Knowledge-Based Systems 304 (2024) 112447

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Feature/vector entity retrieval and disambiguation techniques to create a supervised and unsupervised semantic table interpretation approach

Roberto Avogadro [a], Fabio D'Adda [b], Marco Cremaschi [b,*]

[a] SINTEF, Forskningsveien 1, Oslo, 0373, Norway

[b] Department of Informatics, Systems and Communication - University of Milano-Bicocca, Viale Sarca 336/14, Milan, I-20126, Italy

ARTICLE INFO

Keywords:
Semantic web
Knowledge base
Knowledge base construction
Knowledge base extension
Knowledge graph
Semantic table interpretation
Table annotation
Data enrichment
Tabular data

ABSTRACT

Recently, there has been an increasing interest in extracting and annotating tables on the Web. This activity allows the transformation of textual data into machine-readable formats to enable the execution of various artificial intelligence tasks, *e.g.*, semantic search and dataset extension. Semantic Table Interpretation (STI) is the process of annotating elements in a table. The paper explores Semantic Table Interpretation, addressing the challenges of Entity Retrieval and Entity Disambiguation in the context of Knowledge Graphs (KGs). It introduces LamAPI, an Information Retrieval system with string/type-based filtering and s-elBat, an Entity Disambiguation technique that combines heuristic and ML-based approaches. By applying the acquired know-how in the field and extracting algorithms, techniques and components from our previous STI approaches and the state of the art, we have created a new platform capable of annotating any tabular data, ensuring a high level of quality.

1. Introduction

In today's data-driven world, organisations and researchers encounter both opportunities and challenges. The availability of structured and unstructured data presents incredible potential for extracting valuable insights using Machine Learning (ML) models, which are adept at identifying patterns and relationships within the data. However, amidst these opportunities, it is crucial to recognise the role of data quality in determining the success of these approaches. Therefore, organisations need to prioritise the collection and management of data in order to obtain accurate and reliable datasets.

The creation and use of structured data in tables are common in many contexts [1]. For instance, in the current version of Wikipedia, it is possible to identify 2803424 tables. This implies that tables contain extensive data that can be utilised in data analytics or ML solutions.

In this scenario, the table-to-KG matching problem, also referred to as STI, has recently collected much attention in the research community [2–4] and is a key step to improve data quality [5], enrich data [6,7] and construct and extend Knowledge Graphs (KGs) from semi-structured data [8,9] (Fig. 1).

In relation to the possibilities that STI techniques enable, this topic is constantly evolving and gaining enthusiasm in time. The growing interest is proved by the international *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching* (SemTab) that has been proposed since 2018[1] [2–4]. This challenge is repeated annually. Over the years, several approaches, datasets, and related Gold Standards (GSs) have been released.

As previously stated, STI approaches make use of KG for the annotation task. When information is available in unstructured or semi-structured formats, such as tables or texts, identifying connections between strings (or mentions) in these sources and the entities they refer to in background KGs is crucial for integrating, enriching and extending the data and/or KGs. This process is called Entity Linking (EL), which may vary depending on the considered data formats but with some shared features.

For example, because of the ample entity search space, most of the approaches to EL include a first step where candidate entities for the input string are collected, *i.e.*, Entity Retrieval (ER) [10], and a second step where the string is disambiguated by eventually selecting one or none of the candidate entities, *i.e.*, Entity Disambiguation (ED) [11].

In most approaches, the ER returns a ranked list of candidates, while ED consists of re-ranking the input list. ED is at the heart of EL, with different approaches that leverage different kinds of evidence depending on the format and features of the input text [12]. However, the ER step is also significant, considering that its results define an upper bound for the performance of the end-to-end linking: if an entity is not among the set of candidates, it cannot be selected as the target

* Corresponding author.
E-mail address: marco.cremaschi@unimib.it (M. Cremaschi).

[1] cs.ox.ac.uk/isg/challenges/sem-tab

https://doi.org/10.1016/j.knosys.2024.112447
Received 19 December 2023; Received in revised form 2 August 2024; Accepted 28 August 2024
Available online 3 September 2024

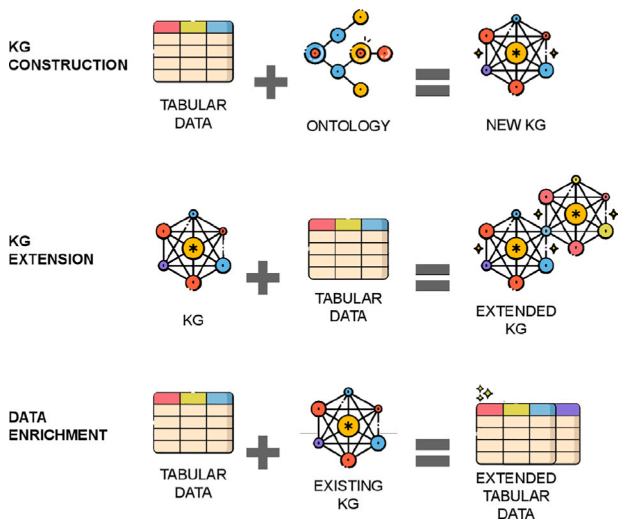


Fig. 1. Example of motivations and research values of STI.

for the link. Furthermore, while it is theoretically possible to scroll the list of candidates at any depth, maintaining acceptable efficiency levels requires cutting off the results of ER at a reasonable depth.

1.1. Entity retrieval

Approaches to entity searches can either resort to existing lookup APIs, *e.g.*, DBpedia SPARQL Query Editor,[2] DBpedia Spotlight[3] or Wikidata Query Service,[4] or use recent approaches to dense ER [13], when entities are searched in a pre-trained dense space, an approach becoming especially popular in EL for textual data. The APIs mentioned above provide access to the SPARQL endpoint since the elements are stored in Resource Description Format (RDF) format. Such endpoints are usually offered on local dumps of the original KGs to avoid network latency and increase efficiency. For instance, DBpedia can be accessed by OpenLink Virtuoso, a row-wise transaction-oriented RDBMS with a SPARQL query engine,[5] and Wikidata employs instead Blazegraph,[6] a high-performance graph database, providing RDF/SPARQL-based APIs. An issue faced with these solutions is the time required for downloading and setting up the datasets: Wikidata 2019 requires some days to set up since the full dump is about 1.1TB (uncompressed).[7] Moreover, writing SPARQL queries may be an issue since specific knowledge of the searched KG is required, besides the knowledge of the required syntax. Some limitations related to the use of these endpoints are:

- the SPARQL endpoint response time is directly proportional to the size of the returned data. As a consequence, sometimes it is not even possible to get a result because the endpoint fails for timeout;
- the number of requests per second may be severely limited for online endpoints (to ensure feasibility) or computationally too expensive for local endpoints (a reasonable configuration requires at least 64 GB of RAM with tons of CPU cycles);
- there are some intrinsic limits in the SPARQL language expressiveness (*i.e.*, full-text search capability, which is required for matching table mentions, can be obtained only with extremely slow “contains” keyword or “regex” queries[8]).

Regarding the approaches to dense ER, some limitations can be mentioned [14,15]:

- the results are strictly related to the type of representation used. Consequently, careful and tedious feature engineering is required when designing these systems;
- generalising the trained ER model to other KGs or domains is challenging due to the strong dependence on the specific KG and domain knowledge in the process of designing features;
- these systems depend excessively on external data, and the effectiveness of the algorithms is directly affected by the quality of the training data, and their utility is indispensably restricted.

Information Retrieval (IR) approaches based on search engines still provide valuable solutions to support ER, mainly because they do not require training. They work with any KG, and they easily adapt to changes in the reference KG. For these reasons, IR-based entity search has been used extensively, especially in table-to-kg matching [16–19]. The use of IR systems have been frequently left to custom optimisations and not adequately discussed or documented in scientific papers. As a result, researchers who seek to use such solutions must create them from the ground up, including data indexing techniques, query formulation, and service setup.

1.2. Entity disambiguation

Most of the ED techniques proposed in literature are based on the use of heuristics, *i.e.*, techniques that combine matching rules and similarity measures linearly or with different weights [20]. Simple matching rules based on strings, for instance, utilise measures such as Levenshtein distance or Jaccard similarity. Using these measures makes it possible to obtain a preliminary matching based solely on the values of the strings.

These methods are frequently adjusted manually by analysing the outcomes or comparing them with a Gold Standard (GS) to annotate domain-specific or case-specific data. However, this narrow focus introduces biases that make the methods less applicable in broader contexts. Identifying the optimal combination of matching and similarity features is a challenging task that demands significant effort.

As a result, it is essential to identify the configuration parameters which produce the highest quality semantic annotations by leveraging datasets and their associated GSs.

In most cases, it is difficult to gauge the effectiveness and adaptability of heuristics-based approaches since we cannot fully comprehend how they have been modified to perform better on particular datasets with intrinsic characteristics. On the other hand, when trained with good-quality data, supervised approaches tend to have higher generalisation due to the ability to implement model refresh cycles. However, as previously stated, they require training datasets.

1.3. Our contributions

Two main challenges regarding ER and ED are being faced. To summarise, the first challenge concerning ER is to efficiently handle data originating from KG to search over them and identify the set of the best candidates for a given mention. The second challenge concerns ED, and requires an algorithm capable of automatically identifying the features to use and determining their weights for the data to be processed.

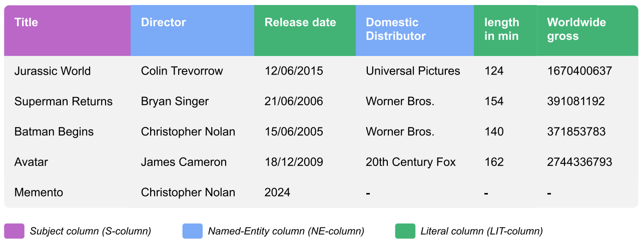
[2] dbpedia.org/sparql
[3] www.dbpedia-spotlight.org
[4] query.wikidata.org
[5] virtuoso.openlinksw.com
[6] blazegraph.com
[7] addshore.com/2019/10/your-own-wikidata-query-service-with-no-limits-part-1

[8] docs.openlinksw.com/virtuoso/rdfsparqlrulefulltext

For this reason, in this paper, we aim to present LamAPI (*La*bel *m*atching *API*) [21], a comprehensive tool for IR-based ER, augmented with type-based filtering features. The tool facilitates string-based retrieval, a process involving searching and extracting information from a dataset by matching strings of characters. When queries are submitted, the retrieval task entails identifying relevant entities that match the given query or criteria. Entity extraction from a KG goes beyond mere string matching; it also incorporates both hard and soft filters [22] based on the input entity type (*e.g.*, `rdf:type` for DBpedia and `Property:P31` for Wikidata). Type filters specify a concept defined in an ontology to filter out irrelevant entities. Two distinct filters are recognised: (i) hard-type filters and (ii) soft-type filters. The former aims to match specific types to significantly narrow down the set of entities, posing the challenge of identifying the most suitable concept to represent the searched entity in the ontology hierarchy. To address this, soft filters have been introduced to broaden the types included in the retrieval step, thereby increasing the number of extracted entities. Soft types can be incorporated by expanding types based on the hierarchy structure or predicting similar types based on shared features. Hard-type filters remove non-matching results, while soft-type filters promote or demote results when an exact match is not feasible. These filters are helpful to support either EL in texts (*e.g.*, by exploiting entity types returned by a classifier [23,24]), or in tables (*e.g.*, by exploiting a column type (`rdf:type`) to filter out irrelevant entities). While the approach is general, the tool provides EL support for semi-structured data. Therefore, our study evaluates different retrieval strategies with-/without filters on EL in the table-to-KG matching settings, considering two different large KG such as Wikidata and DBpedia. Finally, the tool also contains mappings among the latter two KGs and Wikipedia, thus supporting cross-KG bridges. The tool,[9] its repository[10] and the documentation[11] are publicly available. All the resources used for the experiments are released following the FAIR Guiding Principles.[12] LamAPI is released under the Apache 2.0 licence.[13]

Regarding ED, we propose s-elBat,[14] a supervised and semi-supervised STI approach. s-elBat employs several techniques to consider all of the STI challenges that will be discussed in Section 2. A detail on how s-elBat addresses these challenges will be proposed in Section 8. s-elBat is a new approach that inherits and improves what has been proposed by our previous STI approaches [25–27]. The experiences acquired with those tools (Section 3.1) and their participation in the various editions of the SemTab challenge led to the definition of new techniques for a feature vector-based entity disambiguation approach, which as mentioned before, combines both heuristic and ML approaches to achieve accurate disambiguation results. The s-elBat tool,[15] its repository[16] and the documentation[17] are publicly available. s-elBat is released under the GNU Affero General Public License.[18]

Comprehending table data can be straightforward without effectively visualising the output results. A well-designed and intuitive interface is essential for inspecting algorithm results. For this reason, we introduce SemTUI (*Sem*antic *T*able *U*ser *I*nterface). This framework provides a user interface to streamline interaction between users and both the LamAPI and s-elBat systems. Utilising SemTUI, users can delve into the functionalities of LamAPI, enabling the exploration of candidate mentions derived from a KG. Similarly, SemTUI assists in interfacing with s-elBat, allowing users to initiate the table annotation process. Upon completing this process, they can explore the results in detail through the UI, accessing a range of information, including names, descriptions, and types associated with each candidate. It also presents confidence scores, providing an option to filter results accordingly. Both SemTUI repository[19] and documentation[20] are publicly available.



Title	Director	Release date	Domestic Distributor	length in min	Worldwide gross
Jurassic World	Colin Trevorrow	12/06/2015	Universal Pictures	124	1670400637
Superman Returns	Bryan Singer	21/06/2006	Worner Bros.	154	391081192
Batman Begins	Christopher Nolan	15/06/2005	Worner Bros.	140	371853783
Avatar	James Cameron	18/12/2009	20th Century Fox	162	2744336793
Memento	Christopher Nolan	2024	-	-	-

🟪 *Subject column (S-column)* 🟦 *Named-Entity column (NE-column)* 🟩 *Literal column (LIT-column)*

Fig. 2. Example of a well-formed relational table.

This paper is organised as follows: Section 2 defines the STI process formally and summarises the challenges. Section 3 proposes a detailed examination of the ER and ED techniques used by all STI approaches in the state of the art. Sections 4 and 5 describe LamAPI, and s-elBat respectively. Section 4.3 introduces the GSs, the configuration parameters, and the evaluation results. SemTUI, our User Interface (UI) for managing LamAPI and s-elBat is described in Section 7. Eventually, we conclude this paper and discuss the future direction in Section 8.

2. Definition and challenges of semantic table interpretation

In this Section, we formally describe the task we are going to solve and list the key challenges related to STI tasks. The input of STI is (i) a *well-formed and normalised* relational table (*i.e.*, a table with headers and cells filled with string values that we call *mentions* in this paper), as in Fig. 2, and (ii) a reference *KG* that describes real-world entities in the domain of interest (*i.e.*, a set of concepts, datatypes, properties, instances, and relationships among them) (Fig. 3). The output returned is a semantically annotated table, as in the example of Fig. 4.

The STI process is typically broken down into the following four tasks:

- Column Type Annotation (CTA): the *CTA* task concerns the prediction of the semantic types (*i.e.*, KG classes) for every given table column in a table;
- Column Property Annotation (CPA): the *CPA* task concerns the prediction of semantic properties (*i.e.*, KG properties) that represent the relationship between some pair of columns;
- Cell Entity Annotation (CEA): the *CEA* task aims to predict the entity (*i.e.*, instances) that a cell in table represents;
- Cell-New Entity Annotation (CNEA): the *CNEA* task aims to predict which cell in the table represents an entity that does not occur in the KG and should be therefore labelled as Not In Lexicon (NIL).

In Appendix A, a formalisation of STI is proposed.

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) *Heterogeneity of domains and data distributions*: the tables may cover information that refers to very different domains (*e.g.*, Geography *vs* Sports); the specificity of the table content may vary significantly (from a table with basic information about most famous mountains, like Fig. 2 to a table that contains the composition of the rocks of this mountains[21]).

9 lamapi.datai.disco.unimib.it
10 github.com/unimib-datAI/lamAPI
11 unimib-datai.github.io/lamapi-docs/
12 www.nature.com/articles/sdata201618
13 apache.org/licenses/LICENSE-2.0
14 The name comes from taBle-s and Semantic Entity Linking to BAtch Table.
15 selbat.datai.disco.unimib.it
16 github.com/unimib-datAI/s-elbat
17 unimib-datai.github.io/s-elbat-docs/
18 www.gnu.org/licenses/agpl-3.0.en.html
19 github.com/I2Tunimib
20 i2tunimib.github.io/I2T-docs
21 en.wikipedia.org/wiki/List_of_rock_formations

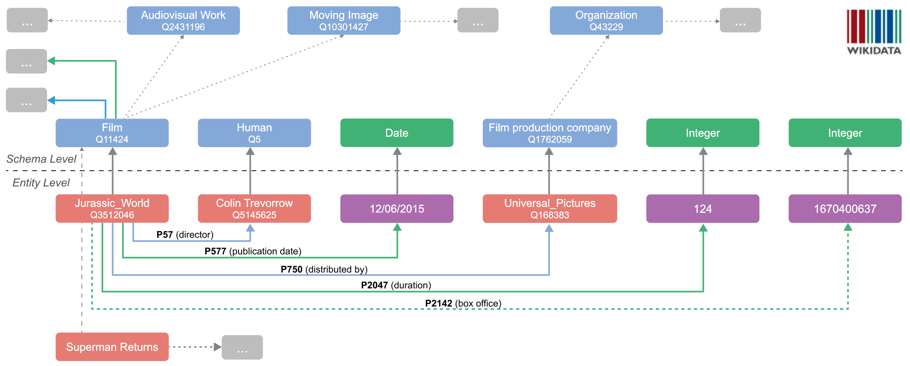
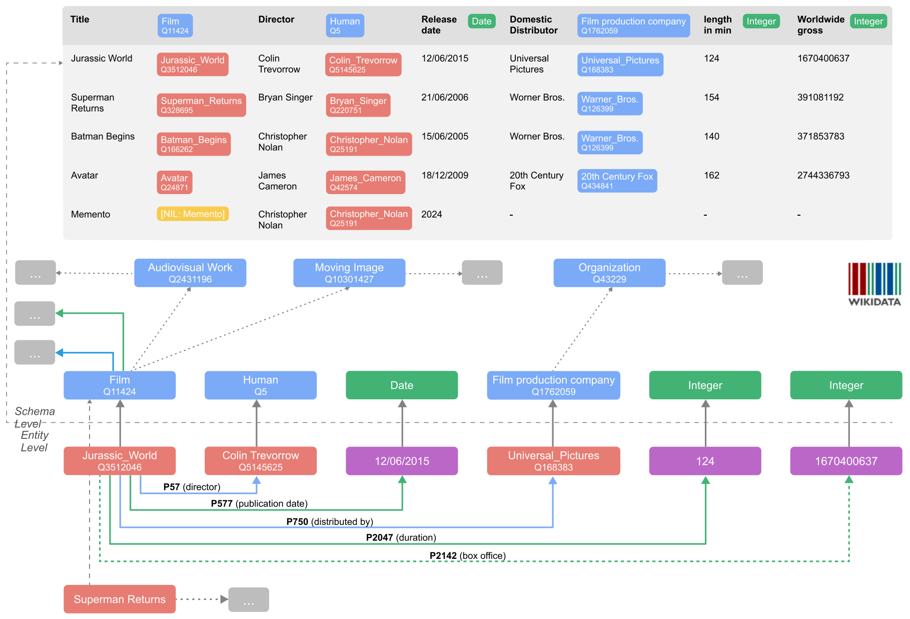


Fig. 3. A sample of a Knowledge Graph.



