JenTab: Bridging Tabular Data and Knowledge Graphs – A Detailed System Overview

Nora Abdelmageed [a], Sirko Schindler [a] and Birgitta König-Ries [a,b]
[a] *Heinz Nixdorf Chair for Distributed Information Systems, Friedrich Schiller University Jena, Fürstengraben 1, 07743 Jena, Germany*
[b] *Michael Stifel Center Jena, Friedrich Schiller University Jena, Fürstengraben 1, 07743 Jena, Germany*
E-mails: nora.abdelmageed@uni-jena.de, sirko.schindler@uni-jena.de, birgitta.koenig-ries@uni-jena.de

Abstract.
Semantic Table Annotation (STA) stands as a crucial process in the realm of data interpretation and knowledge extraction, especially within the context of big data and the Semantic Web. Tables, ubiquitous across diverse domains from scientific literature to business reports, contain a wealth of structured information waiting to be unveiled. However, this information remains largely untapped in the absence of effective methods for semantic annotation. In essence, STA enriches raw tables with semantic metadata such as entities, classes, and relations obtained from Knowledge Graphs (KGs). It bridges the gap between unstructured data and structured knowledge representation, enabling sophisticated data analytics, information retrieval, and decision-making processes. It unlocks the potential of tabular data in the era of data-driven decision-making. However, automating this semantic annotation, particularly for noisy tabular data, remains a formidable challenge. In this paper, we give a detailed overview of our STA system, JenTab[1]. We developed and tested JenTab under the umbrella of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) challenge 2020-2022. However, we extended the evaluation of JenTab beyond the scope of these challenges. JenTab is a core system for STA, it is a top-3 ranked systems among other participants throughout its years of development. In addition, we present a detailed evaluation for its individual components, extensive discussion of JenTab with its limitations, and a demonstration of the system configuration, execution and output.

Keywords: Tabular Data, Knowledge Graph, Matching, JenTab, SemTab, Semantic Table Annotation

1. Introduction

Semantic Table Annotation (STA) is a pivotal process in the field of data interpretation and knowledge extraction, particularly in the age of big data and the Semantic Web. Tables, ubiquitous in various domains ranging from scientific literature to business reports, harbor a wealth of structured information waiting to be unlocked. However, this information remains largely unexplored without effective methods for semantic annotation. In essence, STA transforms raw data tables into rich sources of knowledge that machines can comprehend and analyze. By annotating tables with semantic metadata, such as entity labels, relations, and attributes, STA bridges the gap between uncontextualized data and structured knowledge representations, enabling advanced data analytics, information retrieval, and decision-making processes. In addition, it facilitates data integration, interoperability, and knowledge

[1]https://github.com/fusion-jena/JenTab

discovery across heterogeneous data sources. Moreover, STA plays a crucial role in various applications, including information extraction from documents, database augmentation, and semantic search. This caught the attention of a broad community where a variety of STA systems have developed in the past years [1].

The objective of STA is to map individual table elements to their counterparts from a KG as illustrated by a biodiversity example in Figure 1 (naming according to [2]): Cell Entity Annotation (CEA) matches cells to individuals, whereas Column Type Annotation (CTA) does the same for columns and classes. Furthermore, Column Property Annotation (CPA) captures the relationship between pairs of columns.

JenTab is a modular system to map table contents onto large KGs like Wikidata [3] or DBpedia [4]. This toolkit can annotate large corpora of tables. It follows a general pattern of Create, Filter, and Select (CFS): First, for each annotation, initial candidates are generated using appropriate lookup techniques (Create). Subsequently, the available context is used in multiple iterations to narrow down these sets of candidates as much as possible (Filter). Finally, a solution is picked from the remaining candidate(s) (Select). We provide several modules for each of these steps. Different combinations (pipelines) allow for fine-tuning the annotation process by considering both the modules' performance characteristics and their impact on the generated solutions. Besides the default pipeline that implements the complete picture of the CFS pattern, since 2021, we have continuously developed various pipelines based on the benchmark characteristics we tried to tackle.

We developed and tested JenTab during the participation in the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[2] challenge 2020-2022[3]. JenTab received a second place prize (Usability Track) of the SemTab challenge sponsored by IBM Research[4] during ISWC 2021 [5]. In addition, JenTab was awarded the Artifacts Availability Badge by SemTab 2022. All experiments are based on the large corpora provided by SemTab [6–10] matching the content to Wikidata, DBpedia, or schema.org. In addition, we evaluated JenTab in the biodiversity domain using the BiodivTab [11, 12] benchmark.

The core contributions distinguishing this paper from previous publications [13–16] are as follows with additional details being available in [17]:

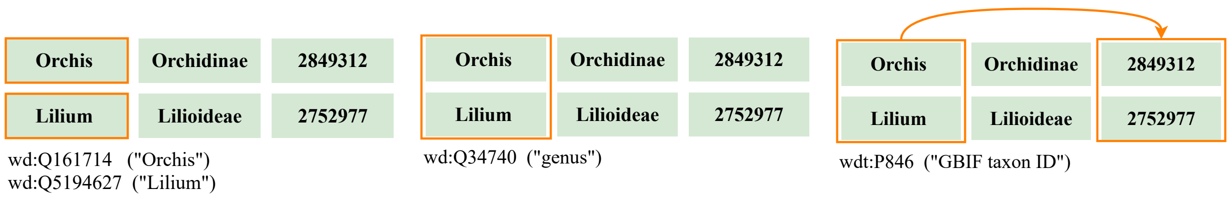
(i) Comprehensive overview of the related work.
(ii) Detailed evaluation comparing to other systems.
(iii) Inclusion of a new benchmark, tFood.
(iv) Assessment of its individual components.
(v) Extensive discussion of JenTab with its limitations.
(vi) Demonstration of JenTab configuration, execution, and output.

In this paper, we explain the required background in Section 2. We demonstrate a comprehensive overview of the state-of-the-art systems in Section 3. We present JenTab approach including its architecture, preprocessing, offline resources, disambiguation contexts, etc. in Section 4. We demonstrate various experiments to evaluate the performance of JenTab over the years of continuous development in Section 5. This includes a detailed analysis of individual components, the impact of different approaches, accuracy scores, and runtime. We follow with an extensive discussion of JenTab with its limitations in Section 6. Finally, we summarize this paper and conclude in Section 7.

2. Background

In this section, we explain the necessary background for this paper defining our primary data (tables[6]) and KGs. We explain two dimensions that describe tables: Inner-relationship and orientation dimension. Our source data is

[2]http://www.cs.ox.ac.uk/isg/challenges/sem-tab/
[3]A SemTab challenge has been held in 2023 but we did not participate and, hence, do not cover it here.
[4]https://research.ibm.com/
[5]We use the following prefixed throughout this paper:
wd: http://www.wikidata.org/entity/
wdt: http://www.wikidata.org/prop/direct/
dbr: http://dbpedia.org/resource/
dbo: http://dbpedia.org/ontology/
[6]We use the terms "tabular data" and "table" interchangeably.



(a) CEA. (b) CTA. (c) CPA.

Fig. 1. Illustration of STA tasks[5](from [12]).

raw in the sense that it lacks any semantic layer. A table consists of a set of cells, each of which contains plain text. Further, we define annotation tasks for both tabular and textual data. Such tasks aim to extract subjects, predicates, and objects which can be mapped to the target KG. We demonstrate an example for each task that describes the operations applied to perform a semantic annotation.

2.1. *Knowledge Graph*

A knowledge graph is a graph-based model built to accumulate and convey real-world knowledge. It contains a set of nodes and edges representing entities of interest and their relations [18, 19]. Auer et al. [20] propose them to bring scholarly communication to the 21st century. However, there is no exact definition of a KG. We adopt the inclusive definition from [18, 19] in Definition 2.1. A more formal definition of a KG is given by Definition 2.2.

Definition 2.1. *A knowledge graph is a graph of data used to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities.*

Definition 2.2. *A knowledge graph is a subset of the cross product $N \times E \times N$, where N is a set of nodes (entities or classes), and E is a set of edges (relations). Each member of this set is referred to as a triple.*

2.2. *Tabular Data*

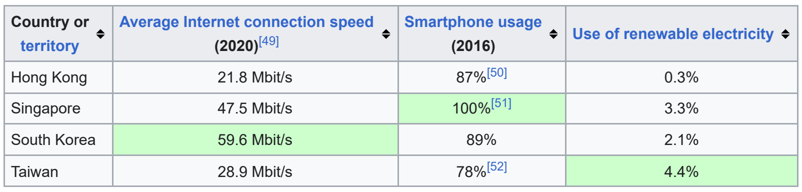
Tabular data, such as CSV files, are a common way to publish data and represent a precious resource [2, 5, 21]. Traditionally, they capture a lot of knowledge using free text entries. We define this data unit in Definition 2.3.

Definition 2.3. *A table is a two-dimensional arrangement of data with n rows and m columns. It enables a compact visualization for reading. A cell is the basic element of a table where $T_{ij}(0 \leqslant i \leqslant n, 0 \leqslant j \leqslant m)$ indicates the cell from row i and column j of the table T.*

Tables are highly heterogeneous in terms of structure, content, and purpose. Therefore, before interpreting a table, it is important to identify its type, so that specific characteristics can be taken into account in the Semantic Table Annotation (STA) process [22]. A table could be just layout or be encapsulating a certain amount of information [23]. The former is used for visualization (layout table). However, the latter expresses a topic or thing (genuine table). We express a genuine table in two dimensions [1, 24]: (i) Inner-relationship; a table can be Relational (Figure 2a), Matrix (Figure 2b), or Entity (Figure 2c, (ii) Orientation considers the direction of relationships inside a table. It can be horizontal, vertical, or matrix. Entities are described row-wise in horizontal tables (Figure 2a). In vertical tables, entities are described by a column as shown in Figure 2c. Matrix tables cannot be interpreted row by row or column by column but rather cell by cell while simultaneously considering both horizontal and vertical headers as given by Figure 2b. In our work, we focus on relational, horizontal tables and annotate them from a KG.

