MantisTable SE: an Efficient Approach for the Semantic Table Interpretation

Marco Cremaschi, Roberto Avogadro, Andrea Barazzetti, and David Chieregato

University of Milan - Bicocca, Viale Sarca 336, 20126 Milan, Italy
marco.cremaschi@unimib.it
{r.avogadro,d.chieregato,a.barazzetti1}@campus.unimib.it

Abstract. In this paper, we present a novel unsupervised and automatic approach for Semantic Table Interpretation. The technique presented is performed against DBpedia and Wikidata, and it can be easily adapted to any other Knowledge Graph (KG). Moreover, we provide a tool (LamAPI) that allows to efficiently fetch data needed for Semantic Table Interpretation (STI) tasks from the KG dumps.

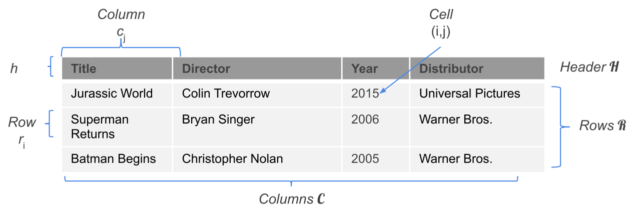
Keywords: Semantic Table Interpretation · DBpedia · Wikidata · Knowledge Graph

1 Introduction

The STI is a research field in continuous evolution with increasing interest over time. To mention a few numbers describing the phenomenon:

– Web Tables: The WebTables system [1] extracts 14.1 billion HTML tables and finds 154 million are high-quality tables (1.1%);
– Web Tables: Lehmberg et al. [5] extract 233 million content tables from Common Crawl 2015 (2.25% of all tables);
– Wikipedia Tables: The current snapshot of Wikipedia contains more than 3.23 million tables from 520k articles [3];
– Spreadsheets: The number of worldwide spreadsheet users is estimated to exceed 400 million, and about 50% to 80% of businesses use spreadsheets.

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 Figure 1, and ii) a *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 Figure 2. The output returned is a semantically annotated table, as shown in Figure 3. This work is an improvement and extension of MantisTable [2]. Our approach used DBpedia as first matching


Column
c_j
Cell
(i,j)
h
Header $\mathcal{H}$
Row
r_i
Rows $\mathfrak{R}$
Columns $\mathcal{C}$

Title	Director	Year	Distributor
Jurassic World	Colin Trevorrow	2015	Universal Pictures
Superman Returns	Bryan Singer	2006	Warner Bros.
Batman Begins	Christopher Nolan	2005	Warner Bros.

Fig. 1: Example of a well-formed relational table, with notes that are used in this paper.

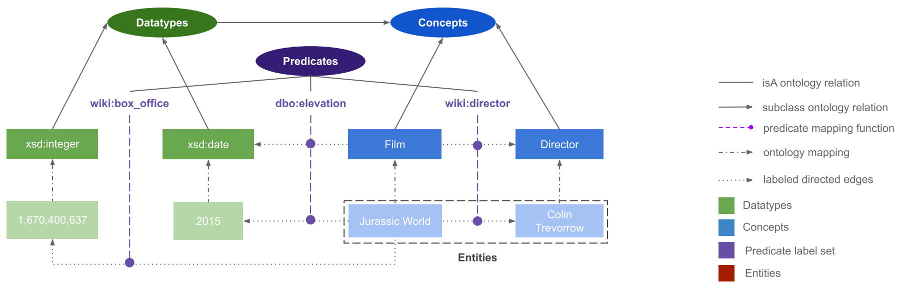


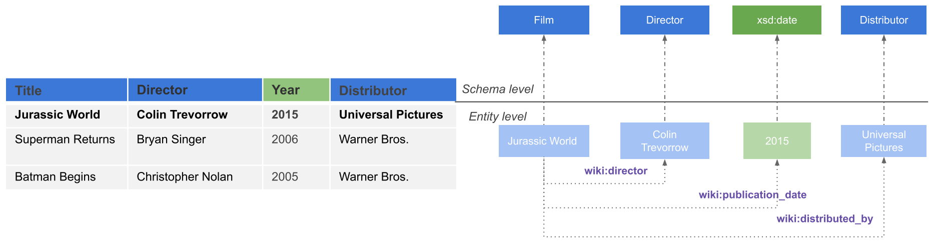
Fig. 2: A sample of Knowledge Graph.

