

s-elBat: a Semantic Interpretation Approach for Messy taBle-s

Marco Cremaschi¹, Roberto Avogadro¹ and David Chierigato¹

¹University of Milan - Bicocca, viale Sarca 336, Edificio U14, 20126, Milan, Italy

Abstract

This paper describes s-elBat, a Semantic Table Interpretation approach. The approach inherits and improves the part of the techniques belonging to the MantisTable, an approach used and tested in previous editions of the SemTab challenge. s-elBat adds an innovative and optimised lookup approach for generating candidate entities for the annotation.

Keywords

Semantic Table Interpretation, Tabular Data, SemTab Challenge, Knowledge Graph

1. Introduction

Tables are everywhere and play a crucial role in creating, organising, and sharing information on the Web of Data. The increase in the spread of tables 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 [1]. The phenomenon can be sized by the number of available tables or the number of users who use Google Sheets or Microsoft Excel:


- Web Tables: in 2008 were extracted 14.1 billion HTML tables, and it was found that 154 million were high-quality tables (1.1%);
- Web Tables: 233 million content tables in Common Crawl 2015 repository¹;
- Wikipedia Tables: the 2022 English snapshot of Wikipedia contains 2 803 424 tables from 21 149 260 articles [2];
- Spreadsheets: there are 750 million to 2 billion people in the world who use either Google Sheets or Microsoft Excel².

Tables contain high-value data, but they can be challenging to understand for humans and machines due to several factors, such as the ambiguity of the values contained therein and the lack of contextual information (e.g., metadata).

SemTab: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching 2022

✉ marco.cremaschi@unimib.it (M. Cremaschi); roberto.avogadro@unimib.it (R. Avogadro); david.chierigato@unimib.it (D. Chierigato)

ORCID 0000-0001-7840-6228 (M. Cremaschi); 0000-0001-8074-7793 (R. Avogadro)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

¹commoncrawl.org

²askwonder.com/research/number-google-sheets-users-worldwide-eoskdoxav

The table-to-KG matching problem, also referred to as Semantic Table Interpretation (STI), provides explicit semantic annotations (e.g., identifying and annotating entities in cells, their types/classes and the connections/properties between entities), thus capturing knowledge from tables [3]. STI has recently collected much attention in the research community [4, 5, 6] and is a key step to enrich data [7, 8] and construct and extend Knowledge Graphs (KGs) from semi-structured data [9, 10].

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), and ii) a *Knowledge Graph (KG)* (e.g., Wikidata, DBpedia) which describes real-world entities in the domain of interest (i.e., a set of types, datatypes, predicates, instances, and the relations among them). The output is a semantically annotated table, obtained by mapping its elements (i.e., cells/mentions, columns, rows) to semantic tags (i.e., entities, types, properties) from KGs as shown in Figure 1. This process is typically broken down into the following tasks: (i) cell/mentions to KG entity matching (CEA task), (ii) column to KG type matching (CTA task), and (iii) column pair to KG property matching (CPA task) [6].

