaluation of the Subject Column Analysis phase. We test MantisTable for the Subject Column Detection with four different configurations: (i) with fraction of empty cells, fraction of cells with unique content, distance from the first *NE-column*, and the average number of words in each cell, (ii) all above without fraction of empty cells feature, (iii) all in (i) without fraction of cells with unique content feature, (iv) all in (i) without the average number of words in each cell. As it can be noticed from the results, we outperform the best performing approach so far [11], which uses additional features such as the acronym and the out-table context.

However, there can still be problems to reach a precision equal to 1 as shown by the examples in Table 12. In general, the problems regard the disambiguation of entities containing unique values, or two different columns with very similar entities. For instance, the table in T2Dv2, in the topic of Radio, contains values such as AM 900 and KALI AM where both the frequency and the name of the radio station are unique.

Concept Annotation and Datatype Annotation. The table-to-concept annotations of the T2Dv2 Gold Standard can be used to evaluate our Concept Annotation and Datatype Annotation phase. This association is based on the assumption that the annotation made by the technique is considered valid, according to the Gold Standard, if the table-to-concept annotation of a specific table corresponds to the concept annotation made for the *S-column* of the same table.

The first problem in our approach for this phase is due to the cumulative frequency usage. For instance, table 48456557_0_3760853481322708783 (T2Dv2) contains animal of various species including frogs (e.g. Bullfrog), deers (e.g. Elk) and birds (e.g. Gray Catbird). Our method identifies the concept with the greater cumulative frequency as the winning concept (in this case the concept Bird). We cannot apply this strategy straightforwardly to those columns containing heterogeneous concepts. However such cases are rare within the Gold Standards. To resolve this issue we considered changing the strategy by considering the common ancestor of all concepts of the column under investigation. This strategy cannot be generalised for all tables as sometimes the right concept for the annotation is the most specific one. Another problem is when annotating tables that have no corresponding entities in the KG. For a concrete example for the last problem please refer to table 1438042989018_40_20150728002309-00067-ip-10-236-191-2_57714692_2 (T2Dv2) containing the surnames of cricket players (entities that are not popular).

Table 13
Results of the concept annotation and datatype annotation.

Method	T2Dv2			Limaye200		
	P	R	F1	P	R	F1
Zhang et al. [11]	0.9	0.89	0.9	0.63	0.56	0.59
Ritze et al. [12]	0.89	0.88	0.89	0.51	0.26	0.35
Baseline	0.83	0.45	0.58	0.68	0.60	0.63
MantisTable	0.95	0.95	0.95	0.77	0.93	0.84

Table 14
Results of the predicate annotation for *NE-column*.

Method	T2Dv2		
	P	R	F1
Zhang et al. [11]	0.43	0.31	0.36
Ritze et al. [12]	0.23	0.73	0.35
Baseline	0.49	0.38	0.43
MantisTable	0.57	0.45	0.51

Table 15
Results of the predicate annotation for *L-column*.

Method	T2Dv2
	P
Neumaier et al. [4]	0.32
MantisTable without BKG	0.16
MantisTable with BKG	0.43

Table 13 shows the results of precision (P), recall (R), and F-measure (F1) for Concept Annotation and Datatype Annotation.

Predicate Annotation. The T2Dv2 Gold Standard attribute-to-property annotations can be used for the Predicate Annotation evaluation. Table 14 shows the results of Predicate Annotation, respectively for the Baseline, T2K [29] and MantisTable. While we obtain better results for the property annotation between two *NE-columns*, lower results are obtained for the predicate annotation between a *NE-column* and a *L-column*. As shown in Table 15, all the approaches get lower results for *L-columns* (having numerical values). The reason is that connecting an entity to a literal through a predicate is not straightforward. Context is needed in order to better disambiguate literals.

4.5. Challenge evaluation results

In order to evaluate our approach on a larger dataset and compare with other state-of-the-art approaches we participated in the Semantic Web Challenge on Tabular Data to KG Matching. The challenge is composed of three tasks:

1. Assigning a semantic type (e.g. a KG class) to an (entity) column (CTA task);
2. Matching a cell to a KG entity (CEA task);
3. Assigning a KG property to the relationship between two columns (CPA task).

The challenge has four rounds and the datasets were generated as follows:

- Round 1 (sandbox): extended T2Dv2 dataset;
- Round 2 (fine-tuning): Wikipedia tables dataset + automatically generated dataset;
- Round 3 (limited tests): automatically generated dataset;
- Round 4 (limited tests): automatically generated dataset with only hard cases.

Table 16 shows the characteristics of the challenge datasets.

Table 16
Number of tables, concepts, entities and predicates for the challenge dataset.

Round	# table	CTA	CEA	CPA
1	64	120	8418	116
2	11,924	14,780	473,796	6762
3	2161	5762	406,827	7575
4	817	1732	107,352	2747

Excluding the dataset for the first round and part of the second, for the other rounds the dataset was automatically generated by executing SPARQL queries to collect concepts and predicates. For each concept, some predicates have been selected in order to create SPARQL queries that produce "realistic" looking tables (Raw Table Generation). In the last step, some instance values can be replaced in a rule-based fashion (e.g. first names of person entities can be abbreviated, synonyms can be used, the precision of numerical values can be adjusted and full dates can be replaced with months/years). Tables or rows/columns too "easy" for the annotation (e.g. through exact match) have been dropped.