KG because it was the richest data source available with ground truths available, later we adapted our approach to Wikidata because of it's increasing popularity and in order to take part in the SemTab2020 challenge[4]. The requirement to adapt the approach to Wikidata drove us to the development of a new approach. In this way, we showed that the approach can be easily adapted to be used with any KG. We will be referring to our new approach with "MantisTable SE", and it is the 4th Open Source implementation of MantisTable.

2 Methodology

As seen in Section 1, to obtain the STI of tabular data, it is required to link elements of the table with the elements in a KG. The elements in KGs (e.g. DBpedia or Wikidata) are frequently stored in Resource Description Framework (RDF) format, so to access to these elements, it is necessary to query a SPARQL endpoint. For instance, the most popular way to access DBpedia dumps is by using OpenLink Virtuoso, a row-wise transaction-oriented RDBMS with a SPARQL query engine to access to RDF graph store[1]. Wikidata instead uses Blazegraph[2] that is a high-performance graph database supporting Blueprints and RDF/SPARQL APIs. The issue faced with these solutions is the time required for importing the data. Wikidata2019 dump requires some days to set

[1] http://virtuoso.openlinksw.com
[2] https://blazegraph.com



Title	Director	Year	Distributor
Jurassic World	**Colin Trevorrow**	**2015**	**Universal Pictures**
Superman Returns	Bryan Singer	2006	Warner Bros.
Batman Begins	Christopher Nolan	2005	Warner Bros.

Fig. 3: Example of an annotated table.

up[3]. Another problem is given by the amount of information present in KG; for instance, the Wikidata dump is about 1.1TB (uncompressed). The English version of DBpedia instead is split into multiple files of the size of 26GB, which leads to high computation times to obtain a complete STI (e.g. TableMiner+ according to the author [6] took 13.35 hours to process the Limaye200 dataset). However, not all the information present in a KG is necessary to carry out a STI. Therefore, in order to obtain an efficient approach, it is necessary to identify other ways for querying KG. To do that, the approach described in this paper does not use SPARQL queries but queries indexes built on DBpedia and Wikidata dump files. These indexes are accessible through the use of four different API services that allow us to replace all the SPARQL queries. The tool that provides these APIs is called LamAPI. LamAPI is an open-source tool, and the code is available as a Git repository[4]. LamAPI provides the following services:

1. **Label Matching**: given a free text (in this case a data inside the table cell), it retrieves the entities with the greatest similarity, using ElasticSearch's default sorted scoring algorithm with full-text search enabled. For instance, considering a table cell containing "Jurassic World" the result returned is shown in Listing 1.1 or Listing 1.2;
2. **Predicate matching**: given two entities it retrieves all the predicates between them; considering the entity "Jurassic_World" and the entity "Colin_Trevorrow", the list of predicates is shown in Listing 1.4;
3. **Object matching**: given an entity it retrieves all the related objects and predicates; for example, with the entity "Jurassic_World" the result is the one shown in Listing 1.4;
4. **Concept matching**: given an entity it retrieves all it's concepts as shown in Listing 1.3.

[3] https://addshore.com/2019/10/your-own-wikidata-query-service-with-no-limits-part-1/
[4] https://bitbucket.org/disco_unimib/lamapi

Listing 1.1: Wikidata labels.

```
  "count": 70,
  "results": [{
    "uri": "Q55615459",
    "label": "Jurassic World"
  },{
    "uri": "Q3512046",
    "label": "Jurassic World"
  },{
    "uri": "Q18615494",
    "label": "Jurassic World"
  },{
    "uri": "Q2336369",
    "label": "Jurassic World"
  },{
    "uri": "Q37598390",
    "label": "Jurassic World Evol."
  }
```