Title	Film Q11424	Director	Human Q5	Release date	Date	Domestic Distributor	Film production company Q1762059	length in min	Integer	Worldwide gross	Integer
Jurassic World	Jurassic_World Q3512046	Colin Trevorrow	Colin_Trevorrow Q5145625	12/06/2015		Universal Pictures	Universal_Pictures Q168383	124		1670400637	
Superman Returns	Superman_Returns Q328695	Bryan Singer	Bryan_Singer Q220751	21/06/2006		Worner Bros.	Warner_Bros. Q126399	154		391081192	
Batman Begins	Batman_Begins Q166262	Christopher Nolan	Christopher_Nolan Q25191	15/06/2005		Worner Bros.	Warner_Bros. Q126399	140		371853783	
Avatar	Avatar Q24871	James Cameron	James_Cameron Q42574	18/12/2009		20th Century Fox	20th Century Fox Q434841	162		2744336793	
Memento	[NIL: Memento]	Christopher Nolan	Christopher_Nolan Q25191	2024		-		-		-	

Fig. 4. The table in Fig. 2 annotated with the KG.

(ii) *Matching tabular values against the KG*: matching the values in the table to the data in the KG is a common method for collecting evidence to interpret the table. The mention in the table is frequently different from the label of the entity in a KG (*i.e.*, use of acronyms, aliases, and typos); for example, *High Tauern* refers to the *Johannisberg mountain (High Tauern)* entity in DBpedia[22]; literal values, *e.g.*, height of the mountains, may be different, *e.g.*, because outdated, measured differently and so on.

(iii) *Disambiguation of named entities*: the KG may contain many entities with similar or even equal names (homonyms) that may belong to different or the same type. For example, the mention Mont_Blanc in Fig. 2, which refers to the famous mountain located on the French-Italian border,[23] matches labels of different entities associated with different types, including a tunnel, a poem, and a dessert[24]; the same mention matches also the label of a mountain on the Moon.[25]

(iv) *NIL-mentions*: the approach must also consider the NIL-mentions, that are strings that refer to entities for which a representation has not yet been created within the KG;

(v) *Choosing the most appropriate types and properties*: the KG may contain hundreds or thousands of types and properties to choose from for annotating columns and column pairs; entities are classified with multiple types, which may reflect different levels of specificity (*e.g.*, if we consider the subclass of the relation between types, Mont Blanc is a mountain, a summit, a pyramidal peak, an elevation, a landform, a geographical feature, a geographic location and more in Wikidata). The classification may be more or less complete depending on the specific entity; deciding which type or which set of types better describes a set of entities in a column is not trivial. Also, several column pairs could be potentially related, and several properties exist in KG that have similar meanings (also, in this case, properties can be organised into taxonomies and have different levels of specificity) [28]; selecting column pairs for annotation and pinpointing the most suitable property is not straightforward either.

3. Related work

This section reviews research on STI, including methodologies and advances in understanding structured tabular data. This introduces our innovative contributions to improve semantic comprehension of tables in our work. 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]:

[22] dbpedia.org/page/Johannisberg_(High_Tauern)
[23] dbpedia.org/page/Mont_Blanc
[24] en.wikipedia.org/wiki/Mont_Blanc_(disambiguation)
[25] dbpedia.org/page/Mont_Blanc_(Moon)

1. Entity Retrieval (ER). In this module, for each entity mention $m \in M$, irrelevant entities from the KG are filtered out to return a set E_m of candidate entities that may be referred to by m. To achieve this goal, we have applied cutting-edge techniques, such as those relying on name dictionaries, expanding surface forms within the document, and employing 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 candidates. This involves re-ranking based on additional information (*e.g.*, contextual information) besides the simple textual mention m used in the ER module.

According to several experiments conducted in the latest research [29], the main aim of the ER module (also referred to as candidate generation or *lookup*) is to guarantee the inclusion of the accurate entity within the retrieved set, facilitating its identification by the ED module. Therefore, one of the main contributions of this work is to discuss retrieval configurations, *i.e.*, query and filtering strategies, for better entity retrieval.

When a query is made, the system must generate a set of potential candidates for each cell that might meet the query's requirements. Candidates can be generated using numerous strategies, including semantic parsing, entity recognition, and information retrieval. The system might use ML models trained on large data corpora to identify potential candidates based on contextual patterns.

The ER can be divided into four methods: (a) ***custom index*** [16,18, 26,27,30–45], (b) ***external lookup services*** [25,33,38,39,43,46–67], (c) ***hybrid*** (both custom index and external lookup services) [33,38,39,43], and (d) ***other*** [48,63].

Custom index involves creating a specialised index for specific requirements or use cases. When building a custom index, there is flexibility to define mappings, analysers, and other configurations based on specific requirements. Elasticsearch[26] is a popular search and analytics engine that is both robust and scalable. It operates on a document-oriented model, storing and organising data as JSON documents. Several approaches rely on Elasticsearch for the lookup sub-task [18,26, 27,33,35–40,42,44,45]. Limaye et al. [30] introduce a method that relies on a catalogue of types, entities and relations. Entities in the catalogue are linked to lemmas, which are canonical strings extracted from Wikipedia, or synset names from WordNet.[27] Syed [31] develops a hybrid KG that combines both structured and unstructured information extracted from Wikipedia, augmented by RDF data from DBpedia and other Linked Data. The system, named Wikitology, uses an IR index (Lucene) to represent Wikipedia articles. The same technique is used by Mulwad et al. [46,47,49]. Efthymiou et al. [32] employ a lookup-based method to establish connections, using the limited entity context available in Web tables to identify correspondences with the KG. The approach builds a specialised search index called FactBase using Wikidata. This index includes entities paired with corresponding *IDs* and textual descriptions.

In ColNet [34] a lookup step is performed to retrieve entities from the KG by matching cells based on entity labels and anchors (*e.g.*, Wikipedia link). This is achieved using a lexical index containing terminology and assertions from the KG. MTab4Wikidata [41], a variant of MTab [55], focuses on annotating cells to Wikidata entities. It begins by downloading and extracting a Wikidata dump to create an index with hash tables. The lookup process is performed using a fuzzy search. The result is a ranked list of entities based on edit distance scores. In the latest version of MTab 2021 [16], a WikiGraph index is established, combining Wikidata, Wikipedia, and DBpedia. The lookup employs Keyword Search, Fuzzy Search, and Aggregation Search. GBMTab [43] distinguishes between entity extraction from Wikidata and DBpedia to generate candidate entries. The solution is limited to DBpedia and builds an index using hash tables. It then calculates the Levenshtein distance to determine the string similarity between mentions and entities.

The second method employed for candidate generation uses ***external lookup services***. This approach involves using a separate service or system to retrieve specific information or data through lookup or queries. The external lookup service usually uses entity recognition, entity disambiguation, or semantic matching techniques. It may consider factors like textual similarity, context, or other relevant information. In the STI context, many services can be used to extract a set of possible entities given a string as input. The service choice is based on the specific requirements and context, including the KG used to annotate the entities. To annotate tables cells with DBpedia entities, most approaches use associated services, such as DBpedia API [33,55], DBpedia Lookup Service [55,56,64] and DBpedia Spotlight [56,67]. Similarly, for Wikidata, the following services are employed: Wikidata API [33,38,60,63,67], Wikidata Lookup Service [55,57–59,61,64,68] and Wikidata CirrusSearch Engine [33]. Other services used for the lookup sub-task are Wikipedia API [33,55,63,65], MediaWiki API [39, 43] and Wikibooks [63]. Instead of directly employing lookup services, some approaches execute SPARQL queries to retrieve and manipulate data stored in RDF format. This method is the default way to obtain information from triple stores. For approaches not explicitly mentioning the lookup service, it is assumed that SPARQL is used. For example, SPARQL queries are used to retrieve entities from YAGO [50], DBpedia [25,51–54,56,58,59,66,68], and Wikidata [58,59,62,63,66,68]. Furthermore, other sources such as SearX [63] and Probase [48] are used, with pattern matching utilised to extract triples.

As described, ED is the process of resolving ambiguous mentions to entities. Using the same name to refer to several entities in a table might lead to uncertainty. Entity disambiguation in STI seeks to recognise and clarify these entity mentions, ensuring that each mention is correctly linked to the appropriate entity. This task employs various techniques: (a) ***embedding*** [32,33,53,61,65,67,69], (b) ***similarity*** [18, 25–27,30,35,37,38,40,42,44,52,54,56,58,63,64], (c) ***contextual information*** [16,18,26,27,31,32,34,36,38–42,44,45,51,55,57,60,66,69], (d) ***ML techniques*** [38,46,65], (e) ***probabilistic models*** [35,43,47,49,50], (f) ***LLMs-based*** [45,70–75], and (g) ***other*** [48,76].

In graphs and natural language, ***embedding*** involves representing nodes or words as dense vectors in a continuous vector space. These embeddings capture semantic and structural relationships between nodes or words, allowing ML models to perform tasks such as node classification, link prediction, document similarity, sentiment analysis, and more [77]. Embedding techniques are used to create vector representations of entities [32,33,53,61,65,67,69]. Each solution aims to gather context information about entities in the KG and incorporate it into vector representation.

The entity disambiguation sub-task frequently involves calculating ***similarities*** among textual data, which is typically employed by lookup services to retrieve a ranked list of candidates. This process often involves selecting the best candidate based on the similarity of the entity label and mention. Various approaches use similarity, such as, Levenshtein distance [18,25–27,37,38,40,42,44,54,56,63], Jaccard similarity [27,30,35,44], Cosine similarity [30,64], and similarity based on Regular Expressions (Regex) [40]. Limaye et al. [30] in addition to Jaccard applies also the TF-IDF.[28] TableMiner+ [52] measures the similarity between the bag-of-words (bow) representation of the entity with the bow representations of different types of cell contexts, such as row content and column content.

[26] www.elastic.co
[27] wordnet.princeton.edu

[28] The weight assigned to a term in a document vector is the product of its term frequency (TF) and inverse document frequency (IDF).

During the CEA task, ***contextual information*** considers the surrounding context of a table cell, such as neighbouring cells, column headers, or header rows. This contextual information provides additional clues or hints about the meaning and intent of the mention. A system can obtain a better knowledge of the semantics of a cell by analysing its context and making more precise annotations. Contextual information at column and row level is usually provided by CTA and CPA tasks. However, some approaches also consider column types and properties to disambiguate entities, even if these tasks are not expressly stated. Column types are used to disambiguate entities by assuming that entities in a column share the same type. Many approaches rely on this assumption to perform this step [31,32,34,36,38,51,55,60,66]. [31,34,66] limit the number of candidate entities by executing a new lookup query that includes predicted types. [32,36,38,51,55,60] refine the candidates list by filtering out entities that do not match the predicted type at the column level. Similarly, some approaches also consider another assumption: contextual information from CPA at the row level helps to improve understanding of the data in its larger context [16,40,41,45]. For example, [40,45] consider the semantic relations between columns by boosting the scores for each candidate entity when the relation is found. On the other hand, [16,41] calculate context similarity between candidate triples and table row values by ranking entities based on this score. They then select the candidate with the highest context similarity as the final annotation. Furthermore, most approaches adopt a hybrid solution for the disambiguation sub-step, considering the information provided by CPA and CTA tasks [18,26,27,39,42,44,57,69].

Other methods that can be employed are ***ML techniques***. These strategies typically involve training a ML model on a labelled dataset where cells are marked with their respective entities. The model learns patterns and relationships between the cells content and their associated entities. To predict the most appropriate entity, ML techniques consider various cell features, such as the textual content, context, neighbouring cells, and other relevant information. Several ML techniques can be employed to perform the disambiguation task, such as Support Vector Machine (SVM) [46], Neural Network (NN) [38], and Random Forest [65].

Probabilistic models apply probability theory to express and reason about uncertainty. These models vary in their representation of dependencies and use diverse graphical structures. Several Probabilistic Graphical Model (PGM) can be also used to resolve the disambiguation task, such as Markov model [43,47,49,50] or Loopy Belief Propagation (LBP) [35].

The advent of ***Large Language Models (LLMs)*** has led to a new category of approaches for table annotation. Based on the architecture structure of LLMs, these approaches can be categorised into three groups: (i) encoder–decoder LLMs, ii) encoder-only LLMs, and (iii) decoder-only LLMs [78]. Indeed, shortly after the first edition of SemTab, some works [70–72] applied encoder-only LLMs to table interpretation. Although they did not participate in or compare with the SemTab challenge, they created a different experimental setting. During the SemTab2022 instead, a BERT-based [79] model was combined with a more traditional approach [45]. More recently, after the release of GPT-3.5 [80] and open-source decoder-only LLMs such as LLAMA [81] and LLAMA 2 [82], some works have begun applying encoder-based LLMs to table interpretation [73,74]. In SemTab2023, a new decoder-only model was presented that uses BERT [75]. Starting from encoder-based approaches, Ditto [70] utilises Transformer-based language models to perform a slightly different task; in fact, the goal is entity-matching between different tables. TURL [71] leverages a pre-trained TinyBERT [83] model to initialise a structure-aware Transformer encoder. Doduo [72] performs CTA using a pre-trained language model, specifically fine-tuning BERT model on serialised tabular data. DAGOBAH SL 2022 [45] employs an ELECTRA-based cross-encoder, a variant of the BERT model. The Cross Encoder takes a concatenated input, including left-side table headers, the target table header, right-side table headers, and the entity description. TorchicTab [75] is composed of two sub-systems: TorchicTab-Heuristic and TorchicTab-Classification. The classification model utilises Doduo [72].

Regarding decoder-based approaches, TableGPT [73] performs several tasks, including entity linking using GPT. TableLlama [74], performs CEA, along with several other tasks, creating a multi-task dataset for tabular data, in which the entity linking sub-dataset derives from the TURL [71] dataset, and using it to fine-tune LLama2 [82].

Other approaches cannot be categorised in one of the previous groups. For instance, Munoz et al. [76] propose an approach to extract RDF triples from Wikitables by linking each cell to DBpedia entities. The process involves following internal links within Wikipedia tables, as they can be directly mapped to DBpedia. Wang [48] describes the process of understanding a table using the Probase knowledge API. Eventually Kim et al. [62] remove candidates considering the content unrelated to the annotation.

Our proposal integrates various cutting-edge techniques outlined in the current state of the art. Initially, the ER process depends on the utilisation of **custom indexes** constructed from DBpedia and Wikidata, which facilitate the retrieval task. The subsequent phase is executed by s-elBat, incorporating the methodologies proposed in this section. This system comprises two distinct versions, each of them uniquely designed and implemented. The first version operates in a **semi-supervised** and **heuristic manner**, considering contextual information within the table to execute all tasks related to STI, such as CEA, CPA, and CTA. In contrast, the supervised version replaces the ED step by employing ML techniques for the final re-ranking task.

3.1. From MantisTable to s-elBat

The exploration of STI has guided us to formulate and refine a set of tools, continuously enhancing them based on the insights gained (Fig. 5). The first iteration of MantisTable [25] marked the start of a comprehensive pipeline for table annotation. This approach depended mostly on external services like the Oxford English Dictionary[29] during table pre-processing and SPARQL queries, which are performed by SPARQLWrapper,[30] to retrieve entities from the KG. To assess the quality of semantic annotations for each STI task, an automatic evaluation tool, STIlTool [84], was developed. Subsequent versions, starting with MantisTable SE [26], saw the refinement of data pre-processing by incorporating Regex rules, which were capable of identifying various datatypes in tables. The lookup search was also enhanced by integrating a built-in information retrieval service (LamAPI v.1). The evolution continued with MantisTable V [27], which introduced a more robust CEA algorithm. The programme scored each extracted candidate based on string similarity and contextual information. Subsequently, MammoTab [1], a dataset composed of $1M$ Wikipedia tables extracted from over $20M$ Wikipedia pages and annotated through Wikidata, was released. Eventually LamAPI [21] tool was consolidated, incorporating new services to support the entire STI processes. Given the recent proliferation of STI papers in the state of the art, a comprehensive survey was conducted, providing a conclusive overview of each approach presented in the current state of the art.

In this paper, we have shared what we have learned by analysing the state of the art as well as our own experiences in this subject. In particular, by using the acquired knowledge and extracting algorithms, techniques, and components from earlier tools, we have created a new platform capable of annotating any tabular data while maintaining a high degree of quality.