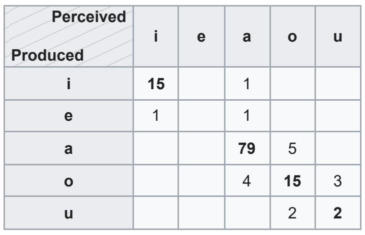
2.3. *STA Tasks*

We analyzed the state-of-the-art to investigate how to annotate tabular data from, e.g., a CSV file using a KG. Figure 3 summarizes our findings. It gives an overview of the five tasks of STA. First, **Cell to Instance** aims at linking a table cell value to a KG entity. In the shown case, “Egypt” would be linked to `wd:Q79` if the target



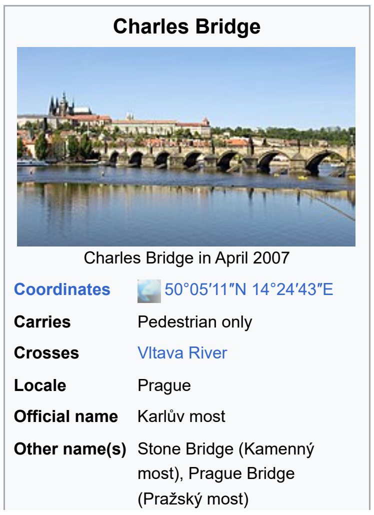
Country or territory	Average Internet connection speed (2020)[49]	Smartphone usage (2016)	Use of renewable electricity
Hong Kong	21.8 Mbit/s	87%[50]	0.3%
Singapore	47.5 Mbit/s	100%[51]	3.3%
South Korea	59.6 Mbit/s	89%	2.1%
Taiwan	28.9 Mbit/s	78%[52]	4.4%

(a) Relational (https://en.wikipedia.org/wiki/Four_Asian_Tigers).



Perceived / Produced	i	e	a	o	u
i	15		1		
e	1		1		
a			79	5	
o			4	15	3
u				2	2

(b) Matrix (https://en.wikipedia.org/wiki/Whistled_language).



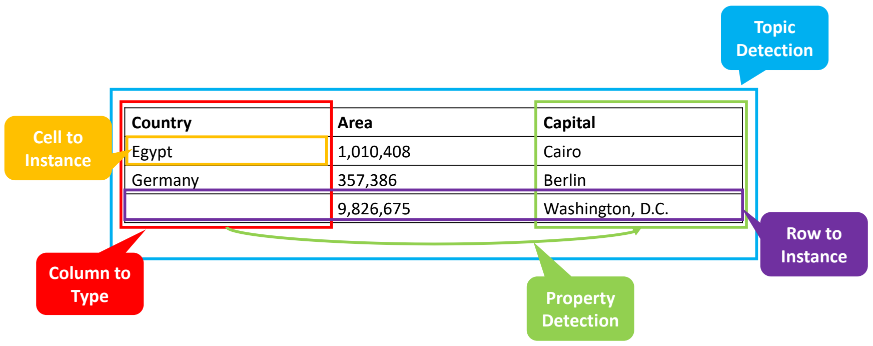
Charles Bridge

Charles Bridge in April 2007

Coordinates	50°05′11″N 14°24′43″E
Carries	Pedestrian only
Crosses	Vltava River
Locale	Prague
Official name	Karlův most
Other name(s)	Stone Bridge (Kamenný most), Prague Bridge (Pražský most)

(c) Entity (https://en.wikipedia.org/wiki/Charles_Bridge).

Fig. 2. Tables: Inner-relationship examples.



Country	Area	Capital
Egypt	1,010,408	Cairo
Germany	357,386	Berlin
	9,826,675	Washington, D.C.

Fig. 3. A summary of STA tasks.

KG is Wikidata or `dbr:Egypt` if the target KG is DBpedia. Second, **Column to Type** maps the entire column to a semantic type. In the example, it annotates the column to `wd:Q6256` if Wikidata is the target KG, or `dbo:Location` or `dbo:Place` if DBpedia is the target KG. Third, **Property Detection** links a column pair (subject-object) with a semantic property from the target KG. Country and capital columns would be linked through `wd:P1376` from Wikidata and `dbo:capital` from DBpedia. Fourth, **Row to Instance (R2I)** maps an entire row to a KG entity. Its output is different from the first task since the subject column is absent in this example. In this case, row to instance would be able to detect that the entire row refers to the "United States of America" (`wd:Q30`) or `dbr:United_States` from Wikidata and DBpedia respectively. Finally, **Topic Detection (TD)** classifies the entire table to a topic. Wikipedia articles are a perfect source of the expected output from this task. In the given example, https://en.wikipedia.org/wiki/Country would be a solution.

The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[7] challenge defines its three core STA tasks along these lines: CEA, CTA, and CPA (cf. Figure 1). It was co-hosted with the ISWC conference and the Ontology Matching (OM) workshop from 2019 to 2023. It provides a unified framework to evaluate the current STA systems. So far, SemTab, used Wikidata, DBpedia, and Schema.org as target KGs to perform the matching tasks. Each year, it provided a series of large-scale datasets. It asked participants to solve the three STA

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

tasks for given targets without any prior knowledge about the ground truth data for each task. After the conclusion of the challenge, the organizers published the hidden ground truth.

3. Related Work

The recent STA contributions (systems or benchmarks) are developed and tested in the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) challenge [2, 5, 21, 25] from 2019 to 2023 as illustrated. Such a challenge presents a unified framework to evaluate STA systems and provides several large-scale benchmarks for such evaluation. In the following, we give an overview of the existing approaches that solve STA tasks. We categorize them into three main categories: Heuristic methods, feature-engineering techniques, and deep-learning models. Table 1 summarizes these recent approaches that tackle the STA tasks in terms of supported STA tasks and target KG.

Table 1
Summary of the current state-of-the-art STA approaches.

Category	Approach	CEA	CTA	CPA	R2I	TD	KG	Released
Heuristic	MTab	Yes	Yes	Yes			Wikidata	2019-2021
	DAGOBAH	Yes	Yes	Yes			Wikidata, DBpedia	2020-2021
	bbw	Yes	Yes	Yes			Wikidata	2020
	CSV2KG	Yes	Yes	Yes			DBpedia	2019
	Tabularisi	Yes	Yes	Yes			DBpedia	2019
	MantisTable	Yes	Yes	Yes			DBpedia, Wikidata	2019-2021
	ADOG	Yes	Yes	Yes			DBpedia	2019
	LinkingPark	Yes	Yes	Yes			Wikidata	2020
	SSL	Yes	Yes	Yes			Wikidata	2020
	Magic	Yes	Yes	Yes			DBpedia, Wikidata	2021
	Kepler-aSI		Yes	Yes		Yes	DBpedia, Wikidata	2020-2023
	AMALGAM	Yes	Yes				Wikidata	2020
	s-elbat	Yes	Yes	Yes			Wikidata	2022
	TSOTSA	Yes	Yes	Yes		Yes	DBpedia, Wikidata	2022-2023
Feature-Engineering	Limaye	Yes	Yes	Yes			YAGO	2010
	Neumaier et.al		Yes				DBpedia	2016
Deep Learning	Efthymiou et. al	Yes			Yes		Wikidata	2017
	DAGOBAH-DL	Yes	Yes				DBpedia	2019
	ColNet		Yes				DBpedia	2019
	Chen et. al		Yes				DBpedia	2019
	Sherlock		Yes				DBpedia	2019
	Turl	Yes	Yes	Yes			DBpedia	2021

3.1. *Heuristic Techniques*

The category of heuristic techniques contains a wide range of approaches. Usually, a typical approach relies on candidate annotation generation via one or more lookup APIs, an iterative disambiguation process, and a final selection strategy. STA tasks are carried out using heuristic techniques, mostly string similarity and majority voting, followed by TF-IDF and probabilistic frameworks. In the following, we walk through heuristic approaches used by the state-of-the-art.

MTab [26–28] solves the three STA tasks jointly. The authors explicitly mention clear assumptions on what kind of table they solve, e.g., the target KG is complete and correct, and input tables are independent, which means there is no sharing of information between input tables. MTab depends on a joint probability distribution for the three tasks. The authors introduce a pipeline of seven steps. The lookup part is built on top of multiple lookup services. It is started with DBpedia lookup and endpoint in 2019. In 2020, the authors identified weaknesses from the previous year; they changed their lookup module to support both fuzzy entity and statement search. They supported Wikidata as a target KG, including the entity dump and all history revisions in a given time interval to generate as precise candidates as possible. MTab is computationally expensive, and the authors had to disable various parts to be able to release it to the public in 2021[8]. MTab is the SemTab first place winner in 2019 and 2020 in the Accuracy Track, and they achieved first place prize in the Usability Track in 2021.