Listing 1.2: DBpedia labels.

```
  "count": 31,
  "results": [{
    "uri": "Jurassic_World",
    "label": "Jurassic World"
  },{
    "uri": "Jurassic_World",
    "label": "Jurassic world"
  },{
    "uri": "Jurassic_World",
    "label": "Jurassic World film"
  },{
    "uri": "Jurassic_World",
    "label": "Jurassic World (film)"
  },{
    "uri": "Lego_Jurassic_World",
    "label": "Lego Jurassic World"
  }
```

Listing 1.3: Query result - Literals and Concepts.

```
Literal
{
  "Jurassic_World": {
  "dbo:budget": [
    "1.5E8"
  ],
  "dbo:gross": [
    "1.67E9"
  ],
  "foaf:name": [
    "Jurassic␣World"
  ],
  "dbo:runtime": [
    "7440.0"
  ]
  }
}
Concept
{
  "Jurassic_World": [
  "Film",
  "Work"
  ]
}
```

Listing 1.4: Query results - Predicate and Objects.

```
Predicate
{
  "Jurassic_World␣Colin_Trev.": [
  "dbo:director"
  ]
}
Object
{
  "Jurassic_World": {
  "dbo:director": [
    "Colin_Trevorrow"
  ],
  "dbo:musicComposer": [
    "Michael_Giacchino"
  ],
  "dbo:cinematography": [
    "John_Schwartzman"
  ],
  "dbo:editing": [
    "Kevin_Stitt"
  ],
  "dbo:distributor": [
    "Universal_Studios"
  ]
```

The loading of the labels (as shown in Listing 1.1 and Listing 1.2) takes place through the use of pre-computed JSON files. The other three services are made with Redis[5] key-value database. As a search engine for the labels service, we use an instance of ElasticSearch[6] that supports HTTP GZip natively, giving a performance boost. KG's dump files need some pre-processing in order to be used for this system, in particular we use namespace abbreviations (e.g. *dbo:* for *dbpedia.org/ontology*) in order to save space and we avoid data duplication.

In the following, we will focus on the algorithmic process of our MantisTable SE. The process is organised into eight phases as follows: i) Data Preparation and Normalisation, ii) Column Analysis and Subject Detection, iii) Data Retrieval, iv) Cell Entity Annotation (CEA), v) Column Predicate Annotation (CPA), vi) Column Type Annotation (CTA), vii) Revision and viii) Export.

The process was designed to retrieve the candidate for a given cell only once, especially if the content of that cell is repeated across multiple tables. This allows

[5] https://redis.io/
[6] https://www.elastic.co/

the approach to avoid repeating queries for the same content and saves network time, but the drawback is more difficult memory management during execution. Every phase must be completed for all the tables before going to the next one. This does not preclude running the tool against only one table, but when running with multiple tables the execution time will be sharply reduced (for the same text in a cell we have the same candidate entities that will be sorted considering the content of every table. This will be explained more in detail in the CEA phase). For examples, we will refer to Table 1.

SUBJ	NE	LIT	NE	LIT	LIT
title	director	release year	domestic distributor	length in min	worldwide gross
jurassic world	colin trevorrow	2015	universal pictures	124	1670400637
superman returns	bryan singer	2006	warner bros	154	391081192
batman begins	christopher nolan	2005	warner bros	140	371853783
avatar	james cameron	2009	20 century fox	162	2744336793

Table 1: Example of a table containing a list of movies.

i) Data Preparation and Normalisation. During this phase, all the cells of all the tables are analysed using a tokeniser managing special characters and removing parenthesis and the text block inside them.

ii) Column Analysis and Subject Detection. During *Column Analysis* we identify literal columns (L-column) by using a set of regex [2] (i.e. BOOLEAN, DATE, EMAIL, GEOCORDS, INTEGER, REAL, ISBN, URL) to identity different datatypes. The other columns are annotated as Named Entity column (NE-column). The subject column (S-column) can be identified between NE-column thanks to content-based scores, but it will not be discussed in this approach as it would not introduce anything new from [2,6]. Besides, target columns are provided during SemTab2019 and SemTab2020 challenges.

iii) Data Retrieval. For each normalised cell, the candidate entities are retrieved from the *Label Matching API* service. Retrieving data implies to seek for the entities in the KG with similar labels to our Named Entities cells. For example, if we query "jurassic world" on LamAPI Wikidata we get a list of candidates:

– Q3512046 ("Jurassic World" - Movie);
– Q18615494 ("Jurassic World" - Comic Strip);
– Q55615459 ("Jurassic World" - Attraction);
– thousands more results.