[29] www.oed.com/information/using-the-oed/
[30] sparqlwrapper.readthedocs.io/en/latest/

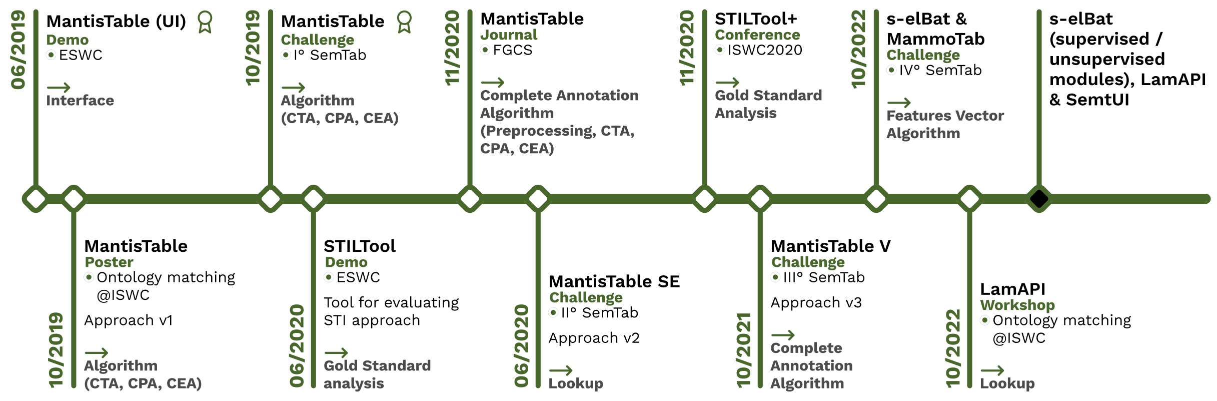


Fig. 5. Evolution of our STI systems.

4. LamAPI

When dealing with information in unstructured or semi-structured formats, such as tables or text, a crucial process involves identifying connections between the strings within these sources and the corresponding entities within KG. Additionally, types and relationships stored as triples in a KG, if extracted, represents an excellent source of knowledge, especially for disambiguation task.

LamAPI is an ER system developed to query and filter entities in a KG by applying complex string matching algorithms. This task, as described in Section 4.1 and referred as "indexing", is crucial for each EL pipeline. The purpose of the system is to support EL in each phase of its pipeline. For this reason LamAPI also provides knowledge extraction services to retrieve entity triples information in different formats.

The current version of the system integrates DBpedia (v. 2016–10 and v. 2022.03.01) and Wikidata (v. 20220708), which are the most popular KGs adopted also in the SemTab challenge.[31]

4.1. Knowledge Graphs indexing

DBpedia, Wikidata and the like are very large KGs that require an enormous amount of time and resources to perform ER. While turtle format[32] is excellent for representing relationships among entities, it is not well-suited for ER algorithms, as the triples in turtle format, represented by URIs, are not designed or optimised for ER tasks. They might be long, cryptic, or not easily interpretable, making it challenging for traditional entity retrieval algorithms to efficiently process and index them. This issue has been tackled by devising a more condensed data representation by utilising MongoDB collections, which can be indexed for swift retrieval of the intended data.

For each indexed KG, the relative dump has been downloaded and parsed to store all triples in a local copy. For Wikidata, a single file named "latest-all.json.bz2"[33] of size 71 GB has been parsed, while for DBpedia multiple turtle files have been parsed to create a complete dump of the KG. Subsequently, an ElasticSearch[34] index has been constructed, leveraging an engine designed to search and analyse extensive data volumes in nearly real-time swiftly. These customised local copies of the KGs are then used to create endpoints to provide ER services. The advantage is that these services can work on partitions of the original KGs to improve performance by saving time and using fewer resources.

31 www.cs.ox.ac.uk/isg/challenges/sem-tab
32 www.w3.org/TR/turtle
33 doc.wikimedia.org/Wikibase/master/php/docs_topics_json.html
34 www.elastic.co

4.2. List of services

LamAPI offers a comprehensive range of services designed to fully support algorithms related to EL at every stage of their pipeline. This Section focuses on services provided by LamAPI and used by s-elBat in its entire execution process. While there are various services available, only those directly relevant to s-elBat's EL task are discussed. These services include: (i) **Lookup**, (ii) **Types**, (iii) **Literals**, (iv) **Properties**, (v) **Objects**, (vi) **Labels**, (vii) **Literal Recogniser**, and (viii) **FastText**.

(i) Lookup: given a string input, the service retrieves a set of candidate entities from the reference KG. The request can be qualified by setting some attributes:

- **limit:** an integer value that specifies the number of entities to retrieve. The default value is 100, and it has been empirically demonstrated how this limit allows a good level of coverage. The experiments are presented in Section 4.3;
- **kg:** specifies which KG and version to use. The default is dbpedia_2022_03_01, and other possible values are 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 an higher value could not be very effective in terms of spelling errors. For instance "albert einstein" using n-grams equal to 3 is split in ['alb', 'lbe', 'ber', 'ert', …]. Ngram value is set to false by default;
- **types:** this parameter allows the specification of a list of types (*i.e.*, `rdf:type` for DBpedia and `Property:P31` for Wikidata) associated with the input string to filter out the entities to retrieve. The list of types must be specified by listing all concepts separated by a single space as parameter (*e.g.*, "Scientist Philosopher" for searching entities of type "Scientist" or "Philosopher"). This attribute plays a key role in re-ranking the candidates, allowing a more accurate search based on input types.

The following example discusses the difference between a SPARQL query and the LamAPI Lookup service. Listings 1 and 2 show a search using the mention "Albert Einstein". The evidence is that LamAPI syntax is simpler than the one employed in SPARQL. The Lookup service allows for managing the presence of misspelled mentions and provide a ranking of the resulted candidates by considering a Levenshtein distance score.

Listing 1: Search with SPARQL.

```
    select distinct ?s where {
    ?s ?p ?o .
    FILTER( ?p IN (rdfs:label)).
    ?o bif:contains "Albert Einstein".
    }
    order by strlen(str(?s))
    LIMIT 100
```

Listing 2: Example of LamAPI Lookup.

```
/lookup/entity-retrieval?
name="Albert Einstein"&
limit=100&
token=insideslab-lamapi-2022&
kg=dbpedia_2022_03_01&
fuzzy=False&
ngrams=False
```

Examples of results using the input string "Albert Einstein" are shown in Listing 3 and Listing 4, referred to Wikidata and DBpedia respectively. Each candidate entity is described by the following fields:

- **id**: the unique identifier *id* provided by the KG;
- **label**: the string label name reporting the label of the entity;
- **type**: a set of types associated to the entity, where each one is described by its unique identifier;
- **description**: an optional description of the entity (*i.e.*, DBpedia does not provide descriptions, while Wikidata does);
- **score**: a score represented by the edit distance measure (Levenshtein distance) between the input textual mention and the entity label.

Listing 3: Lookup: returned data from Wikidata for the label "Albert Einstein".

```
{
"id": Q937
"label": Albert Einstein
"description": German-born ...
"type": Q19350898 Q16389557 ... Q5
"score": 1.0
},
{
"id": Q356303
"label": Albert Einstein
"description": American actor ...
"type": Q33999 Q2526255 ... Q5
"score": 1.0
}
```

Listing 4: Lookup: returned data from DBpedia for the label "Albert Einstein".

```
{
"id": Albert_Einstein
"label": Albert Einstein
"description": ...
"type": Scientist Animal ...
"score": 1.0
},
{
"id": Albert_Einstein_ATV
"label": Albert Einstein ATV
"description": ...
"type": SpaceMission Event ...
"score": 0.789
}
```

The score provides a candidate ranking that can be used by the ED module (Section 5) for a straightforward selection of the actual link.

(ii) Types: Given a list of unique *IDs* of DBpedia or Wikidata entities, it retrieves a list of types for each entity. This service represents the triple in which the property is always *rdf:type* in DBpedia and *P31* in Wikidata.

(iii) Literals: Given a list of unique *IDs* of DBpedia or Wikidata entities, it retrieves all the triples associated with each entity specified, where the object in the triple is a literal, such as strings, numbers, and dates.

(iv) Predicates: Given a list of entity pairs, it retrieves all the properties that bind the entities to determine whether those entities are related or not, and if they are related, it determines which properties are involved.

(v) Objects: Given a list of unique *IDs* of DBpedia or Wikidata entities, it retrieves a list of triples for each entity where the object is another entity of the KG.

(vi) Labels: Given a list of unique *IDs* of DBpedia or Wikidata entities, it retrieves a list of LABELS and ALIASES for each entity.

(vii) Literal recogniser: Given a list composed of a set of strings, the endpoint returns the types of those literals. The list of recognised literals is specified in Listing 5.

Listing 5: Literals recognised by LamAPI.

```
DATES:
145 bc, 145.bc, 145,bc
1997-08-26, 1997.08.26, 1997/08/26
26/08/1997, 26.08.1997, 26-08-1997
26/08/97, 26.08.97, 26-08-97
august 26 1997, august.26.1997, august,26,1997
26 august 1997, 26.august.1997, 26,august,1997
1997 august 26, 1997,august,26, 1997.august.26
1997 26 august, 1997,26,august, 1997.26.august
august 1997, august.1997, august,1997
1997 august, 1997.august, 1997,august
1997-2022, 1997-present, 1997-now

NUMBERS:
2,797,800,564, 2.797.800.564
200,797,800, 200.797.800
1997, 1345, 26, 1
+/- 34, +/- 34657
25 thousand, 25 million, 25 billion, 25 trillion
2 km, 2 km2, 2 cm, 2 cm2, 2 mm, 2 mm2, 10 sq, 10 ft
2,8, 2.8
+/- 5e+/-10

OTHERS
https://unimib.it/
mario.rossi@gmail.com
12pm, 2.30 am, 12.30pm, 12:30pm
```

(viii) FastText: Given an array of strings as input, the endpoint generates and returns the corresponding embedding representations of length 300 using *Fasttext*[35] [85] vectors. The decision to use 300 dimensions is likely a balance between capturing enough semantic information to represent words effectively and keeping the computational requirements reasonable. Increasing the dimensionality may improve the model's ability to capture subtle semantic relationships between words, but it also comes with increased computational cost and memory requirements [85].

Listing 6: Fasttext: DBpedia vector returned for "Alan_Turing" entity.

```
Request:
{
  "json": [
    "Alan Turing",
  ]
}
Response:
{
  "Alan Turing": [
    0.00280323950573802,
    0.010494967922568321,
    -0.00392009224742651,
    -0.0017100191907957196,
    -0.04271041601896286,
    -0.040172189474105835,
    -0.017303353175520897,
    0.0086875194683671,
    -0.007565824780613184,
    ...
  ]
}
```

Listing 7: Fasttext: Wikidata vector returned for "Albert_Einstein" entity.

```
Request:
{
  "json": [
    "Albert Einstein",
  ]
}
Response:
```

[35] github.com/facebookresearch/fastText

Table 1
Statistics of the datasets used to extract coverage data. '—' indicates unknown.

GS		Tables	Cols (min | max | $\bar{x}$)	Rows (min | max | $\bar{x}$)	Classes	Entities	Pred.	KG
T2Dv2 [86]		234	1,2K (1 | 30 | 4,52)	2,8K (1 | 5K | 84,55)	39	–	154	DBpedia
Tough Table (2T) [87]		180	194K (1 | 8 | 4,46)	802 (6 | 15,5K | 108K)	540	667K	0	Wikidata DBpedia
SemTab2019 [87]	R3	2,1K	10,8K (4 | 8 | 4,51)	153K (6 | 207 | 71,69)	5,7K	407K	7,6K	DBpedia
	R4	817	3,3K (4 | 8 | 4,36)	51,4K (6 | 198 | 63,73)	1,7K	107K	2,7K	
HardTable2021 [4]	R2	1,7K	5,6K (2 | 7 | 3,19)	29,3K (5 | 58 | 17,73)	2,1K	47,4K	3,8K	Wikidata
	R3	7,2K	17,9K (2 | 5 | 2,48)	58,9K (5 | 21 | 9,18)	7,2K	58,9K	10,7K	

```
{
  "Albert Einstein": [
    -0.03460921719670296,
    0.01696068048477173,
    0.015138840302824974,
    0.04197229444980621,
    -0.020079581066966057,
    -0.04950999841094017,
    -0.012100357562303543,
    -0.01173518318682909,
    0.01281578280031681,
    ...
  ]
}
```

4.3. Evaluation

In this Section, the evaluation is reported to measure the quality of the LamAPI output. Specifically, the distribution of correct candidates is examined considering various datasets available in the SemTab challenge (Table 1). The datasets used for the analysis are:

- **T2Dv2** [86]: the T2Dv2 GS consists of manually annotated row-to-instance, attribute-to-property, and table-to-class correspondences;
- **SemTab 2019** [88]: a group of benchmark datasets for each round of the challenge without extensive human annotation. It has been designed by developing an automated data generator that creates tabular data given a SPARQL endpoint. The idea is to create tabular data similar to tables found on the Web ensuring a reasonable diversity in size and coverage of classes and properties from various domains. Specifically *SemTab_R3 2019* and *SemTab_R4 2019* datasets has been used from SemTab challenge 2019;
- **Tough Tables (2T)** [87]: the dataset has been built with high-quality tables, including cells that are not immediately linkable due to factors such as ambiguous names, typos, and misspelt entity names;
- **HardTable-2021** [4]: this is a synthetic dataset with tables generated automatically using SPARQL queries. Datasets used from HardTable-2021 are respectively *HardTable_R2 2021* and *HardTable_R3 2021*.

Choosing the right datasets is pivotal for testing the LamAPI service and developing the s-elBat disambiguation algorithm through ML techniques in subsequent sections. These datasets must accurately represent the relevant features and patterns within the problem domain. Ensuring the datasets are extensive and diverse is essential, encompassing a broad spectrum of scenarios and variations. This variety helps the model generalise unfamiliar data effectively and handle various situations. In a real context, the data used for an ML task are not error-free. For this reason, the **2T** dataset has been employed to assess the performance of both LamAPI and s-elBat to manage corrupted data. An example of table extracted from **T2Dv2**, **SemTab_R3 2019**, **2T**, **HardTable_R2 2021** and **TURL** datasets are shown in Appendix A, in Tables C.7, C.8, C.9, C.10 and C.11.

The validation process starts with a set of mentions M, and a number k (which by default is 100), representing the number of candidates that will be extracted for each mention. The *Lookup* service returns a set of candidates E_m that includes all the entities matched. The returned set is then checked against the GS to verify which among the correct entities are present and in what position they are in the result. The metric used to quantify the goodness of LamAPI lookup is the **coverage** and it is computed following this formula:

$$coverage = \frac{\#\, correct\ candidates}{\#\, cell\ processed} \tag{1}$$

where # represents "number of".

Coverage is calculated for all cells in the dataset. The numerator indicates the number of correct candidates, and the denominator represents the number of cells for which a candidate must be generated.

It measures how well the entity retrieval system can capture a broad range of entities that match the query. High coverage implies that the system effectively retrieves many relevant entities, ensuring that the results are not limited to a narrow subset but encompass a broad spectrum of entities that may be of interest. On the other hand, low coverage suggests that the system might miss or overlook significant entities, leading to an incomplete representation of the available information.

Fig. 6 depicts how the coverage score changes with the number of candidates. The results reveal that for most datasets, the coverage remains consistently around 0.90 when the number of candidates ranges from 20 to 30. Nevertheless, there are exceptions, notably in the case of *SemTab_R3 2019* and *HardTable_R3 2021*, where the coverage reaches approximately 0.80 with a hundred candidates. The primary factor contributing to this score in *SemTab_R3 2019* is the prevalence of name abbreviations, making them challenging for LamAPI to be identified. In contrast, *HardTable 2021_R3* faces disambiguation challenges, as most cells consist of only one token, providing a little context for disambiguation and, for this reason, the default number k chosen for performing ER has been set to 100.

The other experiment concerns the average position of the correct candidate across different datasets (Fig. 7). *HardTable_R3 2021* obtains the worst results because one-token mentions have a wide range of possible candidates, so the correct candidates are often in the lower ranking positions. *SemTab_R3 2019* is critical for abbreviations of people's names, but when the correct candidate is found, it is detected

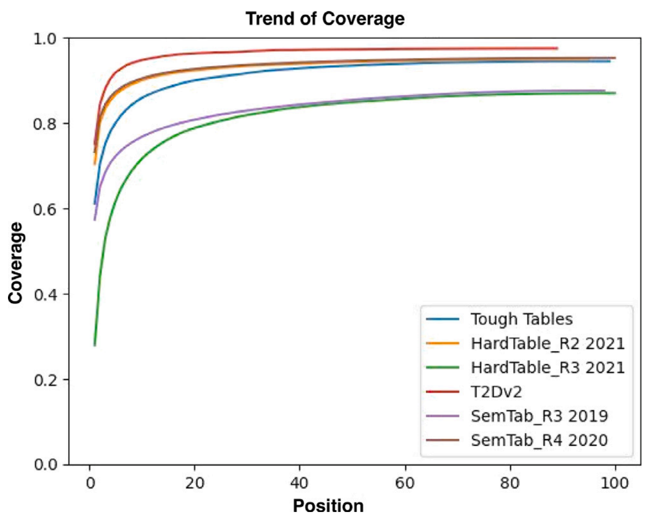


Fig. 6. Coverage score obtained by changing the number of candidates extracted from LamAPI.