DAGOBAH [29, 30] constructs an annotation workflow to solve the three STA tasks. In the first step, the authors perform an entity lookup from Wikidata dump which is stored in a Hadoop distributed file system, using Wikidata toolkit[9] and PySpark[10] for candidate annotation retrieval using Levenstein distance for similarity measurement. In the second step, they follow a candidate scoring technique. A confidence score is calculated as a mixture of a context score using Levenshtein distance and SMAPE for literal values and numbers, respectively, and a weighted similarity score for literals. Then, they continue with CPA annotations. The authors retrieve all entity candidates and select the most co-occurred property, a straightforward majority voting technique. Afterwards, the authors solve the CEA task by selecting a final candidate with maximum CPA score support. Finally, they come up with CTA solutions. Similar to selecting CEA solutions, CTA candidate is selected by a majority vote of CEA in the same column. In addition, the authors take into account three levels of hierarchical types where they prefer the direct types in most cases. DAGOBAH required 250 machines to run its 2020 configurations, although the authors managed to reduce it to 30 machines in 2021. Moreover, it is the third place winner at SemTab 2020 and a first place winner at SemTab 2021.

bbw [31] solves the three STA tasks. bbw consists of a seven-step pipeline; we summarize its steps into two core ideas. First, for candidate generation, the authors use a locally deployed SearX[11] instance as a meta-lookup which enables search over more than 80 engines. The authors do not use any Wikidata dump. This module collects the lookup results and ranks them automatically. Second, the authors rely on the Wikidata SPARQL endpoint for contextual matching using two features: entity and property labels. Then, the authors pick the best matches using edit distance. For the CTA solutions, they select the type through `wdt:P31 (instanceOf)` without any further exploration of multi-hop hierarchical types, achieving the best score on CTA with 98% F1-score during the challenge. bbw won the third place of SemTab 2020.

CSV2KG [32] addresses the three tasks of STA. The authors follow an iterative process with the following steps: (i) get entity matchings using lookup services; the authors adopt several services of DBpedia: DBpedia lookup service, DBpedia Spotlight, and DBpedia resource. Each service returns a ranked list of candidates. On this large pool of candidates, they apply a disambiguation step by selecting the candidate with an `rdfs:label` has the lowest edit distance with the cell value. (ii) infer the column types and relations; the majority vote is the used strategy of the final selection. The authors reported some inaccuracies on the DBpedia level, e.g., Barack Obama and not Donald Trump is the president of USA[12]. (iii) refine cell mappings with the inferred column types and relations. (iv) refine subject cells using the remaining cells of the row. Finally, (v) re-calculate the column type with all the corrected annotations. CSV2KG is the second-place winner at SemTab 2019.

Tabularisi [33] solves the three STA tasks. The authors use lookup services to generate CEA candidates. For each candidate, an adapted TF-IDF score is calculated. An entity candidate is represented by a binary feature vector in which each feature is an indicator (1 if present, else 0) of a property used to describe the entity (e.g., instanceOf). Different features have various expressiveness. They are thus weighted by the TD-IDF technique. Specifically, the "Term Frequency" of a feature is the number of cells whose first entity candidate has this feature, and the "Document Frequency" is the total occurrences of this feature in all entity candidates of all cells. The score of an entity candidate is a weighted combination of its TD-IDF score, the Levenshtein distance between cell value and candidate label,

[8]https://mtab.app/mtabes/docs
[9]https://github.com/Wikidata/Wikidata-Toolkit
[10]https://spark.apache.org/docs/latest/api/python/
[11]https://github.com/searx/searx
[12]Checked on Late Feb 2020.

and a distance measure between cell value and the URL tokens is used to determine the final annotation. The CTA solutions are obtained by a top-down, brute-force search in the KG class hierarchy tree. Finally, the CPA solutions are determined using a row-by-row majority voting technique.

MantisTable [34] tackles the three set of the STA tasks. The authors introduce a pipeline that starts with classifying each column into three types: Named entity, Literal, or Subject. The candidate generation of this approach is based on SPARQL queries which extract all candidates containing the cell mentions. Then, the system handles the CEA annotations by row-wise compatibility analysis and CPA by majority voting. For the CTA task, the authors list all candidate types in addition to their number of occurrences in the table (row coverage). After threshold filtering, the remainder of type candidates are transformed into a graph according to the ontology hierarchy. The type scores are then updated with the distance to the root. The highest score representing the most accurate and specific annotation at the end is finally picked as a solution.

MantisTable SE [35, 36] the authors optimized the system by updating the scoring function, accessing LamAPI[13] (instead of using a SPARQL endpoint) and adding a final disambiguation step. LamAPI is considered an efficient way to retrieve all necessary data for the three tasks independently from the target KG.

MantisTable V [37] is an even more optimized version that still relies on LamAPI and applies complex string similarity functions for the entity generation task.

ADOG [38] solves the three STA tasks. It considers scores combined with string similarities, frequencies of properties, and the normalized Elasticsearch score from each match of DBpedia for the CEA task. The system weighs these scores with TF-IDF score for types. To be able to compute the Levenshtein distance and TF-IDF, the authors use ArangoDB[14] to load DBpedia dump and index its components.

LinkingPark [39] tackles the three STA tasks. Their framework consists of three main modules: i) an entity linker to generate CEA, ii) a type inference to get CTA and, finally, iii) a property linker to obtain the possible relations. The entity linker contains two sub-modules for entity generation and disambiguation. The authors use a regular AccessMediaWiki API search, 1-edit distance typo via a spelling mistake corrector, and other mentions that are used to build a fine-grained Elasticsearch index as a cascaded pipeline for entity generation. For entity disambiguation, the authors follow a coarse-to-fine-grained disambiguation. They rely on an iterative classification algorithm to conduct an approximate inference of an entity. Afterwards, they would able to characterize both types' consistency (coarse) and select the discriminative value. The type inference retrieves the types of the generated CEA candidates. Then it uses the majority vote to select a final CTA candidate. To break the ties, the authors use a minimum average level score that is defined and based on the underlying ontology structure of the target KG. In the property linker, the author support object properties matching by the entity property linker sub-module and literals matching using the either perfect or fuzzy matching technique. For the object properties, the author obtains the relations row-by-row and finally, selects the most co-occurred one as a final selection. The same procedure is applied for the lexical property match but with an error tolerance to allow fuzzy matching. LinkingPark is the second-place winner at SemTab 2020.

SSL [40] generates a Wikidata subgraph over a table using a four-stage pipeline to solve the three STA tasks. At first, it leverages advanced SPARQL queries for the three tasks. Then, it selects the subject with the highest probability value to update the resultant graph. Afterwards, the authors apply a crawling process through Google search engine to suggest better words for not-found subjects and repeat the first two steps. Thus, the authors overcome the problem of spelling mistakes. Finally, for literal properties matching, the authors tolerate $\pm 1.5\%$ of the numerical values.

Magic [41] addresses the three STA tasks. It adopts the approach of generating comparison matrices, namely INK embeddings, to speed up computational efficiency. INK embeddings are representations of attributes and values for an entity or table context for a cell mention. The complete comparison matrix is generated by fusing multiple candidates. The system annotates CEA by measuring the compatibility between INK embeddings from KG and the input table. For annotations, the authors focus on the key column. They do the lookup using public endpoints of the target KG for each cell in the key column, then leverage its neighborhood to find the candidates for surrounding

[13]https://bitbucket.org/disco-unimib/lamapi
[14]https://www.arangodb.com/

cells in the same row (they avoid performing the lookup on the whole table due to the limitation of public API usage). Misspellings might be challenging for Magic, and detecting synonyms of attributes as well. Nonetheless, INK embeddings improve computational efficiency and provide a way to implement column integration.

AMALGAM [42] covers both CEA and CTA only using a three-step pipeline. The authors rely on Wikidata lookup services with a focus on spelling mistakes handling. They use the row context where all entities belong to the same thing as the only annotation context. This gives a reason why AMALGAM is efficient and less computationally expensive. Results show a maximum score of 92% for both tasks.

Kepler-aSI [43–45] is designed with three main modules: Preprocessing, Annotation Context, and Tabular Data to KG Matching. The Preprocessing module handles diverse data types and formats, rectifying issues like spelling errors and noisy textual data. The Annotation Context phase involves identifying candidates for annotation by analyzing processing columns and employing regular expressions. Unlike other systems, semantic types are assigned to columns using entities from Wikidata or DBpedia first then use these solutions to solve the rest of the tasks.

s-elbat [46] improves the technique of MantisTable (see above) by adopting an iterative process for entity linking on tables. It consists of two modules: 1) Entity Linking (EL) that uses various techniques including search engine-based and name dictionary-based methods to filter out irrelevant entity mentions for a given table entity. 2) Entity Disambiguation (ED) that leverages contextual information besides the cell value to re-rank entities.

TSOTSA [47, 48] introduces a KG refinement approach to solve STA, focusing on error correction and completion of missing entities and relations. The pipeline for annotating tabular data involves cell pre-processing and completion with missing entities and relations using SPARQL queries. The methodology has been changed to leverage the Bayes classifier to solve STA in 2023.

3.2. Feature-Engineering Techniques

This category relies on the hand-crafted feature vectors that represent a table. For example, authors of this category extract statistical and lexical features, such as the distribution of numerical values, the occurrence of cell mentions, and textual similarity among table rows and columns. Then, they feed such features to machine learning models like SVM, Random Forest, and KNN algorithms. Since these are all supervised learning techniques, a labeled dataset is required for the training. The amount and quality of training data, and consequently the quality of input features, significantly impact the model performance. We notice that those techniques suites the CTA task more than the others, since columns are more subject to statistical features than other STA tasks.