The candidates will be ranked during the next phases. Cells of NE columns that have the same string value in different tables start by considering the same set of candidate entities; this set will be sorted differently, considering relationships between the entire contents of a table.

iv) Cell Entity Annotation (CEA). For each cell of the S-column and NE-columns a confidence score 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 $tx(i,j)$:

$$1 - norm(LevenshteinDistance(tx(i,j)), e_{i,j}) \tag{1}$$

All the values are normalised in (0,1) range with Divide by Maximum normalisation (for every entity). For L-columns, the confidence score is computed as follows:

– for L-columns with numeric datatype: The confidence score is calculated as of the formula in Equation 2. Where a is the cell and b is the corresponding candidate value in the KG and σ is a constant (experimentally fixed at 100).

$$e^{-0.5\left[\frac{(a-b)}{\sigma}\right]^2} \tag{2}$$

– for L-columns with string datatype: the confidence score is computed like in the case of NE-column for short strings. For long strings, a Jaccard distance is computed, because the number of edits required to change a long string into another one is not necessarily significant while considering n-grams with Jaccard is generally a better similarity measure.
– for L-column with date datatype, the dates are considered as sortable numeric values in the format YYYYMMDDHHmmSS. The confidence score is computed in the same way as in the numeric case.

As an example we consider the cell containing "superman returns" which can refer to both a movie (Table 1) and a videogame (Table 2).

videogame	**publisher**	**release date**
superman returns	electronic arts	2006
pokemon white	nintendo	2010
call of duty	activision	2003

Table 2: Example of table with elements equal to Table 1, but different domain.

Listing 1.5: Candidates for the cell "superman returns" of the movies table (Table 1).

```
"Q328695":
    "label": "Superman Returns",
    "instance_of": ["3D film", "film"],
    "confidence": 0.8
"Q655031":
    "label": "Superman Returns",
    "instance_of": ["video game"],
    "confidence": 0.2
"Q3977963":
    "label": "Superman Returns",
    "instance_of": ["album"],
    "confidence": 0.2
```

Listing 1.6: Candidates for the cell “superman returns” of the Table 2.

```
"Q655031":
    "label": "Superman Returns",
    "instance_of": "video game",
    "confidence": 1.0
"Q7643850":
    "label": "Superman Returns: Fortress of Solitude",
    "instance_of": "video game",
    "confidence": 0.41
```

Considering literal values for the cell “batman begins” with the content of the column “length in min” we are almost sure that the value is correct because after sorting all the values we have the result shown in Listing 1.7.

Listing 1.7: Literal values for the entity “Batman Begins (film)”.

```
"P4632":
    "label": "Bechdel Test Movie List ID",
    "value": 40
    "confidence": 0
"P2047":
    "label": "duration",
    "value": 140
    "confidence": 1.0
"P3110":
    "label": "ISzDb film ID",
    "value": 234
    "confidence": 0
```

v) Column Predicate Annotation (CPA). Considering that all the necessary information is gathered in the previous phase, the CPA is a relatively fast process. For each column, all the predicates are retrieved for every possible candidate couple with confidence greater than 0 and they are sorted by their relative frequency to the entire column. The predicate with the greatest frequency will be ranked first. This process allows to reduce the number of candidate entities for every cell. Confidence scores for the predicates of the *director* column are shown in Listing 1.8.

When we consider the video game in Table 2, the CPA phase allowed us to drop the “film” with the same name. This because it does not have any relationship with the rest of the content of the table (the film does not have anything to do with “electronic arts”, while the video game has a property with exact match).