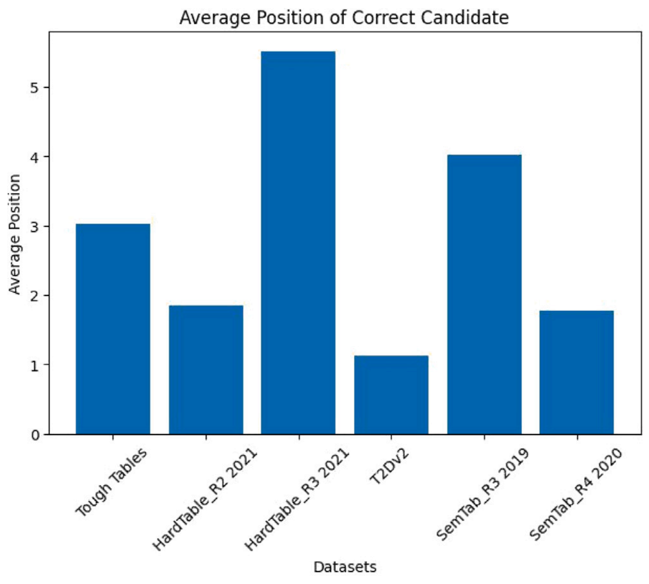


Fig. 7. Average position of correct candidates, for each dataset, retrieved from LamAPI.

around the 4th position. Also *Tough Tables* has a higher average position of the correct candidate (around 3) because this dataset is full of misspelt mentions, so retrieving the correct candidate in the first position is more complicated. However, it must be underlined that this dataset has been widely corrupted in order to insert a large number of typos into the mentions. It, therefore, does not represent a real scenario. *HardTable_R2 2021*, *T2Dv2*, and *SemTab_R4 2019* are more accessible since they have fewer typos and misspelt mentions.

Other approaches perform EL through customised indexes based on various KGs. The only service previously accessible for this purpose was MTab [55], which is no longer operational. We have previously computed the coverage score using the MTab system against the *2T* dataset. The system attained a coverage score of 0.93, while the current version of LamAPI achieves a higher coverage score of 0.97 against the *2T* dataset. The KG employed for the entity resolution task is Wikidata. This result shows that LamAPI competes well with other systems developed in the state of the art.

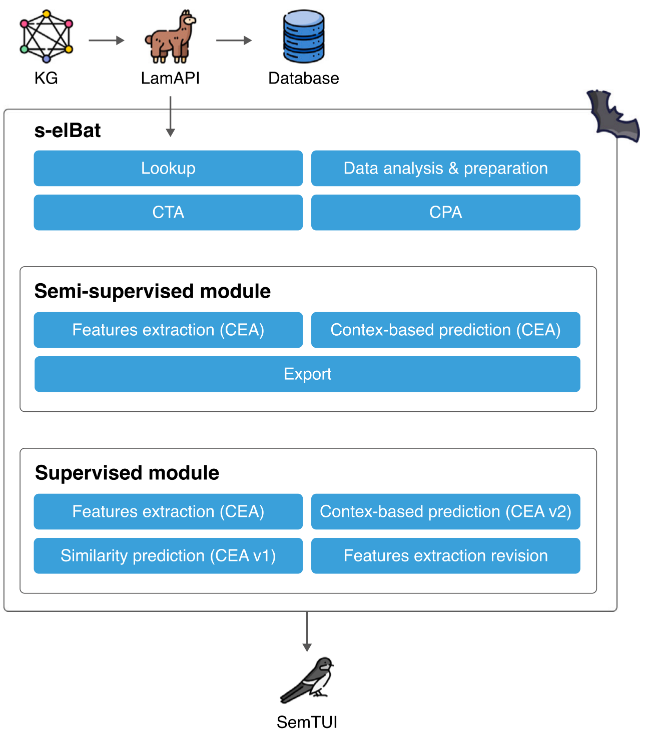

KG
LamAPI
Database
s-elBat
Lookup
Data analysis & preparation
CTA
CPA
Semi-supervised module
Features extraction (CEA)
Contex-based prediction (CEA)
Export
Supervised module
Features extraction (CEA)
Contex-based prediction (CEA v2)
Similarity prediction (CEA v1)
Features extraction revision
SemTUI

Fig. 8. Overall architecture of s-elBat system.

5. s-elBat

In this Section, we present s-elBat [44] system, which is an iterative process for performing ED 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. The overall architecture (depicted in Fig. 8) comprises two primary modules: the semi-supervised module and the supervised module, which are explained in Sections 5.1 and 5.2 respectively.

5.1. s-elBat Semi-supervised Module

The first version of s-elBat [44] (the semi-supervised module) was developed during the international challenge SemTab 2022[36] and it adopted an iterative heuristic-based algorithm. The pipeline of the process is represented in Fig. 9 and it is composed of 7 sequential phases: (i) **Preprocessing and Data Preparation**, (ii) **Entity Retrieval**, (iii) **CEA**, (iv) **CPA**, (v) **CTA**, (vi) **Revision**, and (vii) **Export**.

The (i) **Preprocessing and Data Preparation** phase aims to convert all cells in each table to lowercase, and any extra spaces and special characters, such as underscores (_), are removed to improve the results of the entity retrieval phase. Following this, a column classification process is carried out, labelling each column as either *L-column* (containing literals) or *NE-column* (containing named-entity mentions).

The column classification is executed by using LamAPI *Literal Recogniser* service, which detects the type of each mention in the column by using a set of RegexTypes (*i.e.*, boolean, date, email, geocoords, integer, float, ISBN, URL, XPath, CSS). In addition, s-elBat uses spaCy[37] to enhance the algorithm for assigning a data type and make it more resilient. When column cells are labelled as “string”, the algorithm automatically treats them as *NE-Columns*.

[36] sem-tab-challenge.github.io/2022
[37] spacy.io

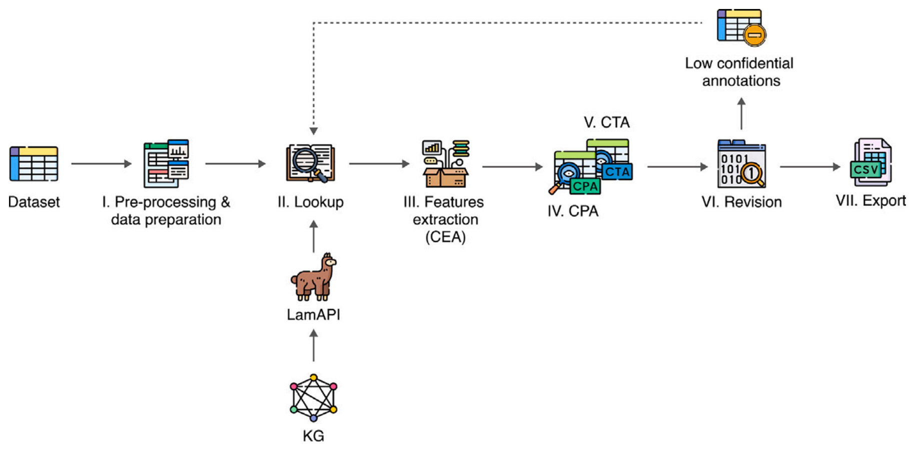
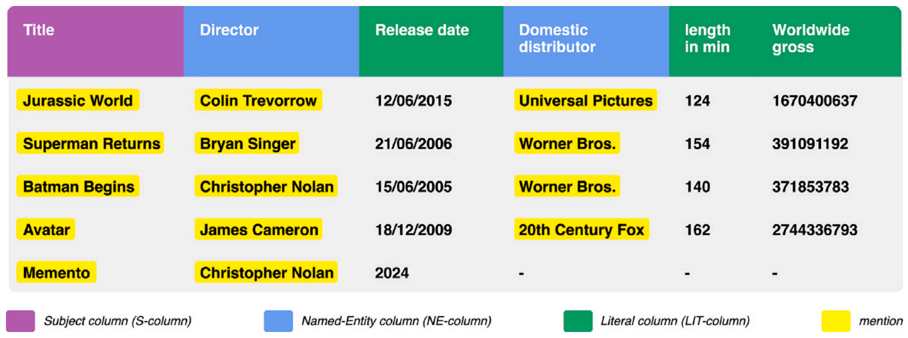

Low confidential annotations
V. CTA
Dataset
I. Pre-processing & data preparation
II. Lookup
III. Features extraction (CEA)
CPA
CTA
IV. CPA
VI. Revision
CSV
VII. Export
LamAPI
KG

Fig. 9. Process of semi-supervised module of s-elBat.



Title	Director	Release date	Domestic distributor	length in min	Worldwide gross
Jurassic World	Colin Trevorrow	12/06/2015	Universal Pictures	124	1670400637
Superman Returns	Bryan Singer	21/06/2006	Worner Bros.	154	391091192
Batman Begins	Christopher Nolan	15/06/2005	Worner Bros.	140	371853783
Avatar	James Cameron	18/12/2009	20th Century Fox	162	2744336793
Memento	Christopher Nolan	2024	-	-	-

🟪 *Subject column (S-column)* 🟦 *Named-Entity column (NE-column)* 🟩 *Literal column (LIT-column)* 🟨 *mention*

Fig. 10. A sample table (Fig. 2) with mentions highlighted in yellow.

For example, in Fig. 10, the columns *"Release date"*, *"Length in min"* and *"Worldwide gross"* are classified as *L-columns*, while *"Title"*, *"Director"* and *"Domestic Distributor"* as *NE-columns*.

For each mention $m \in M$, in the (ii) **Entity Retrieval** phase, the approach employs the LamAPI *Lookup* service to search and retrieve a set of entities E. During the service invocation, various heuristics are applied to handle potential misspellings. Specifically, two distinct requests are made: (i) using only the mention itself and (ii) modifying the mention by removing repeated letters and brackets. This adjustment is crucial as it enhances the performance of the 3-gram search implemented by ElasticSearch in LamAPI, which is sensitive to such mistakes. Simultaneously, fuzzy matching helps in efficiently handling potential double-character omissions. Additionally, brackets can affect the edit distance and often contain irrelevant content. The parameters utilised for the **Entity Retrieval** phase are the following: (i) **name**: "cell mention", (ii) **limit**: "100", (iii) **kg**: "wikidata", **fuzzy**: "true", **ngrams**: "False"

Once candidate entities are obtained from LamAPI, s-elBat estimates the (iii) **CEA** task by computing a feature vector for each candidate. Each feature vector represents the goodness with which the candidate can be linked to the cell. The entries of each vector are computed as follows:

- *Edit Distance*: the score is determined using the Levenshtein distance, which measures the difference between the mention and the label linked to the candidate entity;
- *Jaccard Distance*: the score is akin to string similarity, but it is computed using Jaccard distance rather than Levenshtein. Jaccard compares sets and is used for measuring the similarity between sets of items;
- *Object*: this score is assigned exclusively when a relationship exists between two candidate entities found in the same row for different columns. The algorithm iterates over candidates extracted for two cells (associated with *NE-columns*) in the same row, checking for the existence of a property that links the two candidates. The existence of a property is verified using LamAPI predicates service. If such property is found, the score is incremented;
- *Relation*: similar to the previous object score, the relation score is established when at least one property connects the entities under consideration. Also, in this case, the service used by LamAPI is the "predicates" one;
- *Literal*: this score is computed between each candidate retrieved for each cell by LamAPI and the cell in the *L-column*. This score is useful for assessing relationships between *NE-columns* and *L-columns*.

An example of feature vector computed for the three entities retrieved from LamAPI referred to the mention **"Jurassic World"**, in Fig. 10, is shown in Listing 8.

Listing 8: Feature vector computed for Jurassic World entity.

```
{
    "id": "Q3512046", (Jurassic World)
    "name": "jurassic world",
    "description": "2015 film directed by Colin Tr.",
    "edit_distance": 1, (Levenshtein distance)
    "jaccard_distance": 1, (Jaccard Distance)
    "object": 2.0, (Object relationship score)
    "relation": 0, (Property relation score)
    "literal": 1.061 (Literal matching score)
},
{
    "id": "Q55615459",
    "name": "jurassic world",
    "description": "a ride themed to the jurassic .."
    "edit_distance": 1,
    "jaccard_distance": 1,
    "object": 0,
    "relation": 0,
    "literal": 0
},
{
    "id": "Q18615494",
    "name": "jurassic world",
    "description": "1452 strip of the webcomic xkcd"
    "edit_distance": 1.0,
    "jaccard_distance": 1.0,
    "object": 0,
    "relation": 0,
    "literal": 0
},
```

Eventually, the final score is computed for each entity by using a weighted sum:

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

Where w_i represents the weight given to each feature. Initially, a manual qualitative analysis of a small set of annotations suggested by s-elBat was conducted to determine the set of weights. The set is then used to create five ranges with the attempt to consider cases not identified during the initial analysis: (i) *Edit Distance* = [1, 5, 10], (ii) *Jaccard Distance* = [2, 4, 8], (iii) *Object* = [1, 3, 7], (iv) *Relation* = [1, 2, 4], (v) *Literal* = [1, 4, 7] The ranges have been chosen in such a way as to have an upper and lower limit. Afterwards, these ranges were refined using a grid search algorithm on the HardTable_R2 2021 dataset, which was selected due to its broad and generic characteristics. Considering the maximum value of F1-score on the dataset, the final weights are: *Edit Distance* 10, *Jaccard Distance* 8, *Object* 3, *Relation* 4, *Literal* 7.

The next phase involves the (iv) **CPA** task, where the information gathered in the preceding phase is employed to group properties based on their frequency. This task is performed for every pair of columns. The initial phase involves counting, for each cell in the table, the properties that establish relationships at the row level with another column. Once the ranking for an individual cell is determined, the results are aggregated for the entire column, and the property with the highest score is chosen. The final result is a dictionary for each pair of columns, containing the winning properties and their respective frequencies. The most commonly occurring property is chosen for CPA annotation in the subsequent phase (the *Export* one).

Similarly, for the (v) **CTA** task, the approach builds a dictionary containing the frequencies of all the classes associated with the winning entities identified in the previous phase. The class with the highest frequency is chosen as the annotation for the analysed column.

The (vi) **Revision process** analyses all the information gathered in the previous phases to reorder the candidates in the final stage. In particular, this allows for correcting CEA entities previously selected: every entity has to be coherent with the CTA and CPA annotation.

Lastly, during the (vii) **Export** phase, the goal is to export the annotated mentions. The system must make decisions for each mention, determining whether the provided annotation is accurate or not based on confidence levels. An example of output provided by the Export phase is shown in Listing 9.

Listing 9: Example of output provided by Export phase of s-elBat (Unsup. module).

```
{
    "id": "Q3512046",
    "name": "jurassic world",
    "description": "2015 film directed by Colin T."
    "match": false,
    "score": 24.663
},
{
    "id": "Q21877685",
    "name": "jurassic world",
    "description": "2018 5th jurassic park film ..."
    "match": false,
    "score": 23.891
},
{
    "id": "Q2336369",
    "name": "jurassic world",
    "description": "american media franchise"
    "match": false,
    "score": 21.12
}
```

5.2. *s-elBat Supervised Module*

From the first version of s-elBat, new components have been added to improve the entity disambiguation by combining heuristic and ML-based approaches. The pipeline of the new algorithm is illustrated in Fig. 11, outlining the execution flow and key components. The pipeline of the process can be categorised into several phases: (i) **Data analysis and pre-processing**, (ii) **Lookup**, (iii) **Feature extraction**, (iv) **Similarity Prediction**, (v) **CTA**, (vi) **CPA**, (vii) **Feature extraction revision**, (viii) **Context-based Prediction**, and (ix) **Export**.

The creation of additional components in this updated version of s-elBat addresses the challenge of optimising feature weighting. The initial iteration of s-elBat involved manually assigning weights to each extracted feature by a domain expert (as seen in Section 5.1). These weights were determined through the experimentation (discussed in the following sections) until an optimal score was achieved.

However, the experimentation was limited to a subset of all possible values, and a complete evaluation would involve testing too many weight combinations, making the process time-consuming.

So, unlike the previous version, which employed a heuristic algorithm relying on predefined rules, this iteration seeks to harness the power of ML. While the heuristic approach depended on expert-derived strategies and fixed rules, making it less adaptable to dynamic environments and potentially hindering its performance with different data inputs, ML models offer the ability to make decisions based on learned patterns and relationships from data. They adapt and improve with more data, making them well-suited for dynamic and evolving scenarios and capable of excelling with new, previously unseen, data.

In the case of s-elBat, the adopted ML techniques allow to determine features weight by iterating on data and adapting them incrementally. Employing ML improves generalisation and streamlines model updates, making it a superior approach for achieving the most effective weight assignments.

(i) **Data analysis and pre-processing**, (ii) **Lookup**, (v) **CTA**, (vi) **CPA**, and (ix) **Export** phases, presented before, have been used without providing any changes.