Limaye [49] introduce one of the earliest work on STA that tackels the the three tasks. The approach collects the TF-IDF cosine similarity between cell mention and entity label and the compatibility between the cell type and the column type to execute the CEA task. The solution to the CTA task depends on TF-IDF cosine similarity between column header and entity label in the corresponding KG. The CPA annotation depends on the compatibility between the relation and column pairs. All these features are weighted through a machine learning framework.

Neumaier et.al [50] focus on CTA labeling for numerical columns. Their work is not limited to labeling a unique prediction but expands the scope of labeling to the surrounding information. For example, instead of labeling "height", this system will label it as "the height of an athlete playing basketball in the NBA". To achieve this, the authors proposed three steps pipeline: i) build Background Knowledge Graph (BKG) based on DBpedia where its nodes consist of typical numerical values, annotated with context information. For example, grouped by properties and their shared domain (subject) pairs. Such BKG contains the hierarchical structure and is divided into multiple multi-level groups to provide context. ii) to search for mappings using k-nearest neighbours (kNN) for making predictions. Finally, iii) to aggregate the results at different levels of the BKG to find the most likely context in terms of properties and types. The authors also explore the system's performance at different hierarchy levels for the BKG built on DBpedia and Open Data. They pointed out that DBpedia still has limitations in terms of data coverage and freshness when compared with other open datasets. For example, the Austrian Open Data Portal has tables generated by weather stations every 15 minutes. Comparatively, DBpedia typically has numeric values only for "current" or "latest".

3.3. Deep Learning Techniques

Deep Learning has achieved many successes in various domains thanks to the availability of massive amount of data and powerful computing resources. It has attracted the attention of the STA community over the past few years. KG embedding techniques represent one direction beyond this work's scope. However, in the following, we show some examples that leverage deep learning as a direct NLP model in the scope of STA.

Efthymiou et. al [51] provide different methods for entity linking, regular cell to an entity annotation (CEA), as well as mapping the entire row to instance (R2I). The system assumes that a column's correct CEA candidates should be semantically close. From this assumption, a weighted correlation subgraph in which node represents a CEA candidate is constructed. The edges are weighted by the cosine similarity between two related nodes. The best candidates are the ones whose accumulated weights over all incoming and outgoing edges are the highest. In addition, a hybrid system, as a combination of a correlation subgraph method and an ontology matching system, is also introduced, which achieves a significant improvement in the final results.

DAGOBAH-DL [52] Solves the three STA tasks The authors assume that entities in the same column are close in the embedding space. Candidates are first retrieved using a lookup based on regular expressions and the Levenshtein distance. Then, they convert the retrieved entities into vector space. The selection of the candidate is made on several steps starting from the clustering formulation (rows coverage-wise). The authors apply a K-means clustering using TransE's pre-trained embedding to cluster the entity candidates. Then they give each cluster a score, pick the one with the highest score, and rank its candidates. The final ambiguity is resolved via a confidence score based on the row context of the candidates.

ColNet [53] tackles only the CTA task. The authors use a Convolutional Neural Network (CNN) trained by classes contained within a KG. The predicted annotations are combined with the results of a traditional KG. The final annotation is selected using a score that selects the lookup solutions with high confidence and otherwise resorts to the CNN predictions. Results have shown that CNN prediction outperforms the lookup service for a larger knowledge gap. In addition, the authors explored the idea of ensembling the predictions from ColNet and those from the lookup service. Such an ensemble strategy gave the best scores since it benefits from both techniques. Afterwards, the authors extended the entity representation by considering other cells in the same row (row context). Thus they create a property feature vector, a.k.a, Property to Vector (P2Vec). Such P2Vec is an additional signal to the neural network, which yields better results [54].

Sherlock [55] Learns the CTA task using 1588 features extracted from a single column of a given relational table. The features are divided into four categories: i) characterwise statistics (e.g., frequency of the character "c"), ii) column statistics (e.g., mean, std of numerical values), iii) word embedding, and iv) paragraph embedding. Except for the column statistics feature, other features are compressed into a fixed embedding size using a subnetwork. A two-fully connected layer network is trained on both the embedding features and column statistics feature to predict a column type annotation among 75 types inherited from T2Dv2 [56] dataset. The evaluation shows limited results on various column types, including Dates and Industry. However, it is less sensitive to purely numerical values or values appearing in multiple classes.

Turl [57] pioneers the application of pre-trained language models such as BERT in the STA tasks. It provides a universal contextualized representation for each table element (i.e., caption, header, content cells), which can be fine-tuned and applied in various downstream tasks such as CEA, CTA, and CPA. The model employs a Transformer-based encoder [58] to capture the information from table elements. For this goal, the input table is first serialized into a sequence of caption tokens, title tokens, header tokens, and row-by-row cells. A cell consists of its content (mention) and a candidate entity representing it in a KG. The sequence of tokens is then converted into embedding using a word embedding for textual tokens and KG embedding for entity tokens. To reduce the redundancy in the fully connected attention learning and better draw the inter- and intra-column, inter-and intra-row, and column-row interactions, the conventional attention layer is masked by a so-called visibility matrix which allows only a portion of table elements to participate in the modeling of a specific element. For example, cells in the same row or the same column can interact with each other. Apart from the BERT's Masked Language Model objective, TURL introduces an additional Masked Entity Recovery objective to reinforce the learning of factual knowledge embedded in the table and represented by KG entities. The model is pre-trained on $570K$ unlabeled Wikipedia tables.

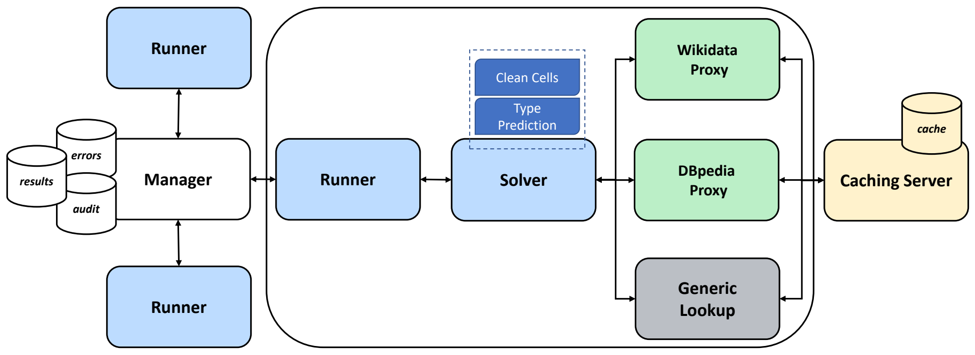


Fig. 4. JenTab: Current system architecture (from [14]).

4. JenTab Toolkit

In this section, we explain the system architecture of JenTab. We outline the preprocessing steps executed before JenTab's actual pipeline. Then, we explain the offline resources calculated before the online run of JenTab and the disambiguation contexts used to annotate a table. Finally, we explain the CFS pattern and the default pipeline of JenTab with its variations.

4.1. Architecture

Figure 4 illustrates the most recent system design of JenTab. For the previous architecture, please refer to [13]. We opted for a distributed approach allowing to split the workload across several nodes. A central "*Manager*" node orchestrates a family of associated "*Runner*" nodes. Runners contact the Manager to request new work items, i.e., raw tables. After a work item is finished, results are sent back to the Manager, and the next one is requested. The result of processing a single table consists of three parts: *results* correspond to annotations, *audit data* that allows assessing the impact of individual modules, and possibly a list of any *errors* raised during the processing. The Manager's dashboard contains information on the current state of the overall system, i.e., processed and still queued tables. In addition, it contains data about connected Runners, errors raised (if any), and an estimate of the remaining processing time. Finally once finished, all gathered annotations can also be accessed. A Runner coordinates the processing of a single table at a time through a series of calls to different services. "*Solver*" nodes are the core service that perform the pre-processing steps and execute the chosen pipelines. This includes data cleaning and primitive type detection (cf. Subsection 4.2). Conceptually, a Solver depends on two kinds of services: A Proxy service to communicate with the target KG. We implemented a "*Wikidata Proxy*", and a "*DBpedia Proxy*" encapsulating Lookup APIs meant for fuzzy search and the SPARQL query endpoints of the corresponding KG. Further, a Solver depends on a Generic Lookup — our primary means of tackling spelling mistakes. Proxy services and the Generic Lookup rely on a centralized "*Caching Server*" reducing the number of queries issued by caching already retrieved responses.

The developed architecture has several advantages. First, caches for computationally expensive tasks or external dependencies increase the overall system performance. Further, it reduces the pressure on downstream systems, especially when public third-party services are used. Second, when the target KG is to be substituted, all necessary changes, like adjusting SPARQL queries, are concentrated within just two locations: the corresponding lookup and endpoint services. Third, the distributed design allows for scaling well regarding the number of annotated tables. Any increase in the number of tables can be mitigated by adding new Runners to cope with the workload. Finally, the implementation allows for reusable and self-encapsulated pieces of code.

4.2. Preprocessing

Before executing the actual pipeline, each table is run through preprocessing steps. We group them in "Data Cleaning" and "Datatype Predicition". We give their details as follows:

Data Cleaning First, we apply a series of steps to fix data issues in a table.