Listing 1.8: Predicates for “Director” column.

```
"P57":
    "label": "director",
    "confidence": 1.0
"P58":
    "label": "screenwriter",
    "confidence": 0.75
"P162":
    "label": "producer",
    "confidence": 0.75
"P161":
    "label": "cast member",
    "confidence": 0.25
```

vi) Column Type Annotation (CTA). In this phase every candidate entity $e_{i,j} \in E_{i,j}$ is analysed again considering the CPA reordering. The process is a bit different in DBpedia and Wikidata because the Wikidata KG contains cycles. For reasons of space, in this paper, only the approach for Wikidata will be considered. In the Wikidata version of MantisTable SE, we retrieve the concepts corresponding to every candidate entity from LamAPI. For every column, we use an in-memory structure storing frequencies of the most common datatypes, as shown in Listing 1.9.

Listing 1.9: Example of the structure storing the most frequent concepts.

```
  "movie_table":
  "0":
    "Q229390": 1,
    "Q11424": 1,
    "Q25110269": 0.33
  "1":
    "Q5": 1
  "3":
    "Q1762059": 1,
    "Q375336": 0.66,
    "Q1107679": 0.33,
    "Q18127": 0.33,
    "Q10689397": 0.33
```

The frequencies are stored with by-max normalization in order to be homogeneous (i.e. values are between 0 and 1). For every column, the winning concept is chosen as follows:

– concepts with a frequency lower than 0.95 are discarded in order to reduce the needed computation on the fully connected Wikidata concepts graph;
– in every single table, we consider every possible column pair and we count inbound and outbound edges; the final concept selected is the one having the highest number of connections in the Wikidata graph.

Listing 1.10: Confidence for first column of the film table (Table 1).

```
"Q229390":
    "label": "3D film",
    "confidence": 1.0
"Q11424":
    "label": "film",
    "confidence": 1.0
"Q11424": {
    "label": "live-action animated film",
    "confidence": 0.33
```

As both “film” and “3D film” have the same confidence, we consider the number of links in the Wikidata Graph. In our algorithm, we initially ranked first concepts with the highest number of links in the graph, but after finding out experimentally that in the SemTab2020 challenge a most specific concept is preferred by the scoring system, we decided to consider the relationship between the final concepts. In this example, as “3D film” is a child concept of “film”, we assign the most specific concept, so “3D film” is the final winning concept. This consideration should not be considered ideal for every dataset.

Listing 1.11: Links in Wikidata Graph for concepts of first column of Table 1.

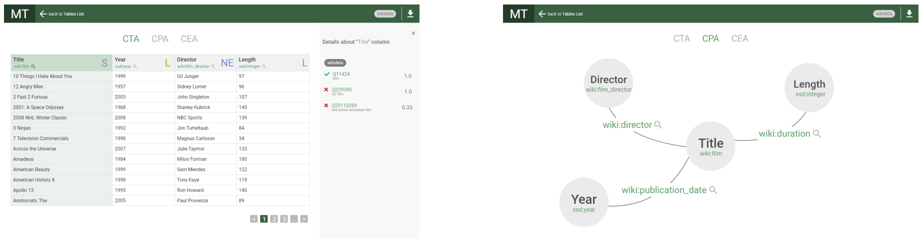
```
"Q229390":
    "label": "3D film",
    "links": 0
"Q11424":
    "label": "film",
    "links": 1
```

vii) Revision. The revision phase analyses all the information gathered in the previous phases in order to do a final reordering of the candidates. In particular, this allows for correcting CEA entities previously selected: every entity have to be coherent with the rest of the concepts and predicates selected in every column. Moreover, predicates are also re-ranked.

viii) Export. The MantisTable SE approach previously described keeps the candidates coming from each phase; all the results are stored. It is possible to apply some thresholds during the export phase to reach a balance between annotation quality and the number of annotations provided. The export threshold can have a significant role in evaluation metrics for every gold standard.