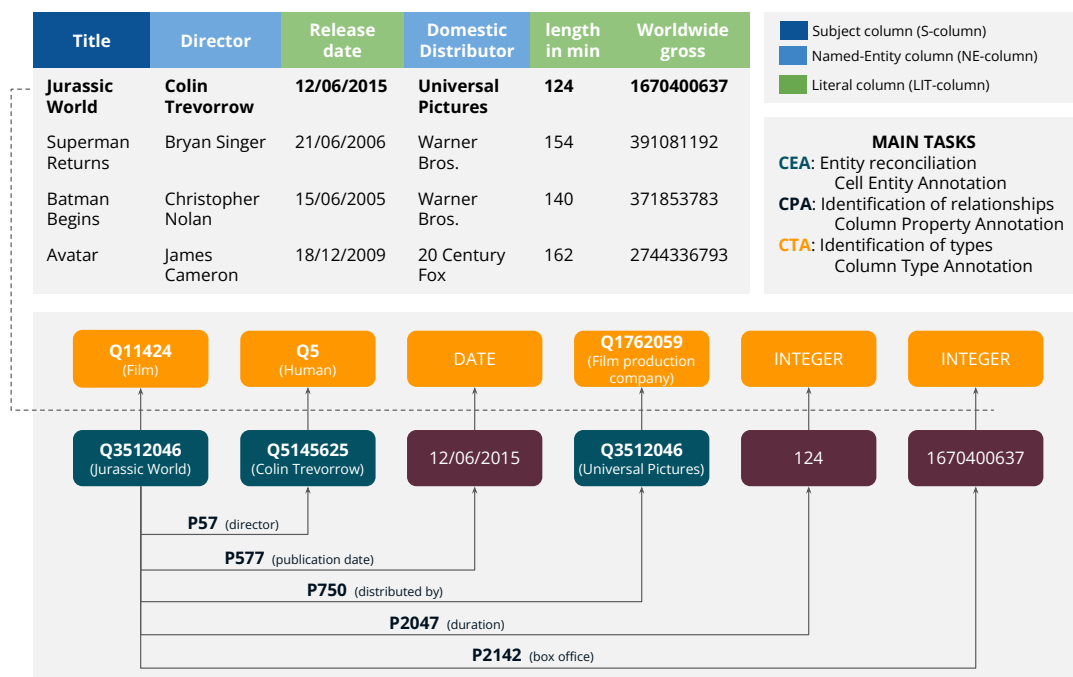


Figure 1: Example of an annotated table.

As depicted in Figure 1 the majority of entities in the Title column are of type Film. publication_date can be identified as the property connecting entities in the Title column with elements in the Year column.

Unfortunately, explicit situations like the ones in the example are not so common; therefore, we need to set up strategies and algorithms to address several issues. An excellent STI approach must consider and adequately balance the different features of a table (or a set of tables). The

annotation involves several key challenges: i) *disambiguation*: the type of the entities described in a table are not known in advance, and those entities may correspond to more than one type in the KG. ii) *homonymy*: this issue is related to the presence of different entities with the same name and type. iii) *matching*: the mention in the table may be syntactically different from the label of the entity in a KG (*i.e.*, use of acronyms, aliases, and typos). iv) *NIL-mentions*: the approach must also consider strings that refer to entities for which a representation has not yet been created within the KG, namely NIL-mentions. v) *literal and named-entity*: in a table, there can be columns that contain named-entity mentions (NE-column) and columns containing strings (L-column). vi) *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. For instance, the header, the first row of a table, which usually contains descriptive attributes for the columns, may or may not be present. vii) *amount of data*: the approach must consider large tables with many rows and columns, and tables with very few mentions. viii) *different domains*: the tables within a set can belong to very general or specific domains.

s-elBat³ is an approach that employs several techniques to consider all of these challenges. It is a new approach that inherits and improves what has been proposed by the MantisTable [3] and MantisTable SE [11] STI approaches. The experiences acquired with the tools mentioned above and their participation in the various editions of the SemTab challenge⁴ led to the definition of new techniques for i) an efficient lookup approach, through the use of indexes on optimised data structures, ii) an Information Retrieval-based Entity Linking, augmented with a type-based filtering feature, and iii) a feature vector based entity disambiguation approach.

The rest of the paper is organised as follows. In Section 2 we describe s-elBat in detail. Section 3 introduces the Gold Standards, the configuration parameters, and the evaluation results. Finally, we conclude this paper and discuss the future direction in Section 4.

2. s-elBat approach

s-elBat provides an iterative process that performs Entity Linking (EL) on tables. Given a KG containing a set of entities E and a collection of named-entity mentions M , the goal of EL is to map each entity mention $m \in M$ to its corresponding entity $e \in E$ in the KG. As described above, a typical EL service consists of the following modules [12]:

1. Entity Retrieval (ER). In this module, for each entity mention $m \in M$, irrelevant entities in the KG are filtered out to return a set E_m of candidate entities: entities that mention m may refer to. To achieve this goal, state-of-the-art techniques have been used, such as name dictionary-based techniques, surface form expansion from the local document, and methods based on search engines.
2. Entity Disambiguation (ED). In this module, the entities in the set E_m are more accurately ranked to select the correct entity among the candidate ones. In practice, this is a re-ranking activity that considers other information (*e.g.*, contextual information) besides the simple textual mention m used in the ER module.

³The name comes from taBle-s and Semantic Entity Linking to BAatch Table.

⁴www.cs.ox.ac.uk/isg/challenges/sem-tab/

In the *s-elBat* these modules are integrated in a pipeline composed of 7 sequential phases: Preprocessing and Data Preparation, Entity Retrieval, Cell Entity Annotation (CEA), Column Property Annotation (CPA), Column Type Annotation (CTA), Revision, Export. The overall framework is described in Figure 2.