The main issues faced in this challenge were: (i) tables with few rows (1–2) are sometimes difficult to annotate; this may be addressed by trying to integrate some other data sources by querying them with content from the table (maybe considering Wikidata); (ii) some target columns are very complex to annotate because the cell contents cannot be directly linked to entities in the KG, because of some target columns probably not having been generated correctly in the challenge's Gold Standard; (iii) tables about people with only surnames are frequently linked to homonym entities, this might be solved using a novel technique for the people names.

MantisTable has been constructed using the data in the T2Dv2 dataset thus resulting to have great results (top 2 performer) in Round I [31]. In Round 2 and Round 3 we performed well in the CTA task (4th out of 8 systems) but the results were not excellent for the other tasks (on average 7th out of 8). Finally, in Round 4 we were once again the top 2 performer on the CEA task thanks to the Improvements made in the detection of the "first letter of name, surname" format for dbo:Person entities and also increased the performance of the other tasks as well (4th out of 7 systems in both CTA and CEA, close to 3rd position). For brevity, we provide the link[43] to the complete results of the challenge. The advantage of the two best approaches of the challenge, Mtab [32] and IDLab [33], was the employment of aggregation of different data sources (e.g. DBpedia lookup, Wikidata, DBpedia spotlight). The former used a custom pipeline with multiple probability estimation steps, while the latter used Shannon's entropy to estimate the correct candidate. A limitation of Mtab was the particular slowness because, according to the authors, the tool was not yet production-ready, so the practicality of the approach has yet to be demonstrated. IDLab was less computationally expensive but was depending on many different external services. Three other approaches (ADOG [34], Tabularisi [35] and LOD4ALL [36]) used ElasticSearch to store the entity labels and searched them with a given tolerance (edit distance). This should be considered a good practice because SPARQL will not allow the same expressiveness but will save much computation time. An overview of the challenge results is shown in Appendix B.

Many of the above approaches base their operation on external lookup services (e.g. DBpedia Lookup, Wikidata API, DBpedia Spotlight). In particular, such functionality is adapted for the identification of the candidate entities. This implies two major problems: (i) the need for these services to be always available, and (ii) the accuracy of the results depends strictly on the quality

[43] https://www.cs.ox.ac.uk/isg/challenges/sem-tab/2019/results.html.

of these services. One of MantisTable's strengths is the ability to identify candidate entities without the use of external services.

4.6. STILTool

In the state-of-the-art there are only two scripts[44] to automate the evaluation of annotations. For this reason we developed STILTool [37],[45] which helps to evaluate annotations results from different approaches with T2Dv2 and Limaye200 Gold Standard tables. The tool takes a JSON file with the annotated table that can be downloaded from the MantisTable tool via the download annotation feature. The input file contains information about the tables of the Gold Standard and the annotations that MantisTable obtained on various tables. The output returned by STILTool is an analysis of the results that evaluate the correctness of the annotations by calculating evaluation metrics. In particular, a comparison of MantisTable annotations with respect to the Gold Standard is returned. STILTool has been developed in NodeJs and Meteor. The code is open-source and freely available at a Git repository.[46]

5. Semantic Table Interpretation: State-of-the-art

This section gives an overview of state-of-the art approaches (Section 5.1) and tools (Section 5.2) on the STI. More than 50 papers collected from different sources have been studied in order to get a comprehensive overview of the existing approaches.

5.1. Approaches supporting STI

For the analysis of the state-of-the-art, 18 approaches have been selected as representative because they are complete with respect to the workflow of STI and are thus considered the front line for comparison. To begin with, we give a short overview of the approaches and analyse them considering 4 criteria as shown in Table 17. The first criterion, according to which we organise this section, is the learning technique: unsupervised and supervised. While the former category does not rely on labelled input data, the latter is only applicable for labelled data. In our analysis, the second criterion refers to the completeness with respect to the STI workflow. The third and the fourth criterion refer to the publication year and to the KG that these approaches use to annotate tables.

Approaches such as [7–10,19–21] use supervised machine learning techniques to label new sources of structured data with the support of training data which have been previously labelled manually. Such approaches use machine learning techniques in which a function of inference is created [8,10,19–21] or a classifier is trained with features corresponding to similarity metrics [7,9]. Supervised approaches are sometimes more effective than the others, as a set of training data is enough to label new tables covering specific domains. However, they can be limited in this aspect for three reasons: (i) it requires a set of training data that should be provided, (ii) such training set may not be available thus forcing the user to create it by means of a manual labelling, and (iii) the results are comparable to the ones achieved by unsupervised approaches with respect to the quality of the annotations. To avoid these limitations, MantisTable uses unsupervised learning techniques, so the approach can be applied to general purpose domains.

The remaining approaches apply unsupervised techniques. Approaches such as [15–18] use custom or personalised KGs, while the rest use Open Source KGs available on the Web. In [16] authors focus on Web tables and synthetically define the process of semantic interpretation of a table as finding its correct positioning within a taxonomy (hierarchical organisation of knowledge). Therefore, the approach tries to associate a specific table with one or more concepts contained in a KG. In particular, after establishing such associations, each row of the table describes the attributes of an entity which has as type a concept in a custom version of Probase (a general purpose taxonomy). For each concept, Probase returns a list of entities associated with a set of scores (plausibility and ambiguity). In order to select eligible entities of a class, authors score and merge candidate attributes and choose the top-ranked.