1. We start with using *ftfy*[15] to fix any encoding issues encountered.
2. We use a regular expression to split up terms that are missing spaces like in "*1stGlobal Opinion Leader'sSummit*" into "*1st **G**lobal Opinion Leader'**s** **S**ummit*".
3. We remove certain special characters like parentheses.
4. We replace the explicitly mentioned unknown values to null like "NA", "Unknown", "Undetermined" etc.
5. We capture the actual value of the classified date type cells. For example, "2010-11-23 November 23, 2010" is translated to only "2010-11-23".
6. We apply an off-the-shelf spell checker, *autocorrect*[16], to fix typos resulting in an "autocorrected value" per cell.

The result of these steps is stored as a cell's "clean value". We use this cleaned value during the actual execution of JenTab's pipeline. We deprecated the *autocorrect* step in 2021 due to its unstable behavior. Some values are corrected in unrealistic words, i.e., do not fit in the given context. For example "Rashmon" that is expected to be "Rashomon" (a movie from 1950) is converted to "Fashon". Instead, we relied on the Generic Lookup (explained in detail in Subsection 4.3). We apply such a series of steps to facilitate the CEA candidates generation by providing a cleaner version of the given cell values. In return, we expect a higher performance of candidates matching.

Datatype Prediction We determine the primitive datatype of each column. While the system distinguishes more datatypes, we aggregate the ones having a direct equivalent in Wikidata as the following four types:

1. `OBJECT` columns represent entities, those we solve CEA task for them since we check the given targets to solve such task.
2. `STRING` columns that were classified as `OBJECT` but not found among the given targets. They would be mapped to, e.g., entity labels or descriptions.
3. `DATE` columns represent date literals, which could be mapped later for a property like inception, start date, etc.
4. `NUMBER` columns represent numerical values that would be mapped to literal properties as well.

Besides locating the entities of interest that would be mapped to CEA, such classification helps us develop a datatype-specific property matching technique. For example, we solve the CPA task differently if the given column is `DATE` or `NUMBER`.

4.3. Offline resources

In this section, we list the steps we conducted before the live run of JenTab. Usually, these steps are conducted to solve a specific issue we have encountered in datasets. For example, Generic Lookup and Tough Tables (2T) Cleaning are meant to fix spelling mistakes but for different conditions. The Biodiversity Dictionaries are developed to tackle the problem of abbreviations in biodiversity datasets.

Generic Lookup Spelling mistakes and artificial noise are common challenges across STA benchmarks. A specialized lookup is our primary strategy for tackling this crucial issue. We created it ahead of time due to the resources required for comparing cell values against all labels (and aliases) within Wikidata or DBpedia. We extracted the unique values from all tables of a given dataset and matched those against labels of the respective KG using an optimized Jaro-Winkler Similarity implementation based on [59] and a threshold for minimum similarity of 0.9.

[15]https://github.com/LuminosoInsight/python-ftfy
[16]https://pypi.org/project/autocorrect/

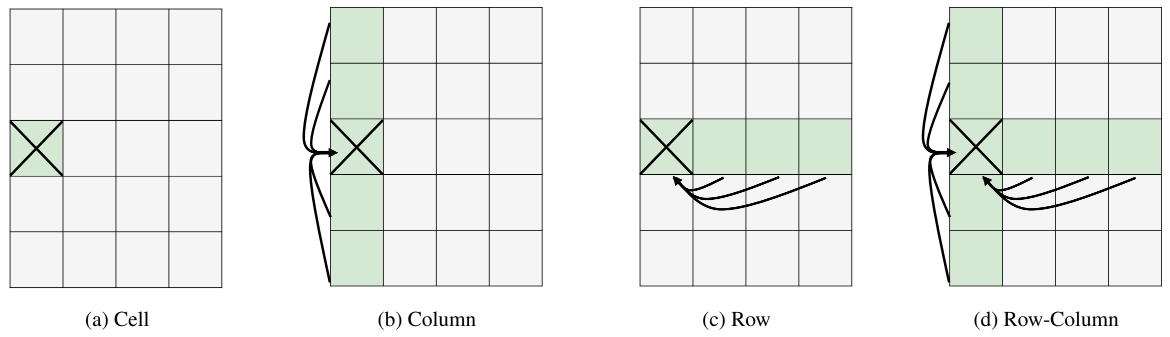


Fig. 5. Possible contexts for resolving and disambiguating annotations for subject cells only. Arrows indicate information used (from [15]).

Such an approach will fetch and store all labels that have a similarity between 1 (exact matches) and 0.9. The result is a sqlite[17] database which has the highest priority for the lookup during CEA-candidate generation.

Biodiversity Dictionaries In 2021, we built biodiversity dictionaries for species and chemical elements or compounds to mitigate the impact of abbreviations. We constructed such dictionaries based on Wikidata. We queried Wikidata with all instances of the taxon (`?species wdt:P31 wd:Q16521`) and programmatically fetched the corresponding labels. From these labels, we created abbreviated variations by using one or two characters from the first word followed by the full second name. So, "Canna glauca", e.g., was converted into both "C.glauca" and "Ca.glauca". These variants were subsequently used as aliases for the respective Wikidata entities. The corresponding lookup was used with the highest priority during our initial creation of CEA-candidates in the biodiversity setting.

2T Cleaning In 2022, we developed a cleaning module for 2T [9] dataset. It is known for the huge amount of artificially added misspellings to its tables. We have developed this strategy since our previously discussed Generic Lookup failed to catch these noisy elements. Thus, our core idea is to locate the cells that are correctly spelled and then replace all the artificial occurrences with the correct word. The first step is to find the correctly spelled words by querying those cells in Wikidata. Those with exact matches are considered correct words. The second step is to match the remaining values in the tables to the correctly identified values. We converted all the given cells into the embedding space using fasttext [60] to avoid the out-of-vocab (OOV) problem. Then, we applied cosine similarity among those vectors; we picked the correct target value if the similarity is $\geqslant 70\%$. We ran this step offline before the actual running of JenTab to solve the STA tasks. Thus, JenTab can work on relatively cleaner tables than those provided in the original 2T dataset.

4.4. Disambiguation contexts

Tabular data offers different dimensions of context that can be used to either generate annotation candidates (Create-Phase) or remove highly improbable ones (Filter-Phase). Figure 5 illustrates this visually. The *Cell Context* is the most basic one, outlined in Figure 5a. Here, nothing but an individual cell's content is available. We can then define a *Column Context* as shown in Figure 5b. It is based on the premise that all cells within a column represent the same characteristic of the corresponding tuples. For the annotation process, this can be exploited insofar that all cells of a column share the same class from the KG. Annotations for cells in `OBJECT`-columns further have a common class as required by the CTA task. Similarly, the assumption that each row refers to one tuple leads to the *Row Context* of Figure 5c. Annotation candidates for the subject cell, i.e., a cell holding the identifier for the respective tuple/row, have to be connected to their counterparts in all other cells within the same row. Finally, all contexts can be subsumed in the *Row-Column Context* as given by Figure 5d. It combines the last two assumptions representing the most exhaustive context.

[17]https://www.sqlite.org/index.html

4.5. CFS pattern

We base our approach on collecting mostly independent building blocks for each task following the pattern of the CFS approach. The individual building blocks differ in what information they use and how accurate their results are. They further differ in their performance characteristics: On the one hand, this refers to the time needed to execute them. On the other hand, creation and filtration blocks vary in the number of candidates they output.

We maintain sets of candidates on different levels: For each cell, we maintain both the candidates for the respective CEA task and those induced by this cell for the corresponding CTA task. Each pair of cells in the same row has a set of candidates for the respective property connecting both cells contributing to the CPA task. The same is mirrored on the column level. Here, we keep the onset of candidates for the CTA task and CPA candidates for the combinations of columns.

Building blocks usually pertain to only one task as well as one CFS-stage: *Create* blocks generate candidates of possible solutions for the respective task. *Filter* blocks reduce given sets of candidates. Here, we take a rather conservative approach and only remove candidates that will not be a solution with a very high probability. Finally, *Select* blocks pick a solution for a given set of candidates. Figure 6 summarizes the developed building blocks which will be described in detail below. Unless noted otherwise, the building blocks only apply to columns of datatype `OBJECT` as classified in the preprocessing. Further to query the KG, we rely on the official SPARQL endpoint of Wikidata[18] which imposes specific rate and execution time restrictions on our queries.

4.6. Create

In the following, we describe different building blocks that generate candidates for individual tasks.

CEA Label Lookup: The Label Lookup is the foundation of the pipeline. It does not depend on any prior information other than the cell's content and their datatype. We apply a series of strategies to retrieve candidates based on the cells' initial, clean, and autocorrected value. As each strategy succeeds in different cases, they are applied in sequence until candidates are retrieved by one. All but the Generic Strategy use the Wikidata Lookup Service[19] to resolve from a given label to Wikidata entities.

– *Generic Strategy* compares the initial cell values with all Wikidata labels and aliases using the Jaro-Winkler distance [61]. We iteratively[20] lower the selection threshold from 1 (exact matches) to 0.9 until a set of candidates is returned.
– *Full Cell Strategy* uses the initial cell values to query the lookup service.
– *All Tokens Strategy* splits a cleaned cell value into tokens removing all stopwords. The lookup service is then queried with all possible orderings of these tokens.
– *Selective Strategy* removes any addition in parenthesis and uses the remainder to query the lookup service.
– *Token Strategy* splits the cleaned cell value into tokens and queries for each token in isolation.
– *Autocorrection Strategy* uses the autocorrected value from the preprocessing to query the lookup service.