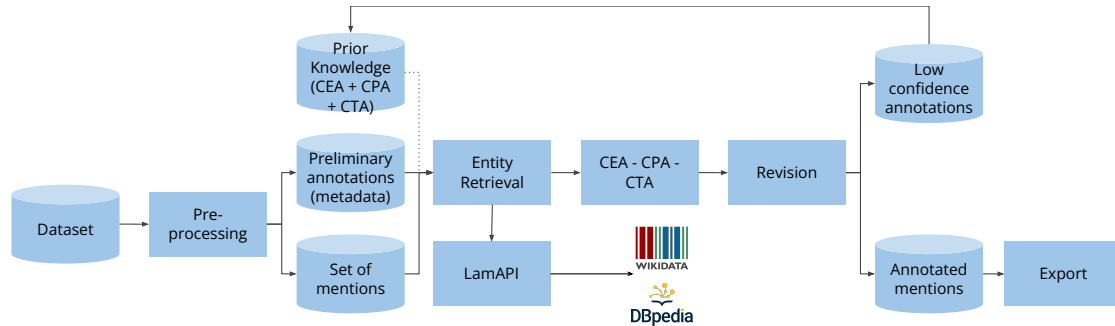


Figure 2: *s-elBat* process.

2.1. Preprocessing and Data preparation

During this phase, as a first step, every table's cells are converted into lowercase. The next step performs a column classification, associating L-column (columns containing strings) and NE-column (columns that contain named-entity mentions) tag to every column [3, 11]. The potential subject column (S-columns, the main column, the one all the others refer to) is identified [3]. With the *s-elBat* approach, the selected subject is not determinant for the final annotation but can positively influence the execution time. Eventually, the cells from NE-column are extracted to generate the set M of mentions for the next step.

2.2. Entity Retrieval

According to some state-of-the-art experiments [13], the role of the ER module is critical since it should ensure the presence of the correct entity in the returned set to let the ED module find it. For this reason, *s-elBat* integrates LamAPI (LAbel Matching API), a comprehensive tool for Information Retrieval (IR)-based ER, augmented with type-based filtering features [14]. The current version of LamAPI integrates the data of DBpedia (v. 2016-10 and v. 2022.03.01) and Wikidata (v. 20220708). The KGs are indexed with ElasticSearch⁵, an engine that can search and analyse huge volumes of data in near real-time. The ElasticSearch index was configured using the *IB similarity* as similarity function⁶ with the default values of hyperparameters. LamAPI is highly modular so it is possible to integrate any indexing engine (*e.g.*, Apache Solr, Apache Lucene, Arango DB). The entity's identifier and label are indexed. For each entity, the length in terms of characters, the number of tokens and the n-grams of the labels, and a value representing the entity's popularity are stored in the index.

⁵www.elastic.co

⁶www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html

Among the services provided by LamAPI to search and retrieve information from a KG, the Lookup and Types are described, which are the relevant services for the STI tasks.

Lookup: given a string as input, it retrieves a set of candidate entities E from the reference KG. The request can be qualified by setting some attributes:

- limit: an integer value specifies the number of entities to retrieve. The default value is 100; it has been empirically demonstrated how this limit allows a good level of coverage.
- kg: specifies which KG and version to use. The default is `dbpedia_2022_03_01`, and other possible values are `dbpedia_2022_03_01`, `dbpedia_2016_10` or `wikidata_latest`.
- fuzzy: a boolean value. When true, it matches tokens inside a string with an edit distance (Levenshtein distance) less than or equal to 2. This gives a greater tolerance for spelling errors. When false, the fuzzy operator is not applied to the input.
- ngrams: a boolean value. When true, it permits to search n-grams. After many empirical experiments, we set ‘n’ of n-grams equal to 3. A lower value can bring some bias in the search, while a higher value could not be very effective in terms of spelling errors. Using n-grams equal to 3 “albert einstein” is split in [‘alb’, ‘lbe’, ‘ber’, ‘ert’, ...]. When false, n-grams search is not applied.
- types: this parameter allows the specification of a list of types (e.g., `rdf:type` for DBpedia and `Property:P31 [instance of]` for Wikidata) associated with the input string to filter the retrieved entities. This attribute plays a key role in re-ranking the candidates, allowing a more accurate search based on input types.

Types: given the unique *id* of an entity as input, it retrieves all the types of which the entity is an instance. For DBpedia entities, the service returns direct types, transitive types, and Wikidata types of the related entity, while for Wikidata, it returns only the list of concepts/types for the input entity.

For each mention $m \in M$, the approach performs a search using the LamAPI Lookup service to retrieve a set of entity E . During the service invocation, some heuristics are applied to handle possible misspelt input. In detail, two different requests are made: i) using only the mention, ii) modifying the mention with the removing of repeated letters and brackets. For instance, by considering the mention “pariss”, the second query is created using “paris”. Repeated characters are a frequent mistake, and the 3-gram search implemented by Elasticsearch will be badly affected by this kind of mistake. At the same time, the fuzzy matching will easy overcome possibly missing double characters. Brackets affect the edit distance, and their content can frequently be irrelevant.