3 Deployment

As described above, the MantisTable SE approach has been integrated into a web application developed with Python and Django (Figure 4a, Figure 4b). A MongoDB database acts as table and KG repository. The code is freely available through a Git repository[7]. In order to achieve the scalability of the application, and therefore improve efficiency, MantisTable SE has been installed in a Docker container to achieve parallelisms at the application level and to facilitate the deployment on servers. The management of resources is performed by using Task Queues (i.e. Celery Workers[8]). The GUI is current under development, when ready it will allow to graphically explore annotations for every phase.



Details about "Title" column

wikidata

✓	Q11424 film	1.0
✗	Q229390 3D film	1.0
✗	Q25110269 live-action animated film	0.33

Title wiki:film S	Year xsd:year L	Director wiki:film_director NE	Length xsd:integer L
10 Things I Hate About You	1999	Gil Junger	97
12 Angry Men	1957	Sidney Lumet	96
2 Fast 2 Furious	2003	John Singleton	107
2001: A Space Odyssey	1968	Stanley Kubrick	145
2008 NHL Winter Classic	2008	NBC Sports	139
3 Ninjas	1992	Jon Turteltaub	84
7 Television Commercials	1998	Magnus Carlsson	34
Across the Universe	2007	Julie Taymor	133
Amadeus	1984	Milos Forman	180
American Beauty	1999	Sam Mendes	122
American History X	1998	Tony Kaye	119
Apollo 13	1995	Ron Howard	140
Aristocrats, The	2005	Paul Provenza	89

< 1 2 3 ... >

(a) (b)

Fig. 4: MantisTable SE page with CTA(4a) and CPA(4b) results

4 Results

This section will report the results obtained by MantisTable SE in Rounds 1, 2, 3 of the SemTab 2020 challenge.

[7] bitbucket.org/disco_unimib/mantistable-4

[8] docs.celeryproject.org/en/latest/userguide/workers.html

Tasks	Round1		Round2		Round3	
	F1	P	F1	P	F1	P
CEA	0.982	0.989	0.991	0.993	0.974	0.979
CTA	0.745	0.753	0.966	0.973	0.958	0.965
CPA	0.888	0.942	0.961	0.966	0.941	0.957

The results obtained in the first three rounds shows that our method is highly promising. Unfortunately, during Round4, our results are not representative. Due to an infrastructural problem, it was not possible to complete the computation in the given timeframe.

5 Conclusions and General comments

MantisTable SE i) provides a comprehensive solution to support all annotation steps; ii) provides an unsupervised method to annotate datasets of tables efficiently; iii) generates context for disambiguation; iv) provides a tool to support KG querying for STI tasks (LamAPI) and a workflow for STI (MantisTable SE) which are publicly available.

Talking about future improvements:

- CEA does take in consideration only directly linked entities. This is likely the most important source of missing/wrong annotations and is going to be solved in the next version;
- CPA is directly computed from CEA results but some ambiguities still remain. A slightly more complex algorithm is currently under development;
- for CTA annotations we will possibly use space-representations and feature-based similarities (e.g. embeddings) to match missing concepts in the KG.

References

1. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: Exploring the power of tables on the web. Proc. VLDB Endow. **1**(1), 538–549 (Aug 2008). https://doi.org/10.14778/1453856.1453916, https://doi.org/10.14778/1453856.1453916
2. Cremaschi, M., De Paoli, F., Rula, A., Spahiu, B.: A fully automated approach to a complete semantic table interpretation. Future Generation Computer Systems **112**, 478 – 500 (2020). https://doi.org/https://doi.org/10.1016/j.future.2020.05.019, http://www.sciencedirect.com/science/article/pii/S0167739X19302663
3. Fetahu, B., Anand, A., Koutraki, M.: Tablenet: An approach for determining fine-grained relations for wikipedia tables. In: The World Wide Web Conference. p. 2736–2742. WWW '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3308558.3313629, https://doi.org/10.1145/3308558.3313629
4. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In: European Semantic Web Conference. pp. 514–530. Springer (2020)

5. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th International Conference Companion on World Wide Web. p. 75–76. WWW '16 Companion, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2016). https://doi.org/10.1145/2872518.2889386, `https://doi.org/10.1145/2872518.2889386`
6. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semantic Web **8**(6), 921–957 (2017)