[18] makes custom queries to Wikitology (a hybrid knowledge base of structured and unstructured information extracted from Wikipedia, augmented by RDF data from DBpedia and other Linked Data resources) using the information inside the rows of a table. This approach infers automatically a (partial) semantic model for the tables using the information in the headings and the information stored in the table cells. Moreover it offers the possibility to export the data in the table as linked data.

[17] uses a majority-rule mechanism, where a potential concept that should be annotated to a specific column is selected because it is the one that occurs more often in the cells of the same column. A characteristic of this approach is that the search for the relevant semantic labels is performed directly in the Web rather than from a predefined KG. The subject column must not necessarily be a key to the table and may contain duplicate values. Moreover, it is possible that the subject column is represented by several columns of the same table. This principle can be ineffective in many aspects, for example the presence of general concepts (e.g. music) can be often combined with different textual content, as well as the consideration of a KG that does not have a hierarchical classification.

For the STI process, [15] uses queries and ranking metrics to generate an initial set of concepts, predicates and entities to be assigned as the headers of the columns, the content of the cells and the relationships between the columns. This approach uses the data in DBpedia, Yago and Wikitology. As for the columns, two sets of candidate concepts are generated (one for DBpedia concepts and the other for Yago concepts), which are formed by the union of the concepts retrieved from the queries performed with the values of the cells belonging to each column. The set of relationships to be considered when annotating the semantic connections between the columns is obtained by the union of the set of candidate relations for each pair of cells within each row of the table. Differently from the above approaches, MantisTable uses an Open Source KG freely available on the Web. Moreover, querying KG is onerous and our approach reduces the number of queries, as it selects only a limited number of rows in a table. Finally, MantisTable generates context utilising the elements inside the table to discriminate the entities and to create high quality annotations.

Approaches such as [6,11–16,18,38] and [1] are unsupervised approaches that use Open Source KGs for the annotation.

[1] uses a comprehensive strategy for semantic annotation which examines the entire content of the table (for example, the classification of a column depends on all the cells in that column) through the application of a probabilistic graphic model. Such a model has been identified by [15] as too expensive in terms of computational effort, thus the authors propose an alternative Semantic Message Passing Algorithm that applies the same type of joint inference to a similar light-weight graphic model. Also, the approach takes into consideration semantic relations

44 https://github.com/olehmberg/winter - https://github.com/sem-tab-challenge/aicrowd-evaluator.

45 http://zoo.disco.unimib.it/stiltool/.

46 https://bitbucket.org/disco_unimib/stiltool.py/.

Table 17
Approaches supporting STI.

	Year	STI workflow									Learning technique	KG
		Data Prep.	Column type analysis			Concept and datatype annotation			Predicate Ann.	Entity link.		
			NE	L	S	NE	L	S				
Kruit et al [38]	2019	–	✓	✓	✓	–	–	–	✓	✓	Unsup	DBpedia
Chen et al [19]	2019	–	✓	–	–	✓	–	–	–	✓	Sup	DBpedia
Takeoka et al [21]	2019	–	✓	✓	–	✓	✓	–	–	–	Sup	Wordnet
Zhang et al [20]	2018	–	–	–	✓	–	–	✓	–	–	Sup	Wikipedia
Zhang et al [11]	2017	–	✓	✓	✓	✓	✓	✓	✓	✓	Unsup	DBpedia
Efthymiou et al [6]	2017	✓	✓	✓	✓	✓	✓	✓	✓	✓	Unsup	DBpedia
Pham et al [7]	2016	–	✓	✓	–	✓	✓	–	✓	–	Sup	Domain independent
Taheryian et al. [8]	2016	–	✓	✓	–	✓	✓	–	✓	–	Sup	Domain independent
Ritze et al. [12]	2015	✓	✓	✓	–	✓	✓	–	✓	✓	Unsup	DBpedia
Ramnandan et al [9]	2015	–	–	✓	–	–	✓	–	–	–	Sup	Domain independent
Deng et al [13]	2013	–	✓	✓	–	✓	✓	–	–	–	Unsup	Freebase, Yago
Quercini et al [14]	2013	✓	✓	✓	–	✓	✓	–	–	✓	Unsup	DBpedia
Mulwad et al [15]	2013	✓	✓	✓	–	✓	✓	–	✓	✓	Unsup	DBpedia, Yago, Wikitology
Wang et al [16]	2012	✓	✓	–	–	✓	–	–	✓	✓	Unsup	Enriched Probase
Knoblock et al [10]	2012	–	✓	✓	–	✓	✓	–	✓	–	Sup	Domain independent
Venetis et al [17]	2011	–	✓	✓	✓	✓	✓	✓	✓	–	Unsup	isA, relation database
Syed et al [18]	2010	–	✓	✓	–	✓	✓	–	✓	✓	Unsup	Wikitology
Limaye et al [1]	2010	–	✓	–	–	✓	–	–	✓	✓	Unsup	Yago
MantisTable	2019	✓[a]	✓[a]	✓[a]	✓[b]	✓[b]	✓[a]	✓[b]	✓[b]	✓	Unsup	DBpedia