For the selection of the best set E_m of candidates, LamAPI computes a similarity between the mention $m \in M$ and the label $l(e)$ (i.e., `rdf:label` of e) of each entity $e \in E$:

$$\text{stringSimilarity}(m, l(e)) = 1 - \frac{\text{LevenshteinDistance}(m, l(e))}{\max\{\text{length}(m), \text{length}(l(e))\}} \quad (1)$$

A threshold determines whether to remove an entity e from the candidate set E . To evaluate the threshold empirically, four Gold Standard (GS) have been selected (2T [15], SemTab 2020 R4 [5], HardTable 2021 R2, and HardTable 2021 R3 [6]). The distribution of string similarity values between the entity labels within the GS and the corresponding mention in the table was analysed (Table 1). It turned out that setting a threshold of 0.40 is a good choice.

Table 1
String similarity values for different Gold Standards.

Dataset	total mentions	avg string similarity	<0.10	<0.20	<0.30	<0.40	<0.50	<0.60	<0.70	<0.80	<0.90	<1.0	exact match = 1.0
2T 2020	73327	81.03%	1839	3550	5263	5996	6904	7951	10281	21972	43055	57793	15534
		%	2.51%	4.84%	7.18%	8.18%	9.42%	10.84%	14.02%	29.96%	58.72%	78.82%	21.18%
SemTab 2020 R4	485190	98.18%	71	257	556	918	1363	2116	3067	5291	25927	93654	391536
		%	0.01%	0.05%	0.11%	0.19%	0.28%	0.44%	0.63%	1.09%	5.34%	19.30%	80.70%
HardTable 2021 R2	37153	98.48%	4	14	55	73	92	129	168	308	1721	5876	31277
		%	0.01%	0.04%	0.15%	0.20%	0.25%	0.35%	0.45%	0.83%	4.63%	15.82%	84.18%
HardTable 2021 R3	51169	97.11%	11	20	26	35	46	67	156	563	6128	13557	37612
		%	0.02%	0.04%	0.05%	0.07%	0.09%	0.13%	0.30%	1.10%	11.98%	26.49%	73.51%
Average		93.70%	0.64%	1.24%	1.87%	2.16%	2.51%	2.94%	3.85%	8.25%	20.17%	35.11%	64.89%

2.3. Cell Entity Annotation

During this phase, for each pair of candidates associated with two cells on the same row but from different columns, the respective properties are extracted by using LamAPI.

For each candidate entity e , a feature vector with the following items is created:

- **string similarity**: the score is based on Levenshtein edit distance; it is calculated between the mention and the label associated with the candidate entity.
- **jaccard**: the score is the same as string similarity but calculated with Jaccard distance instead of Levenshtein.
- **object**: this score is set only if there is a property between two candidate entities. In this case, the subject entity considered in this pair receives a boost equal to the string similarity score of that entity.
- **relation**: as the previous object score, the relation is set only if there is at least one property between the considered entities. This is the exact antagonist of the object score; in this case, the object entity receives a boost.
- **literal**: this score is applied for relations between the cell from the subject column and the cell from the L-column. In this case *date*, *number* and *string* values are compared as explained in [16].

Given a vector of features the final score for each entity is computed as follows:

$$score(e) = \sum_{i=1}^{\#features} w_i features_i(e) \quad (2)$$

The weights w are set as follow: string similarity 10, jaccard 8, object 3, relation 4, literal 7. The final score allows to rank the candidates in order to sort them. Considering Table 2, the candidates for the mention “Jurassic World” are reported in Listings 1. The entity with the highest score is Entity:Q3512046 [Jurassic World]; it is used to annotate the mention.

2.4. Cell Property Annotation

In this phase, the information collected during the previous phase is used to aggregate predicates by frequency. The CPA is relatively fast because all the necessary information was already gathered in the previous phase using LamAPI. The first step consists of creating a dictionary

Table 2

Example of relational table containing a list of movies.

title	director	release year	domestic distributor	length in min	worldwide gross
Jurassic World	Colin Trevorrow	2015	Universal Pictures	124	1,670,400,637
Superman Returns	Bryan Singer	2006	Warner Bros.	154	391,081,192
Batman Begins	Christopher Nolan	2005	Warner Bros.	140	371,853,783
Avatar	James Cameron	2009	Twentieth Century Fox	162	2,744,336,793