CEA by row and column: This approach applies in cases where we could determine candidates for at least some cells of datatype `OBJECT` within a row but failed for the subject cell. Furthermore, for the subject column, existing candidates are required. If all conditions are met, we retrieve entities from the KG that are instances of a subject column candidate and have a connection to at least one of the other candidates in the same row. Subsequently, these entities are filtered such that the remaining ones have a connection to each object cell in the same row. Finally, we compute the string distances between the remaining labels and the initial cell value and discard all that exceed a certain threshold. Here, we use the Levenshtein distance [62] as implemented by edlib[21]. Finally, we add the remaining entities as candidates to the subject cell in question.

[18]https://query.wikidata.org/
[19]https://www.wikidata.org/w/api.php
[20]To speed up the process, we use a many-to-many implementation for calculation [59].
[21]https://pypi.org/project/edlib/

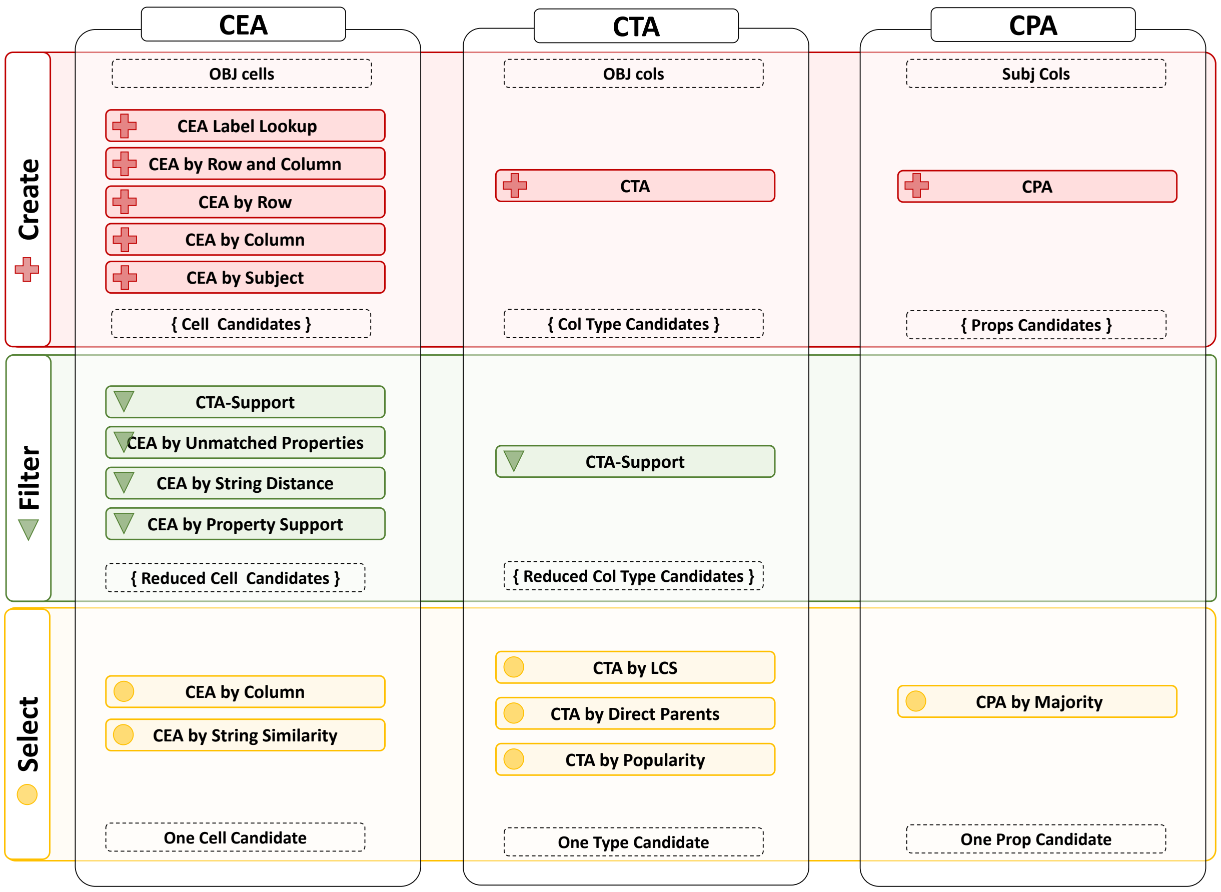


Fig. 6. Pool of building blocks and their assignment to CFS-stage and task. *Create* is indicated in red with a plus icon, *Filter* is represented in green and a triangle sign and *Select* is shown in yellow with a circle symbol (from [13]).

CEA by row and **CEA by column**: These two approaches work in a similar fashion as *CEA by row and column*, but drop one of its preconditions respectively. In *CEA by row*, a candidate does not have to be an instance of the current column's CTA-candidates. On the other hand, we apply *CEA by column*, when there are no other cells of datatype `OBJECT` in the same row, or those cells have no candidates either.

CEA by subject: This approach is the inverse to *CEA by row*. Assuming that the subject cell of a row has a set of candidates but any other cell of datatype `OBJECT` does not, it will fetch all entities from the Knowledge Base (KB) that are directly connected to a subject cell candidate. We filter the resulting entities again by their string-distance to the initial cell value before adding them as candidates.

CTA: To find candidates for each column of datatype `OBJECT`, we first retrieve the types for each cell individually based on its current list of CEA-candidates. In DBpedia, we rely on `rdf:type` property to represent the types. In Wikidata, a type denotes a combination of *instanceOf* (`wdt:P31`) and *subclassOf* (`wdt:P279`) relations. The column's type candidates are then given by the set of types appearing for at least one cell in this column. In addition, we configured three settings of CTA module to Wikidata specifically as follows:

– *P31* includes only direct parents using *instance of* (`wdt:P31`). This strategy does not include any higher levels of classes.
– *2Hops* includes "P31" with one additional parent (higher level) via *subclass of* (`wdt:P279`).
– *Multi Hops* creates a more general tree of parents following *subclass of* (`wdt:P279`) relations.

CPA: Candidates for column pairs' relations are generated by first retrieving candidates for the connections between cells of each row. We assume that there is a single subject column; thus, all other cells have to be connected in some way to the cell of that column.

First, we retrieve all properties for a subject cell's candidates, including both literal and object properties. Second, we try to match individual properties to the values of other cells in the same row. If we have found a match, we add the respective property as a candidate for the corresponding cell pair. Object properties are rather easy to match. Here, we rely on previously generated CEA candidates of the respective target and merely compare those with the object properties retrieved.

On the other hand, literal properties require more care. For them, we only consider matches to a cell whose datatype has been determined as either `DATE`, `STRING`, or `NUMBER` in the preprocessing. If we can not establish an exact match for a cell's value, we resort to fuzzy matching strategies depending on the corresponding datatype. For `DATE`-properties (RDF-type: `dateTime`), we try parsing the cell value using different date format patterns. If the parsing succeeds and both dates share the same day, month, and year, we consider this a match. We omit time and timezones for this comparison. In case of `STRING`-properties (RDF-type: `string` and `langString`), we extract words from the given value and the retrieved Wikidata label. Then, we count how many words are shared between the two string values. We consider a match if the overlap is above a certain threshold. For `NUMBER`-properties (RDF-type: `decimal`) we tolerate a 10% deviation to still be considered a match according to Equation 1 where *cell_value* is the table cell value and *property_value* is the retrieved property label.

$$Match = \begin{cases} true, & \text{if } |1 - \frac{cell\_value}{property\_value}| < 0.1 \\ false, & \text{otherwise} \end{cases} \tag{1}$$

Once candidates for each pair of cells are determined, we aggregate them to retrieve candidates on the column-level. This initial generation corresponds to the union of all candidates found on the row-level for cells within the respective columns.

4.7. Filter

Once we generated candidates for a particular task's solution, we apply filter-functions to sort out highly improbable candidates. For create-functions depending on previously generated candidates, this can substantially reduce the queries required and overall running time.

CTA-support: This filter works separately on each column of datatype `OBJECT`. First, it calculates the support of all current CTA candidates concerning the cells of a column. A cell supports a given CTA candidate if any of its current candidates has the corresponding type[22]. This filter neglects all cells that are either empty or have no CEA-candidates at the moment. Next, we remove all CTA-candidates from the respective column that do not have a support by at least 50% of the cells in this column. Finally, we remove all CEA-candidates from the corresponding cells, which have no types in the remaining CTA-candidates.

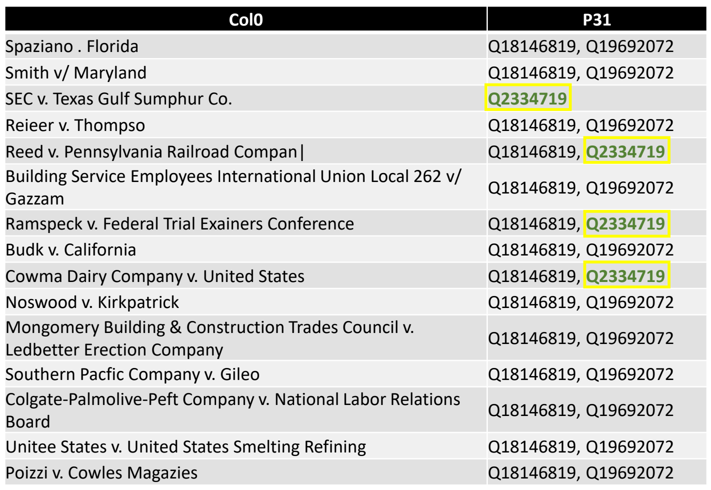
CEA by unmatched properties: After generating the properties for all cells on a row-level, some CEA-candidates will have no connection to any other cell in the same row. This filter removes these candidates, leaving only those that have a connection to at least one cell in their respective row. It applies to all cells of datatype `OBJECT`.