[a]Refers to the adaptation and improvement of the technique which has been already proposed by state-of-the-art approaches for that particular phase of STI.
[b]Refers to the proposal of novel techniques for that particular phase of STI.

among columns as [1] does, but in contrast it takes into account both the headers of the columns and the entities within the rows. [11] uses a similar Semantic Message Passing Algorithm as [15]. Moreover, [11] includes steps such as the identification of the *S-column*, the *L-column* annotation, the analysis of a sample data extracted from the entire data source to reduce computational effort, etc. [6] proposes different unsupervised methods for matching entities of a table to entities of a KG on the Web. The similarity between entities is computed as the cosine distance between their vector representations. Such approaches work well with table contents alone, without relying on any metadata but cells often lack entity correspondences, thus resulting in a decrease of their performance. The annotation process proposed by [12] uses similarity metrics for the creation of candidate concepts for the semantic annotation. Such candidate concepts are then sorted accordingly to the principle of weighted majority voting, where the weights are based on the matching scores, calculated between the values in the table and the entities in the KG associated with them. [13] offers a scalable and efficient solution for determining concepts associated with each *NE-column* within a table, using a MapReduce algorithm with two supporting techniques: knowledge concept aggregation and knowledge entity partition. These techniques allow the identification of top k candidate concepts for the *NE-column* of the table. In contrast to other approaches, fuzzy matching algorithms calculated between the values in the table and the entities of the KG, do not compromise the efficiency of the algorithm itself. The fuzzy matching and the ordering of the obtained results are based on generic similarity functions thus such algorithm obtains better results in particular with respect to [17]. [14] proposes a mechanism for the annotation of cell value with entities that are not present in a KG, through Web searching.

Finally, the goal of [38] is to complete the KG with the information in the Web tables. The approach uses a Probabilistic Graphical Model which considers first the label similarity and then updates the likelihood score to maximise the coherence of entity assignments across the rows using Loopy Belief Propagation (LBP). Unlike other approaches, for entity matching the authors compute coherence as a combination of properties that are shared by the entities in the table and do not use class membership. If the label matching is not sufficient, the approach makes use of embeddings of KG entities. This feature helps the approach to identify novel facts for KG completion. Similarly, MantisTable also uses Open Source KG. Anyhow, the performance of the MantisTable approach is compared to just some of the approaches in the state-of-the-art because (i) the code is not always available, and (ii) when available, it is difficult to be executed. Moreover, the embeddings cannot be always utilised as tables have often missing values.

Considering the annotation phases as in Table 17, [6] is the only approach that applies all the predefined annotation phases for the STI. Most of the approaches do not perform the Data Preparation step but only [6,12,14,15] and [16] do. Most of the approaches perform column type analysis and annotation for both *NE* and *L-columns*. Even though the approach in [8] does not identify the *S-column*, it supports the annotation of the relations between columns. Differently from all the approaches, MantisTable performs the phases of STI proposing novel techniques in order to improve and provide high quality annotations.

Approaches such as [13,14,16,17] mainly focus on the analysis of Web tables, thus limiting their range of actions to the content of Web pages. Such approaches use the information offered by Web tables (e.g. Web page title, table caption, or surrounding text) in the semantic annotation process, while approaches such as [1,7–9,11,12,15,18] can rely on the data in the table. Moreover, considering only Web tables excludes the possibility of analysing data sources that contain a large number of tuples, thus ignoring

Table 18
Tool comparison.

Tool	Working and available	Ontology import	Annotation method	Auto-complete support (if manual)	Annotations				Entity link.	Export
					Col.			Pred.		
					S	NE	L			
Datagraft	✓(online)	✓	manual	✓ with ABSTAT	✓	✓	✓	✓	–	–
Karma	✓(installer)	✓	semi-auto	–	✓	✓	✓	✓	–	–
Odalic	✓(docker)	✓	auto	–	✓	✓	✓	✓	✓	✓
Open Refine	✓(exe)	–	semi-auto	–	–	–	–	–	✓	✓
STAN	✓(online)	–	manual	✓ with DBpedia	✓	✓	✓	✓	–	✓
TableMiner+	–	–	auto	–	✓	✓	✓	✓	✓	?
MantisTable	✓(online)	–	auto	–	✓	✓	✓	✓	✓	–

the problem of performance and execution time. MantisTable does not consider only Web Tables but also other kinds of tables.

5.2. Comparison between semantic interpretation tool

Although there is a large number of approaches in the STI state-of-the-art, the number of available and working tools is small. The purpose of the analysis presented in this section is to analyse the current tools available for the STI process, or more generally for a structured data source.

DataGraft[47] [22] is a cloud-based service, which provides an integrated Web environment for data hosting (linked data and file storage, dataset sharing, data querying) and data transformations (interactively building, modifying, and sharing of repeatable and reusable data transformations). DataGraft considers CSV files as input. The Grafterizer tool inside DataGraft provides an interface to transform tabular data into RDF triples. The semantic annotation needs to be done manually by users; the tool only provides a feature to auto-fill the concept and datatype labels using ABSTAT [28] summaries.