Listing 1: Candidate entities for the mention “Jurassic World” (Table 2).

```

1 "Q3512046":
2   "label": "Jurassic World"
3   "instance_of": ["3D film", "film", "sequel"]
4   "string_similarity": 1.44
5   "jaccard": 1.0
6   "object": 2.88
7   "relation": 0
8   "literal": 1.728
9   "score": 51.78
10 "Q21877685":
11  "label": "Jurassic World",
12  "instance_of": ["film"]
13  "string_similarity": 1.2
14  "jaccard": 1.0
15  "object": 2.88
16  "relation": 0
17  "literal": 0.809
18  "score": 39.99
19 "Q55178974":
20  "label": "Jurassic World 3"
21  "instance_of": ["film", "film project"]
22  "string_similarity": 0.875
23  "jaccard": 0.667
24  "object": 2.888
25  "relation": 0
26  "literal": 0.491
27  "score": 31.37

```

Listing 2: Set of properties for the “director” column (Table 2).

```

1 "P57":
2   "label": "director"
3   "confidence": 1.0
4 "P58":
5   "label": "screenwriter"
6   "confidence": 0.75
7 "P162":
8   "label": "producer"
9   "confidence": 0.5
10 "P161":
11  "label": "cast member"
12  "confidence": 0.25

```

for every column containing all the winning properties and related frequencies. In the second step, the most frequent property is selected for the CPA annotation. Given a *director* the most frequent property is Property:P57 [director] (Listing 2).

2.5. Cell Type Annotation

In this phase, the information collected during the CEA phase is used. To get the CTA annotation for a given column, all the cells of that column are iterated. During the process, the approach

Listing 3: Example of the structure storing the most frequent classes for each column in Table 2.

```
2 "movie_table":
3 "0":
4   "Q229390 (3D film)": 1.0,
5   "Q11424 (film)": 0.667,
6   "Q261636 (sequel)": 0.33
7 "1":
8   "Q5 (Human)": 1.0.
9   "Q2526255 (Film director)": 1.0
10  "Q28389 (screenwriter)": 1.0
11  "Q3282637 (film producer)": 0.667
12 "3":
13  "Q1762059 (film production company)": 1.0,
14  "Q375336 (film studio)": 0.5,
15  "Q1107679 (animation studio)": 0.5,
16  "Q18127 (record label)": 0.5,
   "Q4830453 (business)": 0.5,
   "Q10689397 (television production company)": 0.25
```

creates a dictionary with the frequency of all the classes of the winning entities obtained in the previous step. The type with the maximum frequency will be selected as an annotation for the column under analysis. An example for Wikidata is shown in Listing 3.

2.6. Revision

The revision process consists of setting constraints on types and predicates obtained in the first execution of annotation process. As this implies computing again all the phases; only low confidence mentions are considered to optimise computational efficiency. An experiment was conducted in order to identify the best criteria to be used for classifying these mentions. The experiment was redacted using the datasets available from the previous editions of the SemTab Challenge. For every dataset considered to revise the CEA results, all the mentions are checked against the corresponding ground truth, and errors are noted. In Figure 3 the results of this analysis are graphically represented by a chart. We can consider the x-axis as the cutoff threshold while the y-axis represents the number of wrong mentions that are not considered for the revision step, e.g., with a threshold of 0.6 on average, it would bring an inner error under 5%. Clearly, it is required to minimise the number of wrong mentions while considering computational efficiency. For the SemTab challenge, the threshold was set at 0.4 with an inner error under 2%.

Additional features to the previous feature vector have been added to obtain a better ranker. The new features were not known prior to the CEA phase; in detail:

- **cta**: This score is related to the types of candidates. Considering a candidate entity this score consists of the intersection between the types of the candidate entity with the types found on the whole column.
- **cpa**: In the same way as the previous score, the CPA considers the predicates when the mention is the subject and uses the scores collected from the CPA phase.

The key challenges presented in Section 1 are managed as follows: i) *disambiguation*: the disambiguation is managed by the ED module presented in Section 2; ii) *homonymy*: the homonym cases are generally resolved with row context, the scores “object”, “relation”, and