The (vii) **Feature Extraction** phase builds upon the previous version but with some differences in how the feature vectors are constructed. The goal is to enhance the vector representation by including similarity and contextual information for each candidate. The following entries are added to the vector:

- **Number of tokens (ntoken)**: the number of tokens in the mention m. This feature serves as an indicator of the mention's ambiguity, as empirical evidence suggests that a greater number of tokens typically improves the accuracy of candidate disambiguation;
- **Popularity (popularity)**: the value represents the number of links each entity has in the KG. The score has been derived for each entity from Wikidata. The popularity of a node in a graph refers to the measure of its significance or importance within the network structure. It indicates the number of connections a particular node has within the graph;
- **Position score (pos_score)**: a positional score is calculated as follows: $1/i$, where i represents the position of the candidate entity e in the LamAPI ranking. This feature reflects the significance of the candidate entity's position within the IR system's ranking;

$$\begin{array}{l}\textbf{jurassic world}\\ \textit{(2015 film directed by Colin Trevorrow)}\end{array} \xrightarrow{pos\_score} 1$$

$$\begin{array}{l}\textbf{jurassic world}\\ \textit{(a ride themed to the jurassic world...)}\end{array} \xrightarrow{pos\_score} \frac{1}{2}$$

$$\begin{array}{l}\textbf{jurassic world}\\ \textit{(1452nd strip of the webcomic xkcd)}\end{array} \xrightarrow{pos\_score} \frac{1}{3}$$

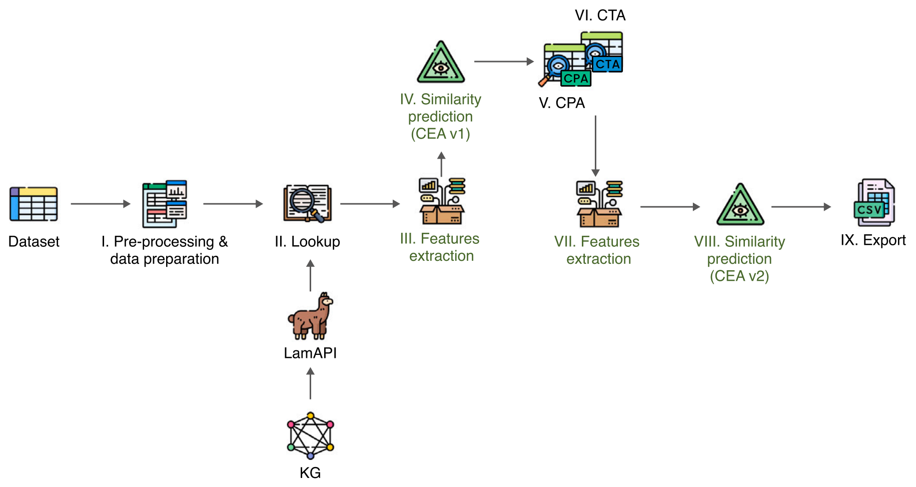
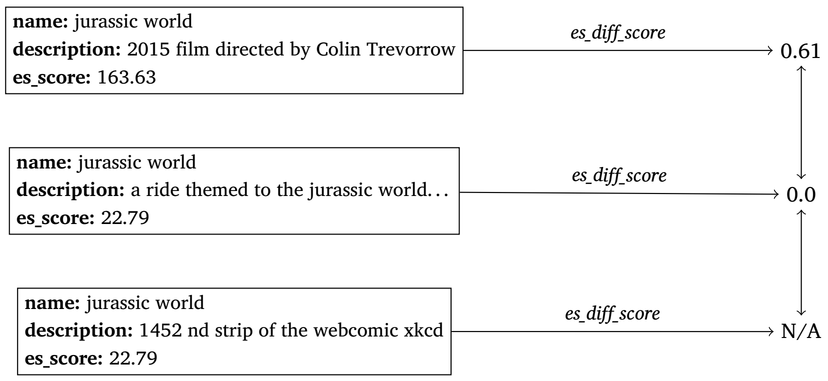

Dataset
I. Pre-processing & data preparation
II. Lookup
LamAPI
KG
III. Features extraction
IV. Similarity prediction (CEA v1)
V. CPA
VI. CTA
CPA
CTA
VII. Features extraction
VIII. Similarity prediction (CEA v2)
IX. Export
CSV

Fig. 11. Process of supervised module of s-elBat. The phases that differ from the semi-supervised version are highlighted in green.



- **Elasticsearch score (es_score)**: this score is internally computed by Elasticsearch and originates from LamAPI, making it a purely syntactic match. This feature represents the score generated by the IR system;
- **Elasticsearch difference score (es_diff_score)**: a score is calculated using the following formula: $(S_{e_i} - S_{e_{(i+1)}})/(S_{e_{(i+1)}})$, where S_{e_i} represents the Elasticsearch score of the ith candidate entity, and $S_{e_{(i+1)}}$ denotes the Elasticsearch score of the subsequent candidate entity in the LamAPI ranking. This feature quantifies the relevance gap between two candidates in the ranking produced by LamAPI system;
- **Jaccard ngram (jaccard)**: this metric quantifies the similarity between two strings by calculating the ratio of matching n-grams to the total count of unique n-grams. It assesses the resemblance between an entity mention and its corresponding name in a KG. Specifically, this feature evaluates the similarity between the n-grams present in the mention m and those found in the candidate entity's name, with a fixed value of n set to 3;
- **Cosine similarity score (cosine_similarity)**: this score is determined through the calculation of cosine similarity between the vectors representing the mention m and the entity's name c, both of which are generated using the FastText library[38];
- **Description score (desc)**: the score quantifies the likeness between the content of a table row and the description of a current candidate in a KG. It leverages Jaccard similarity to compare these two strings. This feature provides insight into the similarity between the tokens present in the row's text and the description of the current candidate, revealing the degree of likeness or shared information between the two text segments;
- **Description ngram score (descNgram)**: it quantifies the similarity between the content of a table row and the description of a current candidate in a KG. It utilises 3-grams (trigrams) and Jaccard similarity to compare the two strings. This feature offers detailed insights into the similarity between the trigrams in the row's text and the current candidate's description. It provides a more nuanced assessment of the degree of resemblance or shared information between the two texts compared to traditional token-level comparisons.

All the following features are used to build each vector associated with each candidate entity extracted by LamAPI. An example of a vector is represented in the following Listing 10. The example considers the mention "Jurassic World", and it shows the top three feature vectors associated with the top three candidate entities extracted by LamAPI.

Listing 10: Candidates: returned candidates for the mention Jurrasic World.

```
{
    "id": "Q3512046", (Jurassic World)
    "name": "jurassic world",
    "description": "2015 film directed by Colin Tr."
    "ed": 1, (Levenshtein distance)
    "jaccard": 1, (Jaccard distance)
    "object": 2.0, (Object relationship score)
    "literal": 1.061, (Literal matching score)
    "relation": 0, (Property relation score)
    "ntoken": 2, (Number of tokens)
    "popularity": 61, (Popularity)
    "pos_score": 1, (Position score)
    "es_score": 163.73, (Elasticsearch score)
    "es_diff_score": 0.060022, (Elastic difference)
    "jaccardNgram": 1, (Jaccard ngram)
    "cosine_similarity": 1, (Cosine similarity score)
    "desc": 0.333, (Description score)
    "descNgram": 0.261 (Description ngram score)
```

[38] fasttext.cc

```
},
{
    "id": "Q55615459",
    "name": "jurassic world",
    "description": "a ride themed to the jurassic .."
    "ed": 1,
    "jaccard": 1,
    "object": 0,
    "literal": 0,
    "relation": 0,
    "ntoken": 2,
    "popularity": 4,
    "pos_score": 0.5,
    "es_score": 22.79,
    "es_diff_score": 0.0,
    "ed": 1.0,
    "jaccard": 1.0,
    "jaccardNgram": 1.0,
    "cosine_similarity": 1.0,
    "p_subj_ne": 0,
    "p_subj_lit": 0,
    "p_obj_ne": 0,
    "desc": 0.25,
    "descNgram": 0.348
},
{
    "id": "Q18615494",
    "name": "jurassic world",
    "description": "1452 strip of the webcomic xkcd"
    "ed": 1,
    "jaccard": 1,
    "object": 0,
    "literal": 0,
    "relation": 0,
    "ntoken": 2,
    "popularity": 1,
    "pos_score": 0.33,
    "es_score": 22.79,
    "es_diff_score": 0.113583,
    "ed": 1.0,
    "jaccard": 1.0,
    "jaccardNgram": 1.0,
    "cosine_similarity": 1.0,
    "p_subj_ne": 0,
    "p_subj_lit": 0,
    "p_obj_ne": 0,
    "desc": 0.0,
    "descNgram": 0.0,
}
```

The success of the ML model used in subsequent phases relies heavily on the quality and relevance of the features contained in the vectors. The chosen features must capture essential and distinctive information from the table data; for this reason, similarities and contextual data are included in each vector. It is important to note that the s-elBat algorithm is an ED technique, so it is focused on disambiguation. Therefore, if the LamAPI algorithm fails to identify the correct entity during the ER phase, s-elBat will produce incorrect annotations.

The (iv) **Similarity Prediction** is the first ED phase computed to have a re-ranking result of entities extracted by ER module. LamAPI already provides a ranking for each named entity cell, but its main issue is that it only considers text similarity scores. Features vectors, instead, incorporate additional information related to the contextual knowledge found in the table. Specifically, they encompass scores related to "object", "relation" and "literal", which explore connections between candidates extracted for cells within the same row. At this point in the algorithm, the problem is to assess the scores obtained within the vector and determine how to assign different importance to the obtained features. In s-elBat semi-supervised module, this task has been done empirically by trying different configurations and eventually adopting the best one. Furthermore, s-elBat supervised incorporates an extensive array of features, making the empirical determination of weights impractical. For this reason s-elBat supervised module adopts a ML model to compute the reranking of the candidate entities. The final ML model has been accurately selected by performing a set of tests using four ML algorithms. The selected models for the tests are: **Support Vector Machine** [89], **Random Forest** [90], **Logistic Regression** [91], and **Neural Network** [92]. These models have been implemented using *scikit-learn*[39] library with the default parameters.

In comparison to other models, the NN requires more thorough experimentation to identify its optimal structure. For this reason, before proceeding with the algorithm description, we provide details of the chosen network architecture.

[39] scikit-learn.org/stable/

Table 2
Model Architecture of the final Neural Network.

Layer (type)	Output shape	Param #	Connected to
dense	(64,)	1344	(20,)
batch_norm	(64,)	256	(64,)
dense_1	(128,)	8320	(64,)
batch_norm_1	(128,)	512	(128,)
dense_2	(256,)	33 024	(128,)
batch_norm_2	(256,)	1024	(256,)
dense_3	(128,)	32 896	(256,)
batch_norm_3	(128,)	512	(128,)
dense_4	(64,)	8256	(128,)
batch_norm_4	(64,)	256	(64,)
dense_5	(2,)	130	(64,)

NN architecture. The NN architecture adopted in s-elBat supervised module (Table 2 and Fig. 12) was derived through multiple experiments. This feed-forward architecture has been built by considering that a model with insufficient parameters would struggle to learn effectively, while an excessive number of parameters would lead to overfitting. The objective was to strike a balance and create an architecture that mitigates both issues.

The deep feed forward NN is structured as follows:

- **Input Layer**: The input layer serves as the NN's starting point, receiving raw input data and acting as the gateway for information processing. In s-elBat context, the raw input data is represented as feature vectors, each with a size of 14.
- **Hidden Layers**: Hidden layers allow NNs to learn complex patterns and representations in the data. There are 5 hidden layers, respectively, with 64, 128, 256, 128, and 64 neurons, which are able to learn and extract features from input data. For each neuron, the activation function is **ReLU (Rectified Linear Unit)** [93], which is used to introduce non-linearity in the model.
- **Output Layer**: The output layer, composed of 2 neurons, represents the final prediction and returns the probability of a candidate to be positive (the entity represents the correct annotation) and negative (the entity is not the correct annotation).

This architecture enables the network to learn hierarchical representations of the data. Lower layers capture simple features, while higher layers build more abstract and complex representations. This hierarchical learning is effective for tasks that require understanding data at different levels of abstraction. Similarly, [38] leverages a 2-layer NN model for the task of CEA. The approach employs the NN model to learn weights and relationships from labelled data.

The NN has been built in this way after some preliminary tests with different configurations. Previous networks, with few layers, did not fit well input data (there was an underfitting problem), so other layers have been added to increment the computational complexity of the network. An evaluation of the model described above will be proposed in Section 4.3. As described in the validation (Section 4.3), the NN was trained using all datasets performing a k-fold cross-validation, where k = 6, which is the number of datasets.

Eventually, the model chosen to perform this phase is NN, which is well-suited for reranking feature vectors in various applications because it captures complex patterns learnt from data. For this task, s-elBat computes reranking using a deep NN architecture as this model has shown superior capability in modelling intricate data relationships. It can automatically learn relevant features from raw data, thereby reducing the need for manual feature engineering, a challenge faced in the semi-supervised module of s-elBat.

The score computed from the NN model creates a new ranking for each candidate extracted for each cell. This new ranking is based on contextual information that LamAPI cannot observe since it acts only as an information retrieval system. An example of the output of the ranking generated by the NN is shown in Listing 11.

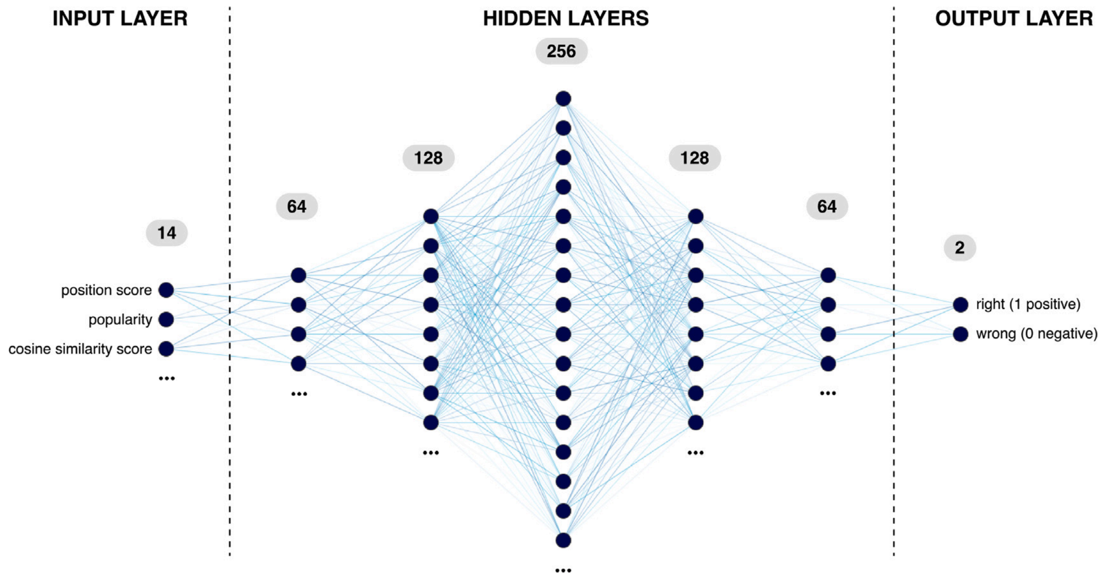


Fig. 12. Neural Network architecture.

Listing 11: Similarity Prediction of s-elBat (Sup. module).

```
{
    "id": "Q3512046",
    "name": "jurassic world",
    "description": "2015 film directed by Colin T."
    "match": false,
    "score": 0.992
},
{
    "id": "Q55615459",
    "name": "jurassic world",
    "description": "a ride themed to the jurassic .."
    "match": false,
    "score": 0.977
},
{
    "id": "Q18615494",
    "name": "jurassic world",
    "description": "1452 strip of the webcomic xkcd"
    "match": false,
    "score": 0.128
}
```

The (vii) **Feature extraction revision** aims to extract further information related to CPA and CTA tasks. The previous phase already returns a ranked list of candidate entities for each cell that can be already used to select the winning entity. Features vectors computed during the previous phase contain information related to the similarity and the context around the investigated cell, but they do not consider information related to CTA and CPA phases. Therefore, incorporating details at both column and row levels can improve the ultimate ranking of candidates for each cell. In this phase, additional features are computed to consider also types and properties information at column and row level. The following features are added to the previous feature vector of the first three entities:

- **Column score (cta)**: The score is determined by the types gathered during the CEA phase. Each candidate entity possesses a distinct set of types extracted through the KG ontology. Subsequently, a new score is determined by aggregating the scores associated with these types. To accomplish this, the dictionary generated in the CTA phase is applied. Upon completing the summation of all candidate types, the score is normalised based on the number of types present in the set. This feature offers insights into how closely the types linked to the current candidate entity align with the frequencies of types collected in the column, indicating the level of consistency or agreement between them;
- **Column score max (ctaMax)**: The score is determined based on the categories compiled during the CEA phase. The same process used for computing the "cta" score, as outlined previously, is employed here. However, in this instance, the maximum frequency within the type set is taken into account. This feature measures the alignment of types linked to the current candidate entity by identifying the type with the highest frequency among those collected during the CTA phase in the column;
- **Property score (cpa)**: The score relies on the properties collected during the CEA phase. Similarly to the "cta" score, but computed on properties. For each gathered property, it calculates the cumulative frequency exclusively for properties relevant to the candidate. This feature assesses the coherence between the properties linked to the current candidate entity and those gathered in the column during the CEA phase;
- **Property score max (cpaMax)**: Starting from the previous score ("cpa"), for each collected property, it extracts only the max property that belongs to the candidate. This feature captures the consistency of the properties associated with the current candidate entity by considering the property that has the maximum occurrences among the properties collected during the CEA phase in the column;
- **Pre-linking score (ceaP)**: Stands for "cea pre-linking score", which is a score assigned by a model before the revision phase, without taking into account consistency with types and properties. This feature captures the score value assigned to the candidate entity during the initial prediction, which serves as input for the subsequent prediction phases. It allows for the incorporation of the score from the previous prediction when performing the second prediction, considering its impact on the final outcome;
- **Pre-linking difference score (diff)**: Refers to a score based on the difference between the ceaP score and the score of the first candidate, which is the candidate with the highest ceaP score. This feature captures a measure of the disparity or difference in scores between the initial prediction and the top candidate. A lower "diff" score indicates a smaller difference between the "ceaP" score and the score of the first candidate, suggesting lower ambiguity in the first prediction. Conversely, a higher "diff" score indicates a greater disparity between the scores, suggesting higher ambiguity in the first prediction.

An example of CTA and CPA features computed for the first three "Jurassic World" candidate entities are reported in Listing 12.

Listing 12: CTA and CPA scores for Jurassic World candidates.

```
{
    "id": "Q3512046",
    "name": "jurassic world",
    "description": "2015 film directed by Colin T."
    "cpa": 0.46, (Property score)
    "cpaMax": 1.0, (Property score max)
    "cta": 0.5, (Column score)
    "ctaMax": 0.75, (Column score max)
    "ceaP": 0.973, (Pre-linking score)
    "diff": 0 (Pre-linking difference score)
},
{
    "id": "Q55615459",
    "name": "jurassic world",
    "description": "a ride themed to the jurassic .."
    "cpa": 0,
    "cpaMax": 0,
    "cta": 0,
    "ctaMax": 0,
    "cea": 0.93,
    "diff": 0.043
},
{
    "id": "Q18615494",
    "name": "jurassic world",
    "description": "1452 strip of the webcomic xkcd"
    "cpa": 0,
    "cpaMax": 0,
    "cta": 0,
    "ctaMax": 0,
    "cea": 0.391,
    "diff": 0.582
}
```

The (viii) **Context-based Prediction**, which is the last phase of the s-elBat supervised module, aims to produce the final ranking. The phase has been implemented by using a second NN that takes as input the feature vector obtained in the previous phase and produces a score associated to each candidate. Also in this case, the model used is a NN with the same configuration of the previous adopted in the first prediction phase (Table 2 and Fig. 12).