Karma[48] [23] is an Open Source information integration tool that allows users to integrate data coming from different sources such as databases, spreadsheets, delimited text files, XML, JSON, KML and Web API. The graphical interface provides an easy way to transform data in order to normalise, restructure and express them in different forms. The graphical interface, in addition, is built to help users to integrate information by modelling it according to one or more ontologies. Karma learns to recognise the mapping of data to ontology classes and then uses the ontology to propose an automatically generated model that ties together these classes. This model can be adjusted by the user. Once the model is complete, the integrated data can be published as RDF or stored in a database.

Odalic[49] [24] is a tool that interprets tabular data and publishes them as Linked Data. Odalic annotates columns using a KG, and links cell values to the entities of a KG. It takes as input CSV files and one or more KG. The Odalic Server wraps the TableMiner+ algorithm [11]. Along the process, users can provide feedback anytime since they can delete the suggested class/property for the annotation and propose a new one, which is then added to the KG. In Odalic, users can manually specify multiple *S-columns* and it supports any knowledge base which is accessible via SPARQL query language and it also supports PoolParty KG.[50] The result of the STI process could be exported in several formats including an extended CSV version and an RDF dataset.

OpenRefine[51] is a tool for cleaning, transforming and extending messy data. It can perform a semi-automatic reconciliation process against any database that exposes a Web service using Reconciliation Service API[52] specification or a SPARQL endpoint. If a cell has multiple entity candidates the user needs to pick the correct one manually. To improve the quality of the matches, a class for the rows can be selected in order to restrict the matches only to items which are instances of any subclass of the given class. In addition, the reconciliation interface can be configured to take into account other dataset's columns in the matching scores. OpenRefine functionalities can be extended by installing extensions; moreover there are other distributions of the tool that have been customised for a specific usage or integration with other technologies (e.g. LOD refine).

STAN is an online tool that semantically annotates tables from popular KGs. Like DataGraft, it uses a self-completion API, offered by ABSTAT [28], for the properties and concepts contained in a KG. It considers CSV files as input. The *S-column* must be set manually through a window in which the user is able to specify the concept associated with it. If this concept refers to one of the KGs in ABSTAT, an auto-completion system helps the user to fill in the form correctly. A similar process is also applied for the annotation of predicate columns. Furthermore, it is possible to define the object that specifies the type to be assigned to the column as a datatype or as another ontological concept.

The TableMiner+ [25] tool refers to the homonyms approach [11]. It supports only Web tables for which the user must provide a URL. The whole process is run in batch and users can be notified with an email message when it finishes. The annotation is provided in JSON format, which is interpreted and displayed using an annotated table and a graph visualisation module. Both visualisation components are interactive to allow users feedback and customisation of the output. However, testing TableMiner+ was not possible because, although the code for the tool was available, it could not be run because of interdependencies in the code.

Table 18 allows to compare the previous tools on the basis of 7 criteria; the availability of the tool, the possibility to import the ontology, the annotation method, the availability of the auto-complete module in the case the annotation method is manual, the STI phases and the possibility to export the results.

Regarding the possibility of defining an ontology for the STI, only Karma, Odalic and Datagraft allow to import one or more and combine them, while the others use only predefined KGs. In order to prepare data, Karma, OpenRefine and Datagraft give users the ability to manipulate tables and refine them by allowing column modification such as renaming, eliminating, or changing their order. Moreover OpenRefine has features that are not common

47 https://datagraft.io/.
48 https://usc-isi-i2.github.io/karma/.
49 http://github.com/odalic/github.com/odalic/github.com/odalic/.
50 https://www.poolparty.biz/.
51 http://openrefine.org/.
52 https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API.

Table 19
Mathematical notation lookup table.

	Definition	First defined in
(i, j)	Cell of the table T	Section 1 Fig. 1
$co \in CO$	co is the concept. CO is the set of concepts.	Section 2
$c_j \in C$	c_j is the jth column. C is the set of columns of the table T	Section 1 Fig. 1
$e_{i,j} \in E_{i,j}$	$e_{i,j}$ is the entity for cell (i,j). $E_{i,j}$ is the set of entities for cell (i,j) extracted from E	Section 2
$h_j \in H$	h_j is the header of the jth column. H is the set of headers of the table T	Section 1 Fig. 1
KG	Knowledge graph	Section 1
KGs	Set of knowledge graphs	Section 1
$r_i \in R$	r_i is the ith row. R is the set of rows of the table T	Section 1 Fig. 1
T	Table	Section 1 Fig. 1
w	Word	Section 2 Formula 2
$x_{i,j} \in X_{i,j}$	$x_{i,j}$ is the context for cell (i,j). $X_{i,j}$ is the set of context for cell (i,j)	Section 2 Formula 2
$bow(\cdot)$	Return bag-of-words representation (multiset) of text with the occurrence of words	Section 2
$bowset(\cdot)$	Return the set of unique tokens of $bow(.)$	Section 2 Formula 2
$dice(w, \cdot))$	Return Dice similarity between w and .	Section 2 Formula 2
$econf\ (e_{i,j})$	Return confidence score for entity $e_{i,j}$	Section 2 Formula 4
$econtext(e_{i,j})$	Return entity context score for entity $e_{i,j}$	Section 2 Formula 2
$pconf(p_j)$	Return confidence score for entity p_j	Section 2 Formula 7
$pcontext(p_j)$	Return predicate context score for predicate p_j	Section 2 Formula 6
$tx(i,j)$	Return the text of cell (i,j)	Section 2

Table 20
Results of the challenge: Round 1 (a) and Round 2 (b).