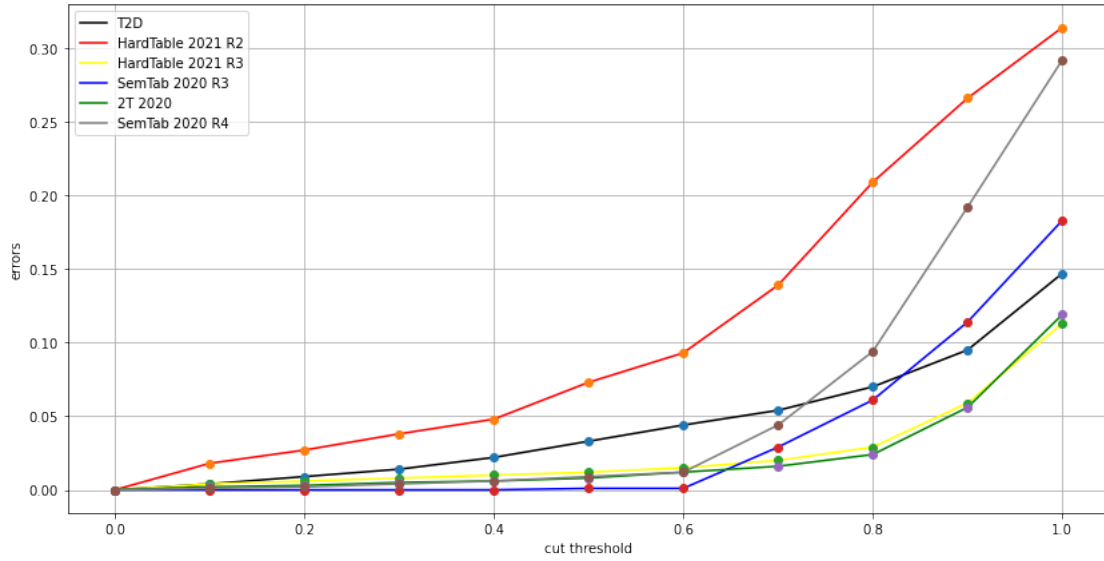


Figure 3: Analysis of the threshold for the Revision phase.

“literal” presented in the CEA phase help the resolution; iii) *matching*: LamAPI manages most of the matching issues that are encountered during the annotation process; iv) *NIL-mentions*: annotations with a confidence lower than 1 are highly likely to be NIL-mentions; v) *literal and named-entity*: the data preparation phase manages the column classification; vi) *missing context*: when the header context is missing, it is possible to use other kinds of context, such as the column context used in the CTA score; vii) *amount of data*: in this Section, it was proved that the proposed annotation process is growing linearly with the size of the data; viii) *different domains*: the approach was validated with general-purpose datasets. In the future, it may be fine-tuned for better performance on domain-specific tasks, for example, by reducing the possible candidate set.

2.7. Export

In this phase, the objective is to export the annotated mentions. For every mention, the system needs to decide whether the given annotation is correct or not based on confidence. This can lead to three possible scenarios: i) there is a clear winning candidate: if there is a score difference higher than 0.3 between the first-ranked candidate and the second one, the system is confident enough to annotate with the first-ranked candidate; ii) the final score is lower than 1, so the mention is considered a “No-annotation” because the system cannot be confident enough to provide an annotation; iii) the first two candidates that may be considered as winning have a final score which is too close to decide which one is correct. This can lead to unresolved mentions. In this case, three possible resolution methods are considered: a) the first one is taken, this can lead to a higher number of wrong annotations, but it is a fast way to annotate more mentions; b) the mentions are not annotated, this will lead to a higher recall; c) a ranking system based on the KG data is used to decide what annotation is the winning one. In Wikidata,

Listing 4: New candidates entities for the cell “Jurassic World” of the movies table (Table 2).

```

1 "Q3512046":
2   "label": "Jurassic World"
3   "instance_of": ["3D film", "film", "sequel"]
4   "string_similarity": 1.44
5   "jaccard": 1.0
6   "object": 2.88
7   "relation": 0
8   "literal": 1.72
9   "cta": 2.0
10  "cpa": 3.692
11  "score": 74.167
12 "Q21877685":
13  "label": "Jurassic World"
14  "instance_of": ["film"]
15  "string_similarity": 1.2
16  "jaccard": 1.0
17  "object": 2.88
18  "relation": 0
19  "literal": 0.809
20  "cta": 0.667
21  "cpa": 3.442
22  "score": 26.939
23 "Q55178974":
24  "label": "Jurassic World"
25  "instance_of": ["film", "film project"]
26  "string_similarity": 0.875
27  "jaccard": 0.667
28  "object": 2.88
29  "relation": 0
30  "literal": 0.491
31  "cta": 0.667
32  "cpa": 1.545
33  "score": 20.733

```

when annotating on general-purpose data, one possible criterion is to take the lowest identifier because more popular entities were generally created before newer ones.

3. Validation

Table 3 shows the results obtained by the s-elBat tool for three main tasks (CEA, CPA, and CTA) on 2T and HardTable datasets. Overall, these results show that s-elBat tool achieves a significant performance across multiple datasets from different domains and generation methods. The results show that the methodology presented in this paper is a general-purpose approach that can be applied to any dataset.