Now, the final ranking is defined and the best candidate is selected as the final annotation to produce for each cell.

In the following listing (Listing 13) the final result given as output by s-elBat is reported.

Listing 13: Final prediction of s-elBat sup. module.

```
{
    "id": "Q3512046",
    "name": "jurassic world",
    "description": "2015 film directed by Colin T."
    "match": false,
    "score": 0.992
},
{
    "id": "Q55615459",
    "name": "jurassic world",
    "description": "a ride themed to the jurassic .."
    "match": false,
    "score": 0.875
},
{
    "id": "Q18615494",
    "name": "jurassic world",
    "description": "1452 strip of the webcomic xkcd"
    "match": false,
    "score": 0.105
}
```

The output demonstrates the ability of the NN to predict a score for each entity. In this instance, s-elBat accurately identifies the correct entity for the mention “Jurassic World”, corresponding to the film directed by “Colin Trevorrow”.

Finally, we also computed the total execution time of LamAPI and s-elBat supervised module in order to give an idea of how long each dataset takes to be processed. The execution times are reported in Table 3 and in Fig. 13.

6. Validation

This Section discusses the validation, indicating the datasets used and their characteristics. The Section is divided into two parts: the first part explains the results of the experiments carried out to choose s-elBat supervised model, and the second part validates the model against other algorithms in the state of the art.

Table 3
Execution time table of LamAPI and s-elBat supervised module on every dataset.

Datasets	Time (s)
T2Dv2	1300.53
HT_R2 2021	2993.83
HT_R3 2021	8080.27
Tough Tables	23 563.15
SemTab R3 2019	24 450.13
SemTab R4 2020	49 269.8

The datasets used are extracted from the SemTab challenges [94]. Specifically, the *SemTab 2019 R1(T2Dv2)-R3-R4*, *SemTab 2020 R4 (Tought Tables)*, *SemTab 2021 R2-R3*, *SemTab 2022 R1-R2*, and *SemTab 2023 R1* datasets were utilised. In addition, the dataset used for validation by the TURL [71], and TableLlama [74] approaches have been included.

6.1. Experiments on ML models for s-elBat supervised

The SemTab datasets have been used to select the best ML models for s-elBat supervised module. For each ML model (*i.e.*, SVM, Random Forest (RF), Logistic Regression (LR) and NN), six training and validation datasets were created, one for each of the six selected datasets. The idea was to perform a kind of k-fold cross-validation, with $k = 6$, to test the score achieved on each of the six datasets. Each training was conducted from scratch to avoid influencing the model during the validation phase and to obtain the cleanest possible score. A single training is performed on five out of the six selected datasets, with testing done on the remaining one. The combination of five datasets provides a large amount of data for training our ML models.

In the context of a table, every cell in a *NE-column* is treated as an individual training example, and the process involves selecting a single candidate for annotating the corresponding mention in the cell. This setup resembles a binary classification task, wherein the goal is to designate one candidate as positive (labelled as 1), while all other candidates generated during the candidate generation phase are considered negatives (labelled as 0). During the training, having both positive and negative training samples is crucial for training models, especially in binary classification tasks. A training dataset has been built in this scenario, where a batch of 11 training samples is generated for each cell. One of these samples corresponds to the correct candidate and is labelled as positive, while the remaining ten examples are labelled as negative. The positive sample is derived from the GS of each dataset, while the set of negatives is obtained from candidate generation by selecting 10 incorrect results. The choice to extract 10 negative candidates from LamAPI (plus the correct candidate) derives from the study on the position of the correct candidate for each dataset. As shown in Fig. 7, in the worst case (*SemTab 2021 R3*), the correct candidate is, on average, in position 6. For this reason, 11 appears to be an optimal choice for the number of candidates in the batch.

In Table 4, we compare Logistic Regression, NN and Random Forest. SVM results have not been included, as the model does not converge.

As highlighted, the NN model achieved the best results, leading to the decision to propose a NN as the final model.

6.2. S-elBat with revision and without revision

As explained in the previous chapter, s-elBat supervised module introduced two prediction phases to perform the CEA task. As the similarity prediction currently overlooks information pertaining to CTA and CPA tasks, context-based prediction has been introduced to incorporate these additional features for computing a secondary classification. To assess whether these pieces of information regarding the CTA and CPA tasks are indeed useful for annotating the cells, an experiment was

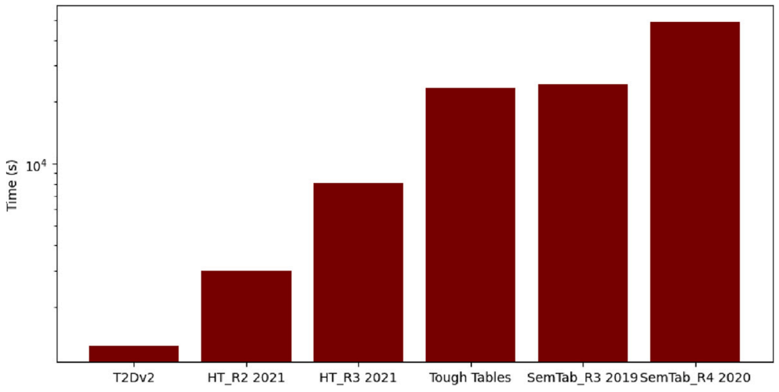


Fig. 13. Execution time of LamAPI and s-elBat supervised module on every dataset.

Table 4
Comparison between Logistic Regression, Neural Network and Random Forest.

Dataset	Logistic regression			Neural network			Random forest		
	P	R	F1	P	R	F1	P	R	F1
T2Dv2	0.89	0.89	0.89	0.89	0.89	0.89	0.87	0.87	0.87
Tough Tables	0.89	0.89	0.89	0.93	0.93	**0.93**	0.89	0.89	0.89
SemTab_R3 2019	0.79	0.79	0.79	0.81	0.81	**0.81**	0.77	0.77	0.77
SemTab_R4 2019	0.95	0.94	0.94	0.95	0.94	**0.95**	0.92	0.92	0.92
HardTable_R2 2021	0.95	0.95	0.94	0.97	0.96	**0.97**	0.89	0.89	0.89
HardTable_R3 2021	0.87	0.87	0.87	0.97	0.96	**0.97**	0.85	0.85	0.85

Table 5
Comparison between s-elBat and state of the art approaches.

			SemTab 2019			SemTab 2020	SemTab 2021		SemTab 2022		SemTab 2023	TURL
			R1 (T2Dv2)	R3	R4	R4 (Tough Table)	R2	R3	R1	R2	R1	dataset
s-elBat	semi-supervised		0.87	0.77	0.94	0.96	0.93	0.92	–	–	–	–
	sup	wo. revision	0.85	0.79	0.83	0.92	0.94	0.91	–	–	–	–
		w. revision	0.90	0.81	**0.98**	**0.97**	**0.99**	0.93	0.87	0.69	**0.85**	0.91
Mtab			**1.0** [95]	**0.97** [95]	0.98 [95]	0.90 [96]	**0.98** [97]	**0.97** [97]	–	–	–	–
DAGOBAH			0.89 [98]	0.72 [98]	0.58 [98]	0.83 [99]	0.97 [100]	0.94 [100]	**0.95** [101]	**0.90** [101]	–	–
KGCODE-Tab [102]			–	–	–	–	–	–	0.89	0.85	–	–
TorchicTab [75]			–	–	–	–	–	–	–	–	0.83	–
TURL [71]			–	–	–	–	–	–	0.41	0.29	0.34	0.84
TableLlama [74]			–	–	–	–	–	–	0.75	0.64	0.71	0.93

conducted to verify if the final score increases when incorporating this second prediction, which we refer to as "Context-based prediction".

The analysis of the data presented in Table 5 (last two rows) highlights a significant impact of using the revision, *i.e.*, the inclusion of types and properties, on the performance of the NN. On average, an improvement of 2%/3% in performance is observed. Furthermore, by setting a confidence threshold of 0.8, it is possible to achieve a precision exceeding 90%, indicating that errors and more ambiguous cases can be identified with high reliability.

6.3. *Comparison with state of the art approaches*

It is also crucial to understand how s-elBat performs compared to other state of the art solutions; the comparison results are shown in Table 5.

Concerning the heuristic-based approaches, the winners of the various editions of SemTab have been selected: MTab (winner in 2019 [95] and in 2020 [96] with MTab4Wikidata), DAGOBAH (winner in 2021 [100] and 2022 [101]), KGCODE-Tab [103] (winner 2022), and TorchicTab [75] (winner 2023). s-elBat has also been compared with other state of the art LLM-based algorithms such as TURL [71] and TableLlama [74]; however, these approaches were developed outside the context of the challenge.

Overall, s-elBat's scores closely resemble the top scores in the SemTab benchmark. However, in *SemTab_R3 2019*, s-elBat registers a slightly lower performance, attributed to the dataset's distinctive feature of containing numerous abbreviations of people's names.

Since s-elBat reports better results using the supervised module with revision, for the datasets of the last two SemTab editions, we reported only the scores of the supervised module with revision. The evaluation of the MTab approach has not been reported because the system is not yet available. Furthermore, the tool did not participate in the last two SemTab challenge editions, so no score is available for these datasets. On the other hand, DAGOBAH achieves excellent results in the CEA task, at the expense of execution time, which is pretty high (*e.g.*, for a table with 6 rows and 4 columns, using 20 candidates, it takes 2094 seconds, so respectively 34 minutes).

Each approach has unique characteristics when it comes to implementation. As a result, other approaches may have employed different training datasets. However, the results are related to the same dataset used for the test set by all approaches reported.

Regarding TURL [71] and TableLlama [74], neither of these systems has been tested against the SemTab datasets. For this reason, we have evaluated them against the latest challenge datasets: *SemTab_R1 2022*, *SemTab_R2 2022*, and *SemTab_WikidataTables 2023*. These datasets were converted into a suitable format for TURL and TableLlama, using 50 candidates retrieved from LamAPI to test them on the CEA task. Table 5 highlights how s-elBat achieves excellent results even compared to LLM-based models. TableLlama, with its 7 billion parameters, achieves very good results on SemTab challenge datasets. However, the

Fig. 14. Splash screen of the tool.

latest model faces more challenges with disambiguation, possibly due to the increased number of candidates at the cell. Furthermore, TURL and TableLlama use other information such as *page title*, *section title*, and *caption* if they are available, while s-elBat does not need these information to perform the annotations.

Ultimately, s-elBat obtains excellent results in terms of F-measures on multiple datasets, even with very different characteristics. Note, instead, how other approaches must use ad-hoc techniques for different datasets to obtain good values. Our proposal performs similarly to the state of the art LLM-based approaches but requires substantially fewer computational resources. In fact, fine-tuning an LLM for a specific task requires great computational power and specific hardware and is an activity characterised by high costs. Consider that TableLlama [74] required a cluster of 48 NVIDIA A100 Tensor Core GPUs for finetuning. Furthermore, the presence of the features allows greater control of the technique and explainability (through the analysis of feature values).

7. SemTUI: Exploration of annotated tables

When trying to understand the errors made by the system, the development of STI algorithms is often challenging to assess. One way to facilitate this process is to have a *UI* available for result visualisation (Fig. 14). Using a web application for tabular annotation is fundamental to understand in which cases the algorithm fails to annotate a cell or to comprehend the scores assigned to various candidates extracted for each cell. s-elBat involves SemTUI [104], which is an open-source tool that provides ways to both annotate a raw table and easily view data of semantically annotated tables, developed using React and Typescript (Fig. 15, Fig. 16).

Through the interface, the user makes a final decision about annotations (*i.e.*, choose the correct annotations among the candidates if the confidence is low) or definitively classifies a mention as NIL (Fig. 17). For the unlinked mention, SemTUI shows an additional screen (Fig. 18).

8. Conclusions and future works

This paper discussed how our STI process, supported by s-elBat and LamAPI, can be used to efficiently and effectively annotate a set of tables with links that refer to a reference KG.

In Section 2, we have presented the key challenges, outlining the identified issues across all STI tasks. In this paper, most of these challenges have been addressed:

(i) ***Heterogeneity of domains and data distributions***: leveraging diverse, versatile datasets for training ML models within the supervised module allows for fine-tuning the s-elBat deep learning model, tailoring it to task-specific solutions.

(ii) ***Matching tabular values against the KG***: to address this concern, LamAPI employs indexing for entity mentions, synonyms, aliases, abbreviations, and acronyms. Furthermore, various matching strategies are utilised to mitigate issues arising from typos and misspelt mentions.

(iii) ***Disambiguation of named entities***: disambiguation is a crucial challenge, particularly in situations where limited contextual information is available. s-elBat tool addresses this challenge by incorporating table context at both the column and row levels, ensuring a comprehensive assessment before selecting the most suitable entity for annotation.

(iv) ***NIL-mentions***: the existing method overlooks *NIL-mentions* as it consistently aims to choose an annotation for every cell within a *NE-column*. In future works, we will focus on the development of a method that utilises techniques and external sources (*e.g.*, search engines) for enriched representations of mentions and entities in order to identify *NIL-mentions*. We will also incorporate domain-specific expert knowledge to enhance this identification.

(v) ***Choosing the most appropriate types and properties***: s-elBat tackles the selection of types and properties by analysing the contextual information within table cells. Consequently, establishing relationships between candidate entities at the row level enhances the likelihood of identifying the correct property linking two distinct columns. Furthermore, enriching types involves examining common types extracted from the lookup task at the column level, ensuring the identification of the accurate set of types characterising each column.

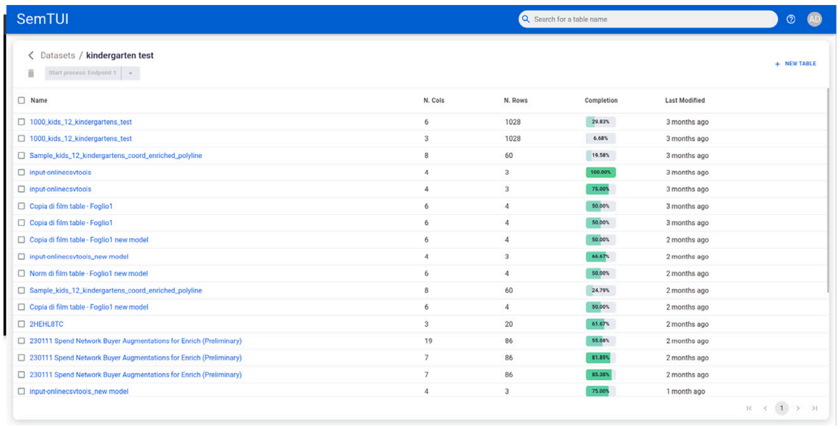
In addition, other transversal challenges were addressed:

(vi) ***Limited contextual information***: contextual information is crucial for executing STI tasks, with table headers and captions playing a significant role in achieving optimal performance. In cases where such information is absent, inferring missing context from the table structure is feasible through a combination of heuristic and ML techniques. s-elBat endeavours to identify contextual information from the table structure, even when limited contextual cues are available. The supervised model within s-elBat excels in capturing such information more effectively by employing deep learning models.

(vii) ***Detecting the type of columns***: the challenge of identifying literals within a tabular structure has been successfully addressed through the strategic application of Regex rules and services, such as spaCy,[40] for recognising string datatypes.

(viii) ***Collective aggregation of evidence from different tasks***: s-elBat leverages outputs from various tasks to carry out a "revision", thereby improving the likelihood of achieving optimal performance by incorporating contextual information. Both modules within the s-elBat system (Semi-supervised and Supervised), efficiently exchange information across sub-tasks. The key distinction lies in the heuristic

[40] spacy.io



SemTUI

Search for a table name

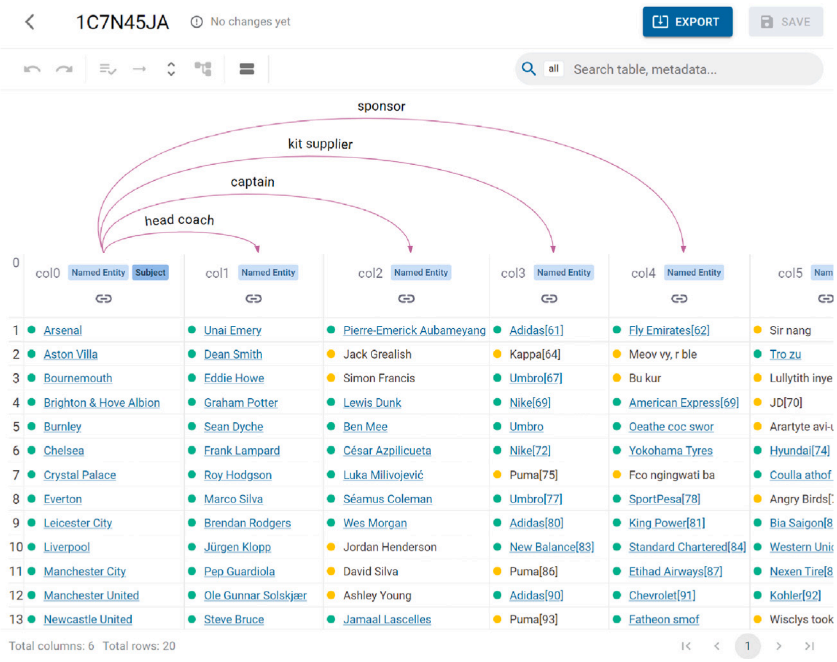
Datasets / kindergarten test

Start process: Endpoint 1

+ NEW TABLE

Name	N. Cols	N. Rows	Completion	Last Modified
1000_kids_12_kindergartens_test	6	1028	29.83%	3 months ago
1000_kids_12_kindergartens_test	3	1028	6.68%	3 months ago
Sample_kids_12_kindergartens_coord_enriched_polyline	8	60	19.58%	3 months ago
input-onlinecsvtools	4	3	100.00%	3 months ago
input-onlinecsvtools	4	3	75.00%	3 months ago
Copia di film table - Foglio1	6	4	50.00%	3 months ago
Copia di film table - Foglio1	6	4	50.00%	3 months ago
Copia di film table - Foglio1 new model	6	4	50.00%	2 months ago
input-onlinecsvtools_new model	4	3	66.67%	2 months ago
Norm di film table - Foglio1 new model	6	4	50.00%	2 months ago
Sample_kids_12_kindergartens_coord_enriched_polyline	8	60	24.79%	2 months ago
Copia di film table - Foglio1 new model	6	4	50.00%	2 months ago
2HEHL8TC	3	20	61.67%	2 months ago
230111 Spend Network Buyer Augmentations for Enrich (Preliminary)	19	86	55.08%	2 months ago
230111 Spend Network Buyer Augmentations for Enrich (Preliminary)	7	86	81.89%	2 months ago
230111 Spend Network Buyer Augmentations for Enrich (Preliminary)	7	86	85.38%	2 months ago
input-onlinecsvtools_new model	4	3	75.00%	1 month ago

Fig. 15. Display list of the tables within a dataset. For each table it is possible to see some statistics like the number of columns and rows.