(a)

Team	CEA		CTA		CPA	
	F1	P	F1	P	F1	P
MantisTable [31]	1	1	0.929	0.933	0.965	0.991
MTab [32]	1	1	1	1	0.987	0.975
Tabularisi [35]	0.884	0.908	0.825	0.825	0.606	0.638
IDLab [33]	0.448	0.627	0.833	0.833	–	–
ADOG [34]	0.657	0.673	0.829	0.851	–	–
LOD4ALL [36]	0.852	0.874	0.85	0.85	–	–
DRAGOBAH [39]	0.897	0.941	0.644	0.581	0.415	0.347

(b)

Team	CEA		CTA		CPA	
	F1	P	AH	AP	F1	P
MTab [32]	0.911	0.911	1.414	0.276	0.881	0.929
IDLab [33]	0.883	0.893	1.376	0.257	0.877	0.926
Tabularisi[35]	0.826	0.852	1.099	0.261	0.790	0.792
MantisTable [31]	0.614	0.673	1.049	0.247	0.460	0.544
ADOG [34]	0.742	0.745	0.713	0.208	0.459	0.708
LOD4ALL [36]	0.757	0.767	0.893	0.234	0.555	0.941
DRAGOBAH [39]	0.713	0.816	0.641	0.247	0.533	0.919

to others in our analysis, such as the automatic creation of new columns, the exploration of the cells through facets, which allows to compare the values on the basis of a chosen constraint, and the use of a clustering feature that takes into consideration groups of cells.

While for the STI tasks TableMiner+ and Odalic are based on an automatic process, Karma and Open Refine require user interaction. Datagraft and STAN, on the other hand, support only manual annotation although they provide an auto-complete service using ABSTAT. STAN uses the auto-completion feature both for the annotation of the *S-column* and properties of the other columns if the class which is being examined refers to a class in DBpedia. All tools, except for OpenRefine, provide an annotation for the *S-column*, the *NE-columns*, the *L-columns* and the relationships between them. Karma, Odalic and TableMiner+ offer a list of suggestions for the correct semantic annotation that users can select in order to adjust the tool output. The linking of the cells with specific entities within the KG is performed by TableMiner+, Odalic, OpenRefine and Datagraft. Furthermore, Tableminer+, Odalic and Karma offers a graphical representation of the semantic mapping. Except TableMiner+, all the other tools have the automatic saving feature.

Regarding the export of mappings, Karma and Odalic allow the export in RDF format or in JSON-LD while STAN in RML format. In order to export the tabular data, Karma uses the R2RML format, which allows to highlight the association between table and the ontology. STAN converts the tabular data into RDF triples while OpenRefine into JSON, YAML, RDF, and others.

Finally, an interesting feature is the use of APIs which allows the integration of external services. These services are present in Karma, OpenRefine and in particular in STAN, where they allow two different annotation services, publicly accessible through HTTP GET and POST operations.

6. Conclusion and future works

This work presents MantisTable, an STI approach that overcomes the limits of the state-of-the-art approaches by (i) providing a comprehensive approach to support all annotations phases; (ii) providing an unsupervised method to annotate independent tables; (iii) generating context for disambiguation; (iv) providing a tool to support STI workflow; (v) providing a tool to support the evaluation through validation indicators; and (vi) both tools are open-source and publicly available.

The experiments have shown that MantisTable outperforms all baselines. On the two Gold Standards covering multiple domains and different table schemata, it significantly improves subject identification, concept and datatype annotation and finally predicate annotation. We have shown that our approach is has achieved good results alsoon the datasets provided by the Semantic Web Challenge on Tabular Data to KG Matching.

However, MantisTable can be improved in many ways. Our main goal is to improve the obtained results by using the analysis obtained with STILTool. STILTool allows users to analytically understand which tables are critical and thus knowing exactly where to improve. That is the case for the Limaye200 Gold Standard, since the evaluation obtained low scores due to incomplete or incorrect annotations.

We will further maintain and evolve the MantisTable tool. In particular, we plan to improve the user experience while editing the final annotations. This feature is fundamental for the quality of final results. Another improvement will regard the annotation of huge tables through the development of a clustered version to support parallel execution on different nodes. Finally, we are working on a new Gold Standard that will provide researchers with high-quality annotations of huge tables, so as to test the performance of STI tools.

Table 21
Results of the challenge: Round 3 (a) and Round 4 (b).