Table 3
Results for the SemTab 2022 dataset

Tasks	HardTable 2022 R1		HardTable 2022 R2		2T 2022	
	P	F1	P	F1	P	F1
CEA	0.964	0.945	0.875	0.825	0.938	0.937
CTA	0.951	0.957	0.878	0.859	0.367	0.366
CPA	0.989	0.983	0.96	0.931	-	-

Experiments were carried out to determine the computational efficiency of the different phases and to validate the assumption that the computation time grows linearly based on the data size. The data from the previous editions of the challenge is used to validate those

assumptions. In Figure 4, it is possible to see how different datasets with an increasing number of mentions have a nearly linear outcome regarding execution time.

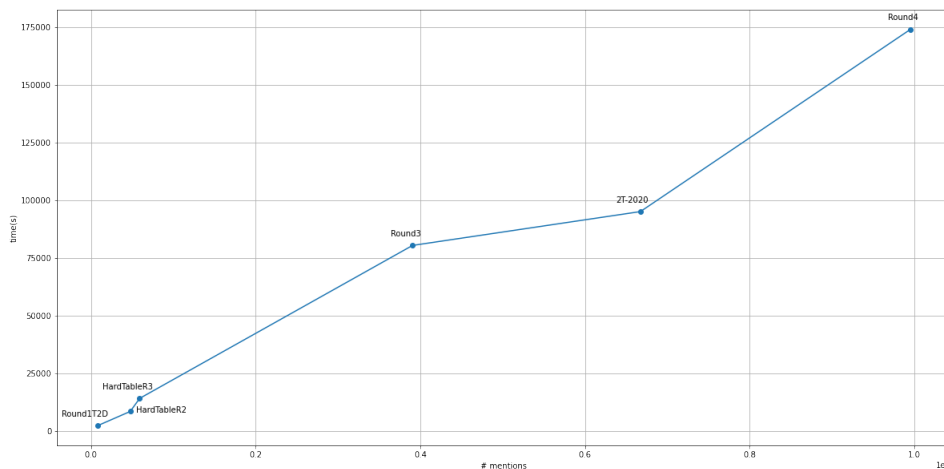


Figure 4: Computation time analysis of s-elBat approach.

In Table 4 the complete data about execution time is available. From this analysis, the results show that the most computationally expensive phase is the candidate generation. More in detail, it is possible to notice how the candidate generation consists of at least 96% of the whole processing time for any dataset while the rest of the phases aggregated as “computation time” use less than 4%.

Table 4

Time analysis on different datasets from SemTab challenges.

Dataset	# Mentions	Time (s)	Using cache time (s)	Entity retrieval time (s)	Entity retrieval time %	Computation time (s)	Computation time %
T2D	8079	2471	81	2390	96.72%	81	3.28%
HardTable 2021 R2	47440	8599	282	8317	96.72%	282	3.28%
HardTable 2021 R3	58949	80528	905	79623	98.88%	905	1.12%
SemTab 2020 R3	390457	14264	1446	12818	89.86%	1446	10.14%
2T 2020	667244	95190	1777	93413	98.13%	1777	1.87%
SemTab 2020 R4	994921	174046	4345	169701	97.50%	4345	2.50%

A further contribution from this paper consists of the definition of a format for a generic API specification useful for STI tasks. In Listing 5 the JSON format specification is reported; after the “name”, “dataset”, and “header” properties, there is the “rows” array. In this array, there is an object where the first element “idRow” is a numbered identifier for the row and the other element “data” contains the row content. The key “semanticAnnotations” allows to specify prior knowledge regarding the annotation of the table. For example, the “cta” key can be filled if the column types are already known. In the same way, also the “cta” and “cea” keys can be populated before the computation.

The experiments were conducted using 16 parallel processes: i) the ER is performed on a server with 40 CPU(s) Intel Xeon 4114 CPU @ 2.20GHz and 40GB RAM; ii) the ED is performed on a server with 32 CPU(s) Intel Xeon E5-2650 @ 2.00GHz and 94GB RAM.

Listing 5: API specification for s-elBat.

```
1  [
2  {
3    "name": "TEST1",
4    "dataset": "Dataset1",
5    "header": ["col1", "col2", "col3"],
6    "rows": [
7      {"idRow": 1, "data": ["Alabama", "United States", "5,024,279"]},
8      {"idRow": 2, "data": ["Colorado", "United States", "5,773,714"]},
9      {"idRow": 3, "data": ["North Carolina", "United States", "10,439,388"]}
10   ],
11   "semanticAnnotations": {
12     "cea": [],
13     "cpa": [],
14     "cta": []
15   },
16   "metadata": {
17     "column": [
18       {"idColumn": 0, "tag": "NE"},
19       {"idColumn": 1, "tag": "NE"},
20       {"idColumn": 2, "tag": "LIT", "datatype": "NUMBER"}
21     ]
22   },
23   "kgReference": "wikidata"
24 },
25 {
26   "name": "TEST2",
27   "dataset": "Dataset1",
28   "rows": [...],
29   ...
30   "kgReference": "wikidata"
31 }
32 ]
```

The tool and all the resources used for the experiments are released following the FAIR Guiding Principles⁷. s-elBat is released under the Apache 2.0 licence⁸.