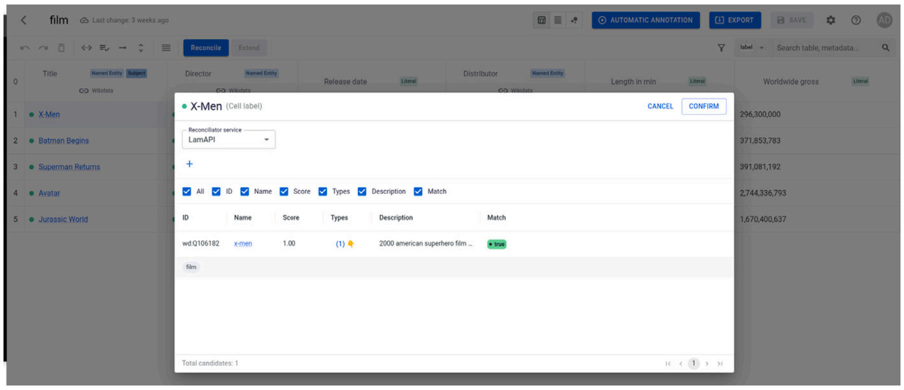
1C7N45JA No changes yet EXPORT SAVE

all Search table, metadata...

0	col0 Named Entity Subject	col1 Named Entity	col2 Named Entity	col3 Named Entity	col4 Named Entity	col5 Nam
1	Arsenal	Unai Emery	Pierre-Emerick Aubameyang	Adidas[61]	Fly Emirates[62]	Sir nang
2	Aston Villa	Dean Smith	Jack Grealish	Kappa[64]	Meov vy, r ble	Tro zu
3	Bournemouth	Eddie Howe	Simon Francis	Umbro[67]	Bu kur	Lullytith inye
4	Brighton & Hove Albion	Graham Potter	Lewis Dunk	Nike[69]	American Express[69]	JD[70]
5	Burnley	Sean Dyche	Ben Mee	Umbro	Oeathe coc swor	Arartyte avi-ı
6	Chelsea	Frank Lampard	César Azpilicueta	Nike[72]	Yokohama Tyres	Hyundai[74]
7	Crystal Palace	Roy Hodgson	Luka Milivojević	Puma[75]	Fco ngingwati ba	Coulla athof
8	Everton	Marco Silva	Séamus Coleman	Umbro[77]	SportPesa[78]	Angry Birds[
9	Leicester City	Brendan Rodgers	Wes Morgan	Adidas[80]	King Power[81]	Bia Saigon[8
10	Liverpool	Jürgen Klopp	Jordan Henderson	New Balance[83]	Standard Chartered[84]	Western Unic
11	Manchester City	Pep Guardiola	David Silva	Puma[86]	Etihad Airways[87]	Nexen Tire[8
12	Manchester United	Ole Gunnar Solskjær	Ashley Young	Adidas[90]	Chevrolet[91]	Kohler[92]
13	Newcastle United	Steve Bruce	Jamaal Lascelles	Puma[93]	Fatheon smof	Wisclys took

Total columns: 6 Total rows: 20

Fig. 16. Detail page of a table, with the annotation display. For each cell it is possible to see the associated entity, or the list of candidates.



film Last change: 3 weeks ago AUTOMATIC ANNOTATION EXPORT SAVE

Reconcile Extend label Search table, metadata...

0	Title Named Entity Subject	Director Named Entity	Release date Literal	Distributor Named Entity	Length in min Literal	Worldwide gross Literal
1	X-Men					296,300,000
2	Batman Begins					371,853,783
3	Superman Returns					391,081,192
4	Avatar					2,744,336,793
5	Jurassic World					1,670,400,637

X-Men (Cell label) CANCEL CONFIRM

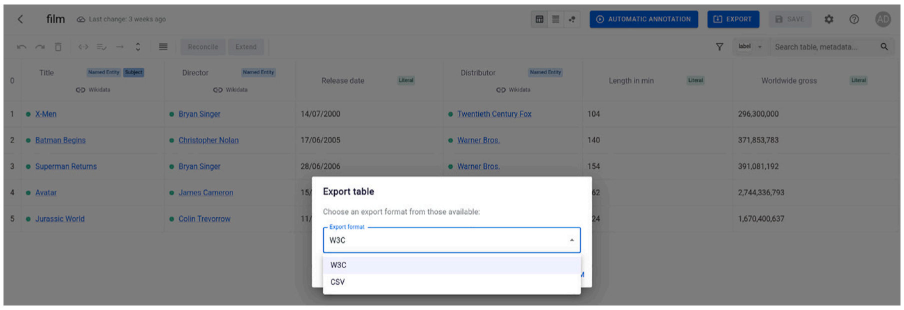
Reconciliator service: LamAPI

☑ All ☑ ID ☑ Name ☑ Score ☑ Types ☑ Description ☑ Match

ID	Name	Score	Types	Description	Match
wd:Q106182	x-men	1.00	(1)	2000 american superhero film ...	true
film					

Total candidates: 1

Fig. 17. Detailed page for the analysis result of reconciling a mention.



0	Title Named Entity Subject Wikidata	Director Named Entity Wikidata	Release date Literal	Distributor Named Entity Wikidata	Length in min Literal	Worldwide gross Literal
1	X-Men	Bryan Singer	14/07/2000	Twentieth Century Fox	104	296,300,000
2	Batman Begins	Christopher Nolan	17/06/2005	Warner Bros.	140	371,853,783
3	Superman Returns	Bryan Singer	28/06/2006	Warner Bros.	154	391,081,192
4	Avatar	James Cameron	15/		62	2,744,336,793
5	Jurassic World	Colin Trevorrow	11/		24	1,670,400,637

Export table

Choose an export format from those available:

Export format: W3C

W3C
CSV

Fig. 18. Page for downloading annotations in different formats.

approach, which relies on expert-derived strategies, while ML techniques enable the determination of feature weights through iterative data-driven adaptation.

(ix) ***Amount and shape of data***: various SemTab datasets have undergone processing using the s-elBat system. These datasets comprise hundreds (or thousands) of tables (as shown in Table 1) that necessitate processing through the complete s-elBat pipeline, encompassing both semi-supervised and supervised approaches. s-elBat excels in parallelising the Lookup phase and extracting crucial features to construct feature vectors, effectively addressing the challenge posed by the abundance of data.

s-elBat tried to resolve all key-challenges explained in Section 2 except the *NIL-mention* one. Several challenges have been effectively addressed, including *Matching tabular values against the KG* and *Disambiguation of named entities*. However, challenges like *Choosing the most appropriate types and properties* persist as open issues, despite s-elBat presenting a proposed solution. Future research will delve deeper into the issue of features to enhance the performance of ML models, with a specific emphasis on optimising features for literal values to improve the accuracy of matches in such cases.

LLMs like GPT have shown incredible performance and could be successfully used for Natural Language Processing tasks, surpassing existing approaches in the literature. Furthermore, these models have demonstrated remarkable robustness to noise, handling noisy inputs or text with typographical errors. However, when presented with zero-shot and one-shot prompts, these models have demonstrated poor performance in STI tasks.

It is worth highlighting the potential for future advancements in these models to enhance accessibility, thereby unlocking new possibilities and applications.

CRediT authorship contribution statement

Roberto Avogadro: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Fabio D'Adda:** Writing – review & editing, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Marco Cremaschi:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work was partially supported by the MUR under the grant "Dipartimenti di Eccellenza 2023-2027" of the Department of Informatics, Systems and Communication of the University of Milano-Bicocca, Italy.

Appendix A

In the following, we propose formalising the STI approach. The formalisation considers the definition of STI proposed in the previous section. To understand the outputs of the approach, a formalisation of the inputs, a table, and a KG, are proposed.

Definition 1. A rectangular array (matrix) of strings arranged in n rows and m columns is called a **table**. Every pair (i, j) with $1 \leq i \leq n$ and $1 \leq j \leq m$, unambiguously identifies a **cell** of the table.

Definition 2. Given an $n \times m$ table, let r_i denote respectively the ith *row* of the table, that is $r_i = \{(i, j) | 1 \leq j \leq m\}$ and c_j denote the jth *column* of the table, that is $c_j = \{(i, j) | 1 \leq i \leq n\}$. Let $R = \{r_i | 1 \leq i \leq n\}$ and $C = \{c_j | 1 \leq j \leq m\}$ be the set of all **rows** and **columns** of the table, respectively.

Definition 3. A function header $LA(c_j \rightarrow LA)$ that associates each column c_j of the table with a word of a language LA is called a Column Header Function.

Definition 4. The pair $T_h = (T, h)$, where T is a table and h is a column header function, we define $H = h(C)$ as the **header** of table T.

See Fig. 2 as an example of the elements just described.

The second input of STI is a KG. Inside a KG, it is possible to identify an ontology. Ontologies are structures for the organisation of knowledge in a particular domain. They are used to classify the terms and possible relationships and define possible constraints on using those terms.

Definition 5. An **ontology** is a multigraph $O = (S, P, A)$, where:

- $S = CO \cup DT$, is the set of **semantic elements** (*e.g.*, DBpedia Ontology, GeoNames Ontology);
  - CO is the set of **concepts** (*e.g.*, dbo:Movie, dbo:Director);
  - DT is the set of **datatypes** (*e.g.*, xsd:date, xsd:integer);
- P is the **property** label set (*e.g.*, "director", "pubblicationDate");
- A is a set of labelled directed **edges** $A \subset S^2 \times P$, where S is a set of semantic elements and P is the property label set. An edge can exist only between concepts (co_a, co_b) or between a concept and a datatype (co_a, dt_c) where $co_a, co_b \in CO$ and $dt_c \in DT$.

According to the definition proposed in [105] and the definition of an ontology described above, the definition of a *KG* is given as follows:

Definition 6. Given an ontology $O = (S, P, A)$, where S is a set of semantic elements, P is the property label set, and A is a set of labelled directed edges, a **knowledge graph** is a directed multigraph defined by the tuple $KG = (V, B, O, map, pmap)$ where:

- $V = E \cup L$ is the set of **vertices**;
  - E is a set of **entities** (*e.g.*,dbr:Jurassic_World, dbr:Colin_Trevorrow);
  - L is a set of **literals** (*e.g.*,“124”, “1670400637”);
- B is a set of directed **edges** connecting two vertices $B \subset V^2$, they represent links between entities, or between entities and literals;
- **map** is the ontology mapping function $map : V \rightarrow S$, where V is a set of vertices and S is a set of semantic elements, which links a vertex to a concept or datatype in the ontology (*e.g.*, dbr:Jurassic_World maps to dbo:Movie concept, dbr:Colin_Trevorrow maps to dbo:Person concept, “4808” maps to xsd:integer datatype);
- **pmap** is the property mapping function $pmap : B \rightarrow P$, where B is a set of directed edges and P is the property label set, which maps an edge to a property (*e.g.*, dbr:Jurassic_World dbo:director dbr:Colin_Trevorrow, dbr:Jurassic_World dbo:runtime “124”).

A **set of knowledge graph KGs** is defined as follows: $KGs = \{KG_1, KG_2, \dots, KG_x\}$.

Definition 7. The **CTA** task concerns the prediction of a set of concept for every given table column c_j in a table T, *i.e.*, $CTA(T, c_j, KG) = co_1, \dots, co_a$.

Definition 8. The **CPA** task concerns the prediction of semantic properties that represent the relationship between some pair of columns c_j and c_l, *i.e.*, $CPA(T, c_j, c_l, KG) = p_1, \dots, p_c$.

Definition 9. The **CEA** task aims to predict the entity e (*i.e.*, instances) that a cell $(i, j) \in T$ represents, *i.e.*, $CEA(T, (i, j), KG) = e$.

Definition 10. The **CNEA** task aims to predict which cell $(i, j) \in T$ represents an entity that does not occur in the KG and should be therefore labelled as NIL, *i.e.*, $CNEA(T, (i, j), KG) = ne$.

Appendix B. Ablation study

In this Section, an ablation study has been proposed. In particular, an experiment was conducted to analyse the impact of individual features of s-elBat supervised in identifying the entity to be used for annotation. The experiment consists of calculating the variation of the F1 score obtained on different datasets by perturbing the vectors, eliminating one feature at a time. This enables to evaluate the specific impact of each feature after fine-tuning.

In Fig. 19, the trend of the F1 score for the different feature sets is shown. For example, the red line shows the results obtained using all the features except “popularity” (the same applies to the other features). There is also a dashed blue line indicating the results obtained using all features, serving as a reference.

As can be seen, the only feature that does not significantly enhance the model is the “position score”. In certain instances, the ranking provided by LamAPI might confuse s-elBat, particularly when the ranking is sub-optimal (for example, if the correct entity is in the third position rather than the first). While this is not true for all datasets, it is clearly evident in SemTab2019_R3, as shown by the trend in the graph.

The numerical results are displayed in the Table B.6. As can be noted, removing the “Position Score” feature can yield a slight improvement (0.1%). The remaining features appear to be more relevant for disambiguation, especially for the textual scores (Jaccard, Description e Description Ngram).

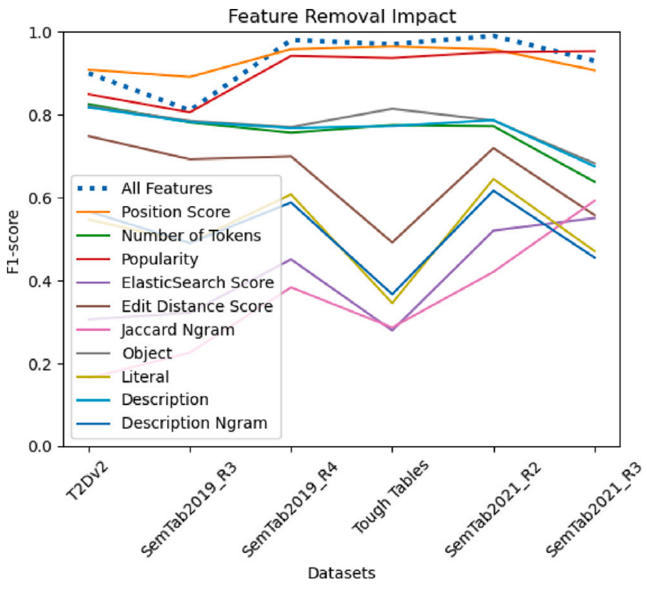


Fig. 19. F1 measure performance on different features sets.

Table B.6
Ablation study on features removal.

Feature	Average decrease on F1
Position Score	−0.1%
Number of Tokens	17.2%
Popularity	2.38%
ElasticSearch Score	52.5%
Edit Distance Score	27.8%
Jaccard Ngram	58.4%
Object	15.3%
Description	16.2%
Description Ngram	41.6%

Appendix C

See Tables C.7–C.12.

Table C.7
Sample data from the *SemTab 2019 R1 (T2Dv2)* dataset.

Party	Votes 2015	% votes 2015	Seats	Poll standing (16/4/15)
Conservative	8 895 066	37,3	254	34
Labour	9 199 875	38,6	291	34
Lib-Dem	2 519 729	10,6	37	8
SNP	1 004 774	4,2	42	–

Table C.8
Sample data from the *SemTab 2019 R3* dataset.

col0	col1	col2	col3
V. Camuto	L. Camuto	2015–01–21	1936–0–0
N. D. Reed	J. A. Reed	1991–09–08	1889–03–06
Irene	F. R. Jones	1962–11–15	1900–12–08
G. W. Ford	E. Ford	2008–08–24	1924–10–02

Table C.9
Example table from *SemTab 2020 R4 (Tough Table)* dataset illustrating name variations.

col0	col1	col2
Zooey Deschanel	Los Angeles	United States
Zooey Dechanel	Los Angeles	United States
Alanis Maurissette	Ottawa	Canada
Alanis Morisa	Ottawa	Canada

Table C.10
Sample data from the *SemTab 2021 R2* dataset.

col0	col1	col2	col3	col4
I Cainae	62,566 874	157,503 663 854 757 46	−4,695 3	51,290 4
42 Persel	10,782 491 8	57,730 546 066 606 14	−12,437 2	90,181
Deneb	2,312 31	311,909 769 651 765	−4,944 1	38,844
HD 3326	6,654 879	77,557 372 240 138 75	−4,391 2	15,075

Table C.11
Sample data from TURL test set dataset.

Team	Location	Stadium	Capacity
Johor FA	Larkin, Johor Bahru	Tan Sri Dato Hj Hassan Yunos Stadium	30,000
Johor FC	Pasir Gudang, Johor Bahru	Pasir Gudang Corporation Stadium	15,000
Kedah FA	Alor Setar	Darul Aman Stadium	32,387
Kelantan FA	Kota Bharu	Sultan Mohammad IV Stadium	20,000

Table C.12
Sample data from the *SemTab 2021 R3* dataset.

col0	col1	col2
tantalum-190	117	tungsten-190
tantalum-189	116	tungsten-189
tantalum-188	115	tungsten-188
tantalum-187	114	tungsten-187

References

[1] M. Marzocchi, M. Cremaschi, R. Pozzi, R. Avogadro, M. Palmonari, MammoTab: a giant and comprehensive dataset for semantic table interpretation, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab2022, 2022.