(a)

Team	CEA		CTA		CPA	
	F1	P	AH	AP	F1	P
MTab[32]	0.970	0.970	1.956	0.261	0.844	0.845
IDLab[33]	0.962	0.964	1.864	0.247	0.841	0.843
Tabularisi[35]	0.857	0.866	1.702	0.277	0.827	0.83
MantisTable [31]	0.633	0.679	1.648	0.269	0.518	0.595
ADOG [34]	0.912	0.913	1.409	0.238	0.558	0.763
LOD4ALL [36]	0.828	0.833	1.442	0.26	0.545	0.853
DRAGOBAH [39]	0.725	0.745	0.745	0.161	0.519	0.826

(b)

Team	CEA		CTA		CPA	
	F1	P	AH	AP	F1	P
MTab [32]	0.983	0.983	2.012	0.3	0.832	0.832
MantisTable[31]	0.973	0.983	1.682	0.325	0.787	0.841
IDLab [33]	0.907	0.912	1.846	0.274	0.83	0.835
ADOG [34]	0.835	0.838	1.538	0.296	0.75	0.767
Tabularisi [35]	0.803	0.813	1.716	0.325	0.823	0.825
LOD4ALL [36]	0.648	0.654	1.071	0.386	0.439	0.904
DRAGOBAH [39]	0.578	0.599	0.684	0.206	0.398	0.874

CRediT authorship contribution statement

Marco Cremaschi: Conceptualization, Methodology, Investigation, Software, Validation, Writing - original draft. **Flavio De Paoli:** Methodology, Supervision, Writing - review & editing. **Anisa Rula:** Methodology, Validation, Writing - original draft. **Blerina Spahiu:** Conceptualization, Investigation, Writing - original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Special thanks to Roberto Avogadro, Andrea Barazzetti, David Chieregato, Alessandra Siano and Carlo Mattioli for their support during the development of the project.

Appendix A. Mathematical notation lookup table

See Table 19.

Appendix B. Results of SemTab2019

See Tables 20 and 21.

References

[1] Girija Limaye, Sunita Sarawagi, Soumen Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proc. VLDB Endow. 3 (1–2) (2010) 1338–1347.
[2] Michael J. Cafarella, Alon Halevy, Jayant Madhavan, Structured data on the web, Commun. ACM 54 (2) (2011) 72–79.
[3] Oliver Lehmberg, Dominique Ritze, Robert Meusel, Christian Bizer, A large public corpus of web tables containing time and context metadata, in: Proceedings of the 25th International Conference Companion on World Wide Web, WWW '16 Companion, Republic and Canton of Geneva, Switzerland, International World Wide Web Conferences Steering Committee, 2016, pp. 75–76.
[4] Sebastian Neumaier, Jürgen Umbrich, Xavier Parreira Josiane, Axel Polleres, Multi-level semantic labelling of numerical values, in: Paul Groth, Elena Simperl, Alasdair Gray, Marta Sabou, Markus Krötzsch, Freddy Lecue, Fabian Flöck, Yolanda Gil (Eds.), The Semantic Web – ISWC 2016, Springer International Publishing, Cham, 2016, pp. 428–445.
[5] Fedelucio Narducci, Matteo Palmonari, Giovanni Semeraro, Cross-lingual link discovery with tr-esa, Inform. Sci. 394–395 (2017) 68–87.
[6] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, Vassilis Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: The Semantic Web – ISWC 2017, Springer International Publishing, Cham, 2017, pp. 260–277.
[7] Minh Pham, Suresh Alse, Craig A. Knoblock, Pedro Szekely, Semantic Labeling: A Domain-Independent Approach, Springer International Publishing, Cham, 2016, pp. 446–462.
[8] Mohsen Taheriyan, Craig A. Knoblock, Pedro Szekely, José Luis Ambite, Learning the semantics of structured data sources, Web Semant.: Sci. Serv. Agents World Wide Web 37–38 (2016) 152–169.
[9] S.K. Ramnandan, Amol Mittal, Craig A. Knoblock, Pedro Szekely, Assigning Semantic Labels To Data Sources, Springer International Publishing, Cham, 2015, pp. 403–417.
[10] Craig A. Knoblock, Pedro Szekely, José Luis Ambite, Aman Goel, Shubham Gupta, Kristina Lerman, Maria Muslea, Mohsen Taheriyan, Parag Mallick, Semi-Automatically Mapping Structured Sources Into the Semantic Web, Springer, Berlin, Heidelberg, 2012, pp. 375–390.
[11] Ziqi Zhang, Effective and efficient semantic table interpretation using tableminer+, Semant. Web 8 (6) (2017) 921–957.
[12] Dominique Ritze, Oliver Lehmberg, Christian Bizer, Matching html tables to dbpedia, in: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS '15, ACM, New York, NY, USA, 2015, pp. 10:1–10:6.
[13] Dong Deng, Yu Jiang, Guoliang Li, Jian Li, Cong Yu, Scalable column concept determination for web tables using large knowledge bases, Proc. VLDB Endow. 6 (13) (2013) 1606–1617.
[14] Gianluca Quercini, Chantal Reynaud, Entity discovery and annotation in tables, in: Proceedings of the 16th International Conference on Extending Database Technology, EDBT '13, ACM, New York, NY, USA, 2013, pp. 693–704.
[15] Varish Mulwad, Tim Finin, Anupam Joshi, Semantic message passing for generating linked data from tables, in: Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, Krzysztof Janowicz (Eds.), The Semantic Web – ISWC 2013, Springer, Berlin, Heidelberg, 2013, pp. 363–378.
[16] Jingjing Wang, Haixun Wang, Zhongyuan Wang, Kenny Q. Zhu, Understanding tables on the web, in: Proceedings of the 31st International Conference on Conceptual Modeling, ER'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 141–155.
[17] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, Chung Wu, Recovering semantics of tables on the web, Proc. VLDB Endow. 4 (9) (2011) 528–538.
[18] Zareen Syed, Tim Finin, Varish Mulwad, Anupam Joshi, Exploiting a web of semantic data for interpreting tables, in: Proceedings of the Second Web Science Conference, vol. 5, 2010.
[19] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, Charles Sutton, Colnet: Embedding the semantics of web tables for column type prediction, in: 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 2019.
[20] Shuo Zhang, Krisztian Balog, Ad hoc table retrieval using semantic similarity, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, Republic and Canton of Geneva, Switzerland, International World Wide Web Conferences Steering Committee, 2018, pp. 1553–1562.
[21] Kunihiro Takeoka, Masafumi Oyamada, Shinji Nakadai, Takeshi Okadome, An efficient probabilistic approach for semantically annotating tables, in: 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 2019.
[22] Dumitru Roman, Nikolay Nikolov, Antoine Putlier, Dina Sukhobok, Brian Elvesæter, Arne Berre, Xianglin Ye, Marin Dimitrov, Alex Simov, Momchill Zarev, et al., Datagraft: One-stop-shop for open data management, Semant. Web 9 (43) (2018) 93–411.
[23] Shubham Gupta, Pedro Szekely, Craig A. Knoblock, Aman Goel, Mohsen Taheriyan, Maria Muslea, Karma: A system for mapping structured sources into the semantic web, in: Elena Simperl, Barry Norton, Dunja Mladenic, Emanuele Della Valle, Irini Fundulaki, Alexandre Passant, Raphaël Troncy (Eds.), The Semantic Web: ESWC 2012 Satellite Events, Springer, Berlin, Heidelberg, 2015, pp. 430–434.
[24] Tomáö Knap, Towards odalic, a semantic table interpretation tool in the adequate project, in: Proceedings of the 5th International Workshop on Linked Data for Information Extraction co-located with the 16th International Semantic Web Conference, ISWC 2017, Vienna, Austria, 2017, pp. 26–37.
[25] Suvodeep Mazumdar, Ziqi Zhang, A tool for creating and visualizing semantic annotations on relational tables, in: LD4IE@ISWC. CEUR Workshop Proceedings, 2016.
[26] Kristina Lerman, Plangprasopchok, A. Knoblock, Semantic labeling of online information sources, Int. J. Semant. Web Inf. Syst. 3 (3) (2007) 36–56, Copyright - Copyright IGI Global Jul-Sep 2007; Last updated - 2016-09-24.