4. Conclusion and Future Works

s-elBat, is a new approach that inherits and improves what was proposed by MantisTable. The results show an improvement in terms of the quality of the annotations and scalability. The formalisation of a STI API specification is an interesting addition to state-of-the-art. Regarding future developments, we want to discover a way to improve the computation time of entity retrieval. Another potentially interesting research would be to analyse how the features used for entity retrieval and disambiguation impact the results on different datasets.

Acknowledgement

This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070284 - *enRichMyData*.

⁷www.nature.com/articles/sdata201618

⁸bitbucket.org/disco_unimib/s-elbat/

References

- [1] S. Neumaier, J. Umbrich, J. X. Parreira, A. Polleres, Multi-level semantic labelling of numerical values, in: *The Semantic Web – ISWC 2016*, Springer International Publishing, Cham, 2016, pp. 428–445.
- [2] M. Marzocchi, M. Cremaschi, R. Pozzi, R. Avogadro, M. Palmonari, Mammotab: a giant and comprehensive dataset for semantic table interpretation, *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab2022* (in press).
- [3] M. Cremaschi, F. De Paoli, A. Rula, B. Spahiu, A fully automated approach to a complete semantic table interpretation, *Future Generation Computer Systems* 112 (2020) 478 – 500.
- [4] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: *The Semantic Web*, Springer International Publishing, Cham, 2020, pp. 514–530.
- [5] E. Jimenez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, V. Cutrona, Results of semtab 2020, *CEUR Workshop Proceedings* 2775 (2020) 1–8.
- [6] V. Cutrona, J. Chen, V. Efthymiou, O. Hassanzadeh, E. Jimenez-Ruiz, J. Sequeda, K. Srinivas, N. Abdelmageed, M. Hulsebos, D. Oliveira, C. Pesquita, Results of semtab 2021, in: *20th International Semantic Web Conference*, volume 3103, *CEUR Proceedings*, 2022, pp. 1–12.
- [7] V. Cutrona, M. Ciavotta, F. D. Paoli, M. Palmonari, ASIA: a tool for assisted semantic interpretation and annotation of tabular data, in: *Proceedings of the ISWC 2019 Satellite Tracks*, volume 2456 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 209–212.
- [8] M. Palmonari, M. Ciavotta, F. De Paoli, A. Košmerlj, N. Nikolov, Ew-shopp project: Supporting event and weather-based data analytics and marketing along the shopper journey, in: *Advances in Service-Oriented and Cloud Computing*, Springer International Publishing, Cham, 2020, pp. 187–191.
- [9] G. Weikum, X. L. Dong, S. Razniewski, F. M. Suchanek, Machine knowledge: Creation and curation of comprehensive knowledge bases, *Found. Trends Databases* 10 (2021) 108–490.
- [10] M. Kejriwal, C. A. Knoblock, P. Szekely, *Knowledge graphs: Fundamentals, techniques, and applications*, MIT Press, 2021.
- [11] M. Cremaschi, R. Avogadro, A. Barazzetti, D. Chierigato, Mantistable se: an efficient approach for the semantic table interpretation., in: *SemTab@ ISWC, 2020*, pp. 75–85.
- [12] W. Shen, J. Wang, J. Han, Entity linking with a knowledge base: Issues, techniques, and solutions, *IEEE Transactions on Knowledge and Data Engineering* 27 (2015) 443–460.
- [13] B. Hachey, W. Radford, J. Nothman, M. Honnibal, J. R. Curran, Evaluating entity linking with wikipedia, *Artificial Intelligence* 194 (2013) 130–150. *Artificial Intelligence, Wikipedia and Semi-Structured Resources*.
- [14] R. Avogadro, M. Cremaschi, F. D’adda, F. De Paoli, M. Palmonari, Lamapi: a comprehensive tool for string-based entity retrieval with type-base filters, in: *17th ISWC workshop on ontology matching (OM)*, in press.
- [15] V. Cutrona, F. Bianchi, E. Jiménez-Ruiz, M. Palmonari, Tough tables: Carefully evaluating entity linking for tabular data, in: *The Semantic Web – ISWC 2020*, Springer International Publishing, Cham, 2020, pp. 328–343.
- [16] R. Avogadro, M. Cremaschi, Mantistable v: A novel and efficient approach to semantic table interpretation., in: *SemTab@ ISWC, 2021*, pp. 79–91.