CEA by property support: This filter applies only to subject cells. We compute the support of a candidate as the number of cells in the same column it can be connected to[23]. We determine the maximum support for each of a cell's candidates and remove all those with lower support.

CEA by string distance: Some create-functions generate a relatively large number of CEA-candidates. This filter reduces that number by removing candidates whose label is too distant from the initial cell value. We rely on a

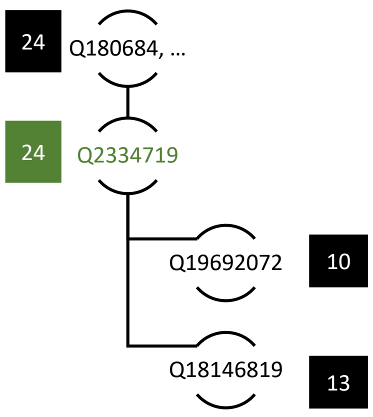
[22]Kindly refer to the definition of "type" in this context as given in the generation of CTA-candidates before.

[23]For non-subject cells, this value can only be 0 or 1, depending on whether they could be matched to the respective subject cell. This case is already covered in a different filter and thus excluded here.



Col0	P31
Spaziano . Florida	Q18146819, Q19692072
Smith v/ Maryland	Q18146819, Q19692072
SEC v. Texas Gulf Sumphur Co.	Q2334719
Reieer v. Thompso	Q18146819, Q19692072
Reed v. Pennsylvania Railroad Compan|	Q18146819, Q2334719
Building Service Employees International Union Local 262 v/ Gazzam	Q18146819, Q19692072
Ramspeck v. Federal Trial Exainers Conference	Q18146819, Q2334719
Budk v. California	Q18146819, Q19692072
Cowma Dairy Company v. United States	Q18146819, Q2334719
Noswood v. Kirkpatrick	Q18146819, Q19692072
Mongomery Building & Construction Trades Council v. Ledbetter Erection Company	Q18146819, Q19692072
Southern Pacfic Company v. Gileo	Q18146819, Q19692072
Colgate-Palmolive-Peft Company v. National Labor Relations Board	Q18146819, Q19692072
Unitee States v. United States Smelting Refining	Q18146819, Q19692072
Poizzi v. Cowles Magazies	Q18146819, Q19692072

(a) Cell values and corresponding types.



(b) Hierarchy of retrieved types.

Fig. 7. Example for CTA selection by LCS, the selected type is highlighted in green (from [13]).

normalized version of the Levenshtein distance [62], which uses the length of the involved strings as a normalizing factor. To keep any valid candidate, we resort to a rather conservative threshold, thus retain all candidates with a value of at least 0.5 for any of its labels.

4.8. *Select*

The target of all tasks (CEA, CTA, and CPA) is to select the most suitable solution per each task. Hence at some point, we have to select a single entry from the remaining candidates. As some of the methods below cannot distinguish between the candidates in a certain situation, we apply them in sequence until we find a solution. If only one candidate is left after the previous filter steps, we pick it for an obvious reason.

CEA by string similarity: For the remaining candidates, we calculate the string distance to the original cell value using the Levenshtein distance [62]. If there are multiple candidates with the same distance, we break those ties using the "popularity" of the respective candidates. We define popularity as the number of triples the respective candidate appears in them. The intuition is that if there was no other way to distinguish among the candidates in previous filter-steps, using the popularity results in the highest probability of selecting the correct candidate.

CEA by column: Sometimes filter-functions accidentally remove the correct solution from consideration. As those cases are quite rare, this affects only a small number of cells in a column. Further, the value of non-subject cells is often not unique throughout their column. In case no candidate is left for a cell, this function looks for occurrences of the same value in other cells of the same column. If we find a match, we apply their solution to the current cell.

CTA by Least Common Subsumer (LCS): The candidates for the CTA task do not stand in isolation but are part of a class hierarchy. Given an example of *Court Cases* in R3, cell types are shown in Figure 7a. We first expand this hierarchy for all remaining CTA-candidates, as shown in Figure 7b. Next, we compute the support for all candidates similar to the respective filtering-function. We remove all candidates with support less than the maximum. We choose the one with the longest path from the root node of the hierarchy as a solution from the remaining candidates.

CTA by Direct Parents: This function selects the CTA-solution by a majority vote. It will fetch the type for all remaining CEA-candidates of a column and then select the one appearing most often. In contrast to the previous definition of type, it only considers the direct connections of an entity, i.e. *instanceOf* (`wdt:P31`) and *subclassOf* (`wdt:P279`) but not their combination (`wdt:P31/wdt:P279`).

CTA by Popularity: In case the other methods failed to produce a result due to ties, this function will break those ties by using the candidates' popularity. Again, this is given by the number of corresponding triples the candidate appears in the entirety of the KB. As there is no semantic justification for this selection method, it is only used as a matter of last resort.

CPA by Majority Vote: We compute the support for a given CPA-candidate. Here, this refers to the number of remaining cell-candidate-pairs that use the respective property. We subsequently select the candidate with the highest support as a solution.

4.9. Default pipeline

Figure 8 shows the details for the default configuration of our pipeline, `pipeline_full`. The current block order results from experimentation using the available input tables as a source. After each run, we scanned the results for tables lacking some mappings and investigated causes and possible solutions. We aggregate the individual building blocks into groups for the sake of brevity in the following description.

Group ① forms the core of our pipeline and is responsible for most of the mappings. Based on the *CEA Label Lookup*, it generates candidates for all three tasks and removes only the most unlikely candidates from further consideration.

Group ② represents a first attempt to find missing CEA-mappings based on the context of a row and a column. Be kindly reminded that all create blocks only work on cells that currently have no candidates attached. So both blocks here apply to cells with no mappings from Group ①. *CEA by Row and Column* is put before *CEA by Row*, as it relies on a narrower context and thus will provide more accurate results. However, it might fail, e.g., when the current CTA-candidates do not include the proper solution. In such cases, *CEA by Row* will loosen the requirements to compensate. If either of these attempts provided new candidates, we re-execute the creation of CTA and CPA candidates afterwards in Group ③.

Group ④ is our first attempt at selecting solutions. After another filtering step on CEA-candidates using the row context, we continue to select high-confidence solutions. As hinted before, this might fail to produce proper mappings for a fraction of CEA-targets. Groups ⑤, and ⑦ try to fill in the gaps for at least some of them. If new candidates are found, groups ⑥ and ⑧ will filter and select from them.

Finally, Group ⑨ represents our means of last resort. Again, they only apply to targets for which we could not generate a solution before. Here, we assume that not only parts of the context are wrong but doubt every part of the context. The used blocks will reconsider all candidates discarded in the previous steps and attempt to find the best solution among them.

4.10. Other pipelines

Since 2021 and based on the original JenTab pipeline, `pipeline_full`, we derived multiple variations and evaluated them subsequently. This provided a better insight into the impact of individual components on the execution time and the quality of results. Most of these variations share the same initial phase to create candidates for CEA, CTA, and CPA if required.

`pipeline_essential` This pipeline reduces `pipeline_full` to its core components. In particular, each step runs only once, excluding any re-execution in the process. It became necessary initially as some tables proved too demanding when executed using `pipeline_full`.

`pipeline_no_cpa` Compared to `pipeline_full`, this pipeline removes any CPA-related components. In particular, filter operations involving relations among cells as well as selecting CPA-solutions are omitted in this pipeline. It was applied in tasks that featured only CEA and CTA targets and omitted any CPA ones.

`pipeline_numeric` This pipeline is specifically geared towards tables that feature a single object (the subject) column and one or more non-object (primarily numeric) column(s). After creating initial candidates for all tasks it filters them with the least frequent properties to determine the most likely CEA and thus indirectly CTA-candidates. The latter is then used while applying `CEA by Column` that adds new candidates to unmatched cells based on all instances of the identified types. After subsequent additional filter steps, the default selection process is used to determine the final solutions.

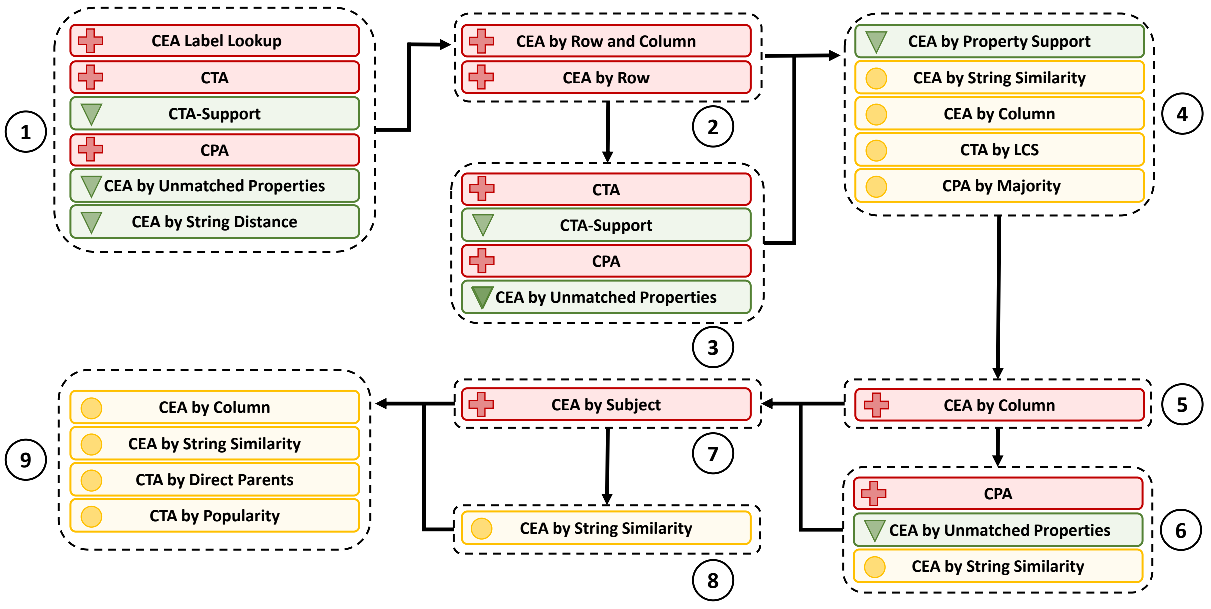


Fig. 8. `pipeline_full` as an arrangement of CFS building blocks (from [13]).