[27] Kun Il Park, Park, Fundamentals of Probability and Stochastic Processes with Applications To Communications, Springer, 2018.
[28] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, Andrea Maurino, Abstat: ontology-driven linked data summaries with pattern minimalization, in: International Semantic Web Conference, Springer, 2016, pp. 381–395.
[29] Dominique Ritze, Christian Bizer, Matching web tables to dbpedia - a feature utility study, in: Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21–24, 2017, Konstanz, 2017, pp. 210–221, OpenProceedings.
[30] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, Octavian Udrea, Apples and oranges: A comparison of rdf benchmarks and real rdf datasets, in: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, SIGMOD '11, ACM, New York, NY, USA, 2011, pp. 145–156.
[31] Marco Cremaschi, Roberto Avogadro, David Chieregato, Mantistable: an automatic approach for the semantic table interpretation, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[32] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, Hideaki Takeda, Mtab: Matching tabular data to knowledge graph using probability models, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[33] Gilles Vandewiele, Bram Steenwinckel, Filip De Turck, Femke Ongenae, Cvs2kg: Transforming tabular data into semantic knowledge, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[34] Daniela Oliveira, Mathieu d'Aquin, Adog - annotating data with ontologies and graphs, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[35] Avijit Thawani, Minda Hu, Erdong Hu, Husain Zafar, Naren Teja Divvala, Amandeep Singh, Ehsan Qasemi, Pedro Szekely, Jay Pujara, Entity linking to knowledge graphs to infer column types and properties, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[36] Hiroaki Morikawa, Semantic table interpretation using lod4all, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.
[37] Marco Cremaschi, Alessandra Siano, Roberto Avogadro, Ernesto Jimenez-Ruiz, Andrea Maurino, STILTool: a Semantic Table Interpretation evaLuation Tool, The Semantic Web: ESWC 2020 Satellite Events, 2020.
[38] Benno Kruit, Peter A. Boncz, Jacopo Urbani, Extracting novel facts from tables for knowledge graph completion (extended version), 2019, CoRR, arXiv:abs/1907.00083.
[39] Yoan Chabot, Thomas Labbe, Jixiong Liu, Raphaël Troncy, Dagobah: An end-to-end context-free tabular data semantic annotation system, in: Proceedings of the Semantic Web Challenge on Tabular Data To Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, ISWC 2019, CEUR-WS.org, 2019.

Marco Cremaschi is a Ph.D. at the University of Milano-Bicocca. His research interest is about Semantic Table Interpretation and recently his interest is also in the Machine Translation research area using Semantic Web technologies. He has been involved in several national and international projects. He has been mentoring a lot of bachelor and master students.