[2] 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.

[3] E. Jimenez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, V. Cutrona, Results of SemTab 2020, in: CEUR Workshop Proceedings, Vol. 2775, 2020, pp. 1–8.

[4] 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, 3103, CEUR Workshop Proceedings, 2022, pp. 1–12.

[5] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, S. Auer, P. Hitzler, Quality assessment methodologies for linked open data, Semantic Web J. 1 (1) (2013) 1–5, submitted for publication.

[6] 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, in: CEUR Workshop Proceedings, Vol. 2456, CEUR-WS.org, 2019, pp. 209–212.

[7] 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.

[8] G. Weikum, X.L. Dong, S. Razniewski, F.M. Suchanek, Machine knowledge: Creation and curation of comprehensive knowledge bases, Found. Trends Databases 10 (2–4) (2021) 108–490.

[9] M. Kejriwal, C.A. Knoblock, P. Szekely, Knowledge Graphs: Fundamentals, Techniques, and Applications, MIT Press, 2021.

[10] L. Ratinov, D. Roth, Design challenges and misconceptions in named entity recognition, in: Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), Association for Computational Linguistics, Boulder, Colorado, 2009, pp. 147–155.

[11] D. Rao, P. McNamee, M. Dredze, Entity linking: Finding extracted entities in a knowledge base, in: Multi-Source, Multilingual Information Extraction and Summarization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 93–115.

[12] W. Shen, J. Wang, J. Han, Entity linking with a knowledge base: Issues, techniques, and solutions, IEEE Trans. Knowl. Data Eng. 27 (2) (2015) 443–460.

[13] L. Wu, F. Petroni, M. Josifoski, S. Riedel, L. Zettlemoyer, Zero-shot entity linking with dense entity retrieval, in: EMNLP, 2020, Online.

[14] W. Shen, Y. Li, Y. Liu, J. Han, J. Wang, X. Yuan, Entity linking meets deep learning: Techniques and solutions, IEEE Trans. Knowl. Data Eng. (2021) 1.

[15] X. Li, Z. Li, Z. Zhang, N. Liu, H. Yuan, W. Zhang, Z. Liu, J. Wang, Effective few-shot named entity linking by meta-learning, 2022.

[16] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, SemTab 2021: Tabular data annotation with mtab tool, in: SemTab@ ISWC, 2021, pp. 92–101.

[17] T.M. Lai, H. Ji, C. Zhai, Improving candidate retrieval with entity profile generation for wikidata entity linking, 2022, arXiv preprint arXiv:2202.13404.

[18] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, F. Jiang, A. Gordon, C.-Y. Lin, LinkingPark: An automatic semantic table interpretation system, J. Web Semant. 74 (2022) 100733.

[19] C. Sarthou-Camy, G. Jourdain, Y. Chabot, P. Monnin, F. Deuzé, V.-P. Huynh, J. Liu, T. Labbé, R. Troncy, DAGOBAH UI: A new hope for semantic table interpretation, in: European Semantic Web Conference, Springer, 2022, pp. 107–111.

[20] G. Zhu, C.A. Iglesias, Exploiting semantic similarity for named entity disambiguation in knowledge graphs, Expert Syst. Appl. 101 (2018) 8–24.

[21] 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, 2022, Online.

[22] V. Cutrona, G. Puleri, F. Bianchi, M. Palmonari, NEST: Neural soft type constraints to improve entity linking in tables, in: SEMANTiCS, 2021, pp. 29–43.

[23] J. Raiman, O. Raiman, DeepType: Multilingual entity linking by neural type system evolution, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, (1) 2018.

[24] Y. Onoe, G. Durrett, Fine-grained entity typing for domain independent entity linking, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, (05) 2020, pp. 8576–8583.

[25] M. Cremaschi, R. Avogadro, D. Chieregato, MantisTable: an automatic approach for the semantic table interpretation, in: SemTab@ ISWC, 2019, 2019, pp. 15–24.

[26] M. Cremaschi, R. Avogadro, A. Barazzetti, D. Chieregato, E. Jiménez-Ruiz, MantisTable SE: an efficient approach for the semantic table interpretation, in: SemTab@ ISWC, 2020, pp. 75–85.

[27] R. Avogadro, M. Cremaschi, MantisTable V: A novel and efficient approach to semantic table interpretation, in: SemTab@ ISWC, 2021, pp. 79–91.

[28] R. Porrini, M. Palmonari, I.F. Cruz, Facet annotation using reference knowledge bases, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, WWW, Republic and Canton of Geneva, CHE, 2018, pp. 1215–1224.

[29] 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.

[30] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proc. VLDB Endow. 3 (1–2) (2010) 1338–1347.

[31] Z. Syed, T. Finin, V. Mulwad, A. Joshi, Exploiting a web of semantic data for interpreting tables, in: Proceedings of the Second Web Science Conference, Vol. 5, 2010, Online.

[32] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, V. Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: C. d'Amato, M. Fernandez, V. Tamma, F. Lecue, P. Cudré-Mauroux, J. Sequeda, C. Lange, J. Heflin (Eds.), The Semantic Web – ISWC 2017, Springer International Publishing, Cham, 2017, pp. 260–277.

[33] Y. Chabot, T. Labbé, J. Liu, R. Troncy, DAGOBAH: An end-to-end context-free tabular data semantic annotation system, in: SemTab@ ISWC, 2019, pp. 41–48.

[34] J. Chen, E. Jiménez-Ruiz, I. Horrocks, C. Sutton, Colnet: Embedding the semantics of web tables for column type prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 29–36.

[35] B. Kruit, P. Boncz, J. Urbani, Extracting novel facts from tables for knowledge graph completion, in: C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. Cruz, A. Hogan, J. Song, M. Lefrançois, F. Gandon (Eds.), The Semantic Web – ISWC 2019, Springer International Publishing, Cham, 2019, pp. 364–381.

[36] H. Morikawa, Semantic table interpretation using LOD4ALL, in: SemTab@ ISWC, 2019, 2019, pp. 49–56.

[37] D. Oliveira, M. d'Aquin, ADOG-annotating data with ontologies and graphs, in: SemTab@ ISWC, 2019, 2019, pp. 1–6.
[38] A. Thawani, M. Hu, E. Hu, H. Zafar, N.T. Divvala, A. Singh, E. Qasemi, P.A. Szekely, J. Pujara, Entity linking to knowledge graphs to infer column types and properties, in: SemTab@ ISWC, 2019, 2019, pp. 25–32.
[39] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, A. Gordon, C.-Y. Lin, Linkingpark: An integrated approach for semantic table interpretation, in: SemTab@ ISWC, 2020, Online.
[40] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin, R. Troncy, DAGOBAH: Enhanced scoring algorithms for scalable annotations of tabular data, in: SemTab@ ISWC, 2020, pp. 27–39.
[41] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, MTab4Wikidata at SemTab 2020: Tabular data annotation with wikidata, in: SemTab@ ISWC, 2775, 2020, pp. 86–95.
[42] V.-P. Huynh, J. Liu, Y. Chabot, F. Deuzé, T. Labbé, P. Monnin, R. Troncy, DAGOBAH: Table and graph contexts for efficient semantic annotation of tabular data, in: SemTab@ ISWC, 2021, pp. 19–31.
[43] L. Yang, S. Shen, J. Ding, J. Jin, GBMTab: A graph-based method for interpreting noisy semantic table to knowledge graph, in: SemTab@ ISWC, 2021, pp. 32–41.
[44] M. Cremaschi, R. Avogadro, D. Chieregato, S-elbat: a semantic interpretation approach for messy table-s, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), CEUR-WS. org, 2022.
[45] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, R. Troncy, From heuristics to language models: A journey through the universe of semantic table interpretation with DAGOBAH, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2022.
[46] V. Mulwad, T. Finin, Z. Syed, A. Joshi, T2LD: Interpreting and representing tables as linked data, in: Proceedings of the 2010 International Conference on Posters &#38; Demonstrations Track - Volume 658, in: ISWC-PD'10, CEUR-WS.org, Aachen, Germany, Germany, 2010, pp. 25–28.
[47] V. Mulwad, T.W. Finin, A. Joshi, Automatically generating government linked data from tables, in: AAAI 2011, 2011, Online.
[48] J. Wang, H. Wang, Z. Wang, K.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.
[49] V. Mulwad, T. Finin, A. Joshi, Semantic message passing for generating linked data from tables, in: H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J.X. Parreira, L. Aroyo, N. Noy, C. Welty, K. Janowicz (Eds.), The Semantic Web – ISWC 2013, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 363–378.
[50] C.S. Bhagavatula, T. Noraset, D. Downey, TabEL: Entity linking in web tables, in: M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, K. Thirunarayan, S. Staab (Eds.), The Semantic Web - ISWC 2015, Springer International Publishing, Cham, 2015, pp. 425–441.
[51] D. Ritze, O. Lehmberg, C. 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.
[52] Z. Zhang, Effective and efficient semantic table interpretation using tableminer+, Semantic Web 8 (6) (2017) 921–957.
[53] S. Zhang, K. Balog, Ad hoc table retrieval using semantic similarity, in: International World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, pp. 1553–1562.
[54] M. Cremaschi, F. De Paoli, A. Rula, B. Spahiu, A fully automated approach to a complete semantic table interpretation, Future Gener. Comput. Syst. 112 (2020) 478–500.
[55] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, MTab: matching tabular data to knowledge graph using probability models, 2019, arXiv preprint arXiv:1910.00246.
[56] B. Steenwinckel, G. Vandewiele, F. De Turck, F. Ongenae, Csv2kg: Transforming tabular data into semantic knowledge, 2019, SemTab, ISWC Challenge.
[57] N. Abdelmageed, S. Schindler, JenTab: Matching tabular data to knowledge graphs, in: SemTab@ ISWC, 2020, pp. 40–49.
[58] N. Abdelmageed, S. Schindler, JenTab meets SemTab 2021's new challenges, in: SemTab@ ISWC, 2021, pp. 42–53.
[59] N. Abdelmageed, S. Schindler, JenTab: Do CTA solutions affect the entire scores, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), CEURWS.org, 2022.
[60] R. Azzi, G. Diallo, E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, AMALGAM: making tabular dataset explicit with knowledge graph, in: SemTab@ ISWC, 2020, pp. 9–16.
[61] Y. Eslahi, A. Bhardwaj, P. Rosso, K. Stockinger, P. Cudré-Mauroux, Annotating web tables through knowledge bases: A context-based approach, in: 2020 7th Swiss Conference on Data Science, SDS, 2020, pp. 29–34.
[62] D. Kim, H. Park, J.K. Lee, W. Kim, E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, Generating conceptual subgraph from tabular data for knowledge graph matching, in: SemTab@ ISWC, 2020, pp. 96–103.
[63] R. Shigapov, P. Zumstein, J. Kamlah, L. Oberländer, J. Mechnich, I. Schumm, Bbw: Matching CSV to wikidata via meta-lookup, in: CEUR Workshop Proceedings, Vol. 2775, RWTH, 2020, pp. 17–26.
[64] S. Tyagi, E. Jimenez-Ruiz, Lexma: Tabular data to knowledge graph matching using lexical techniques, in: CEUR Workshop Proceedings, Vol. 2775, 2020, pp. 59–64.
[65] S. Zhang, E. Meij, K. Balog, R. Reinanda, Novel entity discovery from web tables, in: Proceedings of the Web Conference 2020, WWW '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1298–1308.
[66] W. Baazouzi, M. Kachroudi, S. Faiz, Kepler-aSI at SemTab 2021, in: SemTab@ ISWC, 2021, pp. 54–67.
[67] B. Steenwinckel, F. De Turck, F. Ongenae, MAGIC: Mining an augmented graph using INK, starting from a CSV, in: SemTab@ ISWC, 2021, pp. 68–78.
[68] N. Abdelmageed, S. Schindler, Jentab: A toolkit for semantic table annotations, in: Second International Workshop on Knowledge Graph Construction, 2021, Online.
[69] J. Liu, V.-P. Huynh, Y. Chabot, R. Troncy, Radar station: Using KG embeddings for semantic table interpretation and entity disambiguation, in: The Semantic Web–ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings, Springer, 2022, pp. 498–515.
[70] Y. Li, J. Li, Y. Suhara, A. Doan, W.-C. Tan, Deep entity matching with pre-trained language models, VLDB (2020).
[71] X. Deng, H. Sun, A. Lees, Y. Wu, C. Yu, Turl: Table understanding through representation learning, ACM SIGMOD Rec. 51 (1) (2022) 33–40.
[72] Y. Suhara, J. Li, Y. Li, D. Zhang, c. Demiralp, C. Chen, W.-C. Tan, Annotating columns with pre-trained language models, in: Proceedings of the 2022 International Conference on Management of Data, SIGMOD '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1493–1503.
[73] P. Li, Y. He, D. Yashar, W. Cui, S. Ge, H. Zhang, D.R. Fainman, D. Zhang, S. Chaudhuri, Table-GPT: Table-tuned GPT for diverse table tasks, 2023.
[74] T. Zhang, X. Yue, Y. Li, H. Sun, TableLlama: Towards open large generalist models for tables, 2023.
[75] I. Dasoulas, D. Yang, X. Duan, A. Dimou, TorchicTab: Semantic table annotation with wikidata and language models, in: CEUR Workshop Proceedings, 2023, pp. 21–37.
[76] E. Muñoz, A. Hogan, A. Mileo, Triplifying wikipedia's tables, in: CEUR Workshop Proceedings, in: LD4IE'13, CEUR-WS.org, Aachen, DEU, 2013, pp. 26–37.
[77] M.T. Pilehvar, J. Camacho-Collados, Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning, Morgan & Claypool Publishers, 2020.
[78] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Trans. Knowl. Data Eng. (2024) 1–20.
[79] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186.
[80] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C.L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, R. Lowe, Training language models to follow instructions with human feedback, 2022.
[81] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, LLaMA: Open and efficient foundation language models, 2023.
[82] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C.C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P.S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E.M. Smith, R. Subramanian, X.E. Tan, B. Tang, R. Taylor, A. Williams, J.X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open foundation and fine-tuned chat models, 2023.
[83] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, TinyBERT: Distilling BERT for natural language understanding, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 4163–4174.
[84] M. Cremaschi, A. Siano, R. Avogadro, E. Jimenez-Ruiz, A. Maurino, STILTool: a semantic table interpretation evaluation tool, in: The Semantic Web: ESWC 2020 Satellite Events: ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31–June 4, 2020, Revised Selected Papers 17, Springer, 2020, pp. 61–66.
[85] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.
[86] D. Ritze, C. Bizer, Matching web tables to dbpedia - a feature utility study, in: V. Markl, S. Orlando, B. Mitschang, P. Andritsos, K. Sattler, S. Breß (Eds.), 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017, OpenProceedings.org, 2017, pp. 210–221.

[87] 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.

[88] 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.

[89] W.S. Noble, What is a support vector machine? Nat. Biotechnol. 24 (12) (2006) 1565–1567.

[90] G. Biau, E. Scornet, A random forest guided tour, Test 25 (2016) 197–227.

[91] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.

[92] S.-C. Wang, S.-C. Wang, Artificial neural network, in: Interdisciplinary Computing in Java Programming, Springer, 2003, pp. 81–100.

[93] A.F. Agarap, Deep learning using rectified linear units (relu), 2018, arXiv preprint arXiv:1803.08375.

[94] N. Abdelmageed, J. Chen, V. Cutrona, V. Efthymiou, O. Hassanzadeh, M. Hulsebos, E. Jimenez-Ruiz, J. Sequeda, K. Srinivas, Semantic web challenge on tabular data to knowledge graph matching, in: International Semantic Web Conference, 2022, Online.

[95] P. Nguyen, N. Kertkeidkachorn, R. Ichise, H. Takeda, Mtab: Matching tabular data to knowledge graph using probability models, 2019, arXiv preprint arXiv:1910.00246.

[96] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, MTab4Wikidata at SemTab 2020: Tabular data annotation with wikidata, in: SemTab@ ISWC, Vol. 2775, 2020, pp. 86–95.

[97] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, SemTab 2021: Tabular data annotation with mtab tool, in: SemTab@ ISWC, 2021, pp. 92–101.

[98] J. Liu, R. Troncy, DAGOBAH: an end-to-end context-free tabular data semantic annotation system, in: SemTab@ ISWC, 2019.

[99] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin, R. Troncy, DAGOBAH: enhanced scoring algorithms for scalable annotations of tabular data, in: Semantic Web Challenge on Tabular Data To Knowledge Graph Matching (SemTab 2020) Co-Located with the 19th International Semantic Web Conference (ISWC 2020), Vol. 2775, 2020, p. 3.

[100] V.-P. Huynh, J. Liu, Y. Chabot, F. Deuzé, T. Labbé, P. Monnin, R. Troncy, DAGOBAH: Table and graph contexts for efficient semantic annotation of tabular data, in: SemTab@ ISWC, 2021.

[101] V.-P. Huynh, Y. Chabot, T. Labbé, J. Liu, R. Troncy, From heuristics to language models: A journey through the universe of semantic table interpretation with DAGOBAH, in: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab), 2022.

[102] J. Liu, Y. Chabot, R. Troncy, V.-P. Huynh, T. Labbé, P. Monnin, From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods, J. Web Semant. (2022) 100761.

[103] X. Li, S. Wang, W. Zhou, G. Zhang, C. Jiang, T. Hong, P. Wang, KGCODE-tab results for SemTab 2022, in: SemTab@ ISWC, 2022, pp. 37–44.

[104] M. Ripamonti, F. De Paoli, M. Palmonari, SemTUI: a framework for the interactive semantic enrichment of tabular data, 2022, arXiv preprint arXiv:2203.09521.

[105] B. Shi, T. Weninger, Discriminative predicate path mining for fact checking in knowledge graphs, Knowl.-Based Syst. 104 (2016) 123–133.