`pipeline_conditional` As `pipeline_numeric` only applies to a subset of tables, this pipeline uses a two-step approach: In a first step, `pipeline_numeric` is applied to all tables meeting the respective preconditions. If these conditions are not met, or the returned result covers less than 80% of targets, the table is again processed using `pipeline_full`.

`pipeline_header` This pipeline is based on `pipeline_no_cpa` which contains all modules from the default configuration except the CPA creation, filteration, and selection parts. However, the new pipeline overrides the CTA with header candidates. Indeed, such method suits datasets that contain meaningful header.

5. Evaluation

In the years from 2020 to 2022, we conducted a series of evaluations using the SemTab challenge for large-scale general domain datasets and using a biodiversity-specific dataset. In 2023, we extended our evaluation by adopting an automatically generated large-scale food-specific dataset as well. In the following, we describe the evaluation corpora, preprocessing, and generic lookup strategy's effectiveness. In addition, we evaluate our CEA creation strategies and compare the accuracy scores of JenTab compared to the other systems tackling the same tasks. Finally, we discuss the runtime of JenTab on the evaluation benchmark and give an overview of its performance over the years of development.

5.1. Benchmarks

Before we evaluate JenTab in various dimensions, e.g., accuracy scores, we give an overview of the benchmarks used in the evaluation.

– **SemTab 2020** [7] is an Automatically Generated (AG) benchmark that contains over 130,000 tables from Wikidata and is divided into four rounds. The tasks are to annotate these tables using Wikidata.

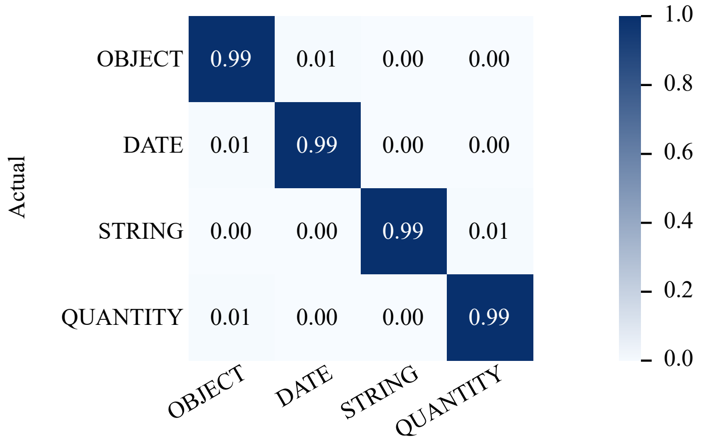


Fig. 9. Confusion matrix for type prediction. A 4-label classification task (from [15]).

- **HardTables** [8, 25] is an AG dataset with a first release during the second round of SemTab 2021 with 7,207 tables. The authors filtered it to include only the hardest cases in SemTab 2022 and released it again with 3,691 and 4,659 tables in the first and second round of that year's challenge. The target KG concerning its tasks is Wikidata.
- **Tough Tables (2T)** [9] is a dataset consisting of 180 tables annotated from both Wikidata and DBpedia. It has been used during SemTab 2021 and 2022. This dataset is known for its very high level of entity ambiguity and its massive amount of artificially added noise to table cells.
- **BioTables** [5, 63] is a biomedical domain-related dataset for STA. It is also an AG dataset comprising 110 tables. It was released and used during SemTab 2021. This benchmark has a unique characteristic that features a high average number of columns, 2500.
- **BiodivTab** [11, 12] is a real-world and manually annotated benchmark from biodiversity research. It consists of 50 tables. This benchmark is released and used for the first time during SemTab 2021, where the target KG is Wikidata. It is released and used again during the third round of SemTab 2022, where the target KG was changed to DBpedia. The authors of this paper have constructed this benchmarks in its two versions, however, details of its construction are beyond this paper.
- **tFood** [64] is an AG dataset derived from the food domain and has two types of tables: 1) horizontal, relational tables each representing a collection of entities, and 2) entity tables, each representing a single entity. In this paper, we use the horizontal tables fold since it provides ground truth mappings to Wikidata for CEA, CTA, and CPA tasks.

5.2. Datatype Prediction Assessment

For evaluation, we start with assessing the preprocessing during the "Type Prediction" step. This step is responsible for determining a column's datatype (cf. Subsection 4.2). Figure 9 shows is confusion matrix. It shows a 99% prediction accuracy score. We used the SemTab 2020 benchmark for evaluating this step. We used the ground truth for CEA and CPA tasks to query Wikidata for their types. Values represent the actual datatypes while predicted values are by JenTab.

5.3. Generic Lookup Coverage

Spelling mistakes are a crucial problem tackled using the "Generic Strategy". We create this lookup before the actual pipeline run due to the resources required. First, we extract the unique values from all tables of a dataset and match them against labels (and aliases) of the target KG using an optimized Jaro-Winkler Similarity implementation

Table 2
Generic Strategy: Unique labels and ratio of resolved labels per round.[24]

Year	Rounds	Dataset	Target	Unique Labels	Matched	Matched
2020	R1	SemTab2020	Wikidata	252,329	249,806	99.0%
2020	R2	SemTab2020	Wikidata	132,948	131,486	98.9%
2020	R3	SemTab2020	Wikidata	361,313	357,700	99.0%
2020	R4	SemTab2020	Wikidata	533,015	515,959	96.8%
2021	R1	2T	Wikidata	69,980	62,908	89.89%
2021	R1	2T	DBpedia	66,340	59,168	89.19%
2021	R2	HardTables	Wikidata	249,625	249,025	99.76%
2021	R3	HardTables	Wikidata	47,809	46,865	98.03%
2022	R1	HardTables	Wikidata	19,107	18,928	99%
2022	R2	HardTables & 2T	Wikidata	74,177	67,986	91.6%
2022	R2	2T	DBpedia	65,223	58,235	89.3%

based on [59] and a minimum similarity of 0.9. This lookup represents our primary and only source to fix spelling mistakes since 2021. The effectiveness of the lookup is illustrated in Table 2 over the three years. For the SemTab 2020 benchmark, almost 99% of unique labels were resolved in the first three rounds. However, this is reduced to $\sim$ 97% in the last round.

In 2021, more than 89% of unique values could be matched including a 99.76% success rate for R2's HardTables. However, it is significantly lower for the 2T dataset. This dataset has a massive amount of spelling mistakes that the selected threshold could not solve.

In 2022, for the synthetic dataset, HardTables, the matching percentage stayed high with up to 99% in R1 while 2T results remained as low as in the previous year reaching only around 89% in R2 for DBpedia. The low matching results guided us to develop a more sophisticated cleaning step before the actual run, as discussed in Subsection 4.3. We have not created a generic lookup for BiodivTab in 2021 and 2022 since our approach here relies on dictionaries for taxons and chemical elements — the prevalent type of cell values in this dataset.

5.4. Audit Results

We have implemented five strategies for creating initial CEA candidates, including Generic Lookup, Full Cell, Tokens, All Tokens, Selective, and Autocorrect. We investigated their use using various benchmarks.

CEA Creation Figure 10 shows how much each strategy across general domain benchmarks. Generic Lookup proves its strength in solving spelling mistakes and, thus, is the most used strategy for all benchmarks. We recently disabled "Autocorrect" and "All Tokens" strategies due to their high computational requirements in the recent benchmarks (BioTables and HardTables). This underlines the need for various strategies to capture a wide range of useful information inside each cell. The shown distribution also reflects our chosen order of methods. For example, "Generic Strategy" has the highest priority and is used most of the time. On the other hand, "Autocorrect" has the lowest priority and is used as a means of last resort. We handle the BiodivTab benchmark differently, i.e., we build a dictionary for taxons and chemical compounds and use it as a customized lookup service. Figure 11 shows the distribution of selected methods to create CEA candidates in both editions of BiodivTab, Wikidata and DBpedia.

CEA Selection Figure 12 outlines the use of each strategy. The dominant select approach is "String Similarity", used by 38% more than the "Column Similarity" in all benchmarks except HardTables where the "String Similarity" managed to solve all cases. For BiodivTab, Figure 13 depicts the methods distribution for the CEA selection strategies. For both versions, i.e., Wikidata and DBpedia, "String Similarity" solved more than double the amount compared to "Column Similarity".

[24]We deactivated the generic lookup option from JenTab during the processing of the tFood dataset, so no results are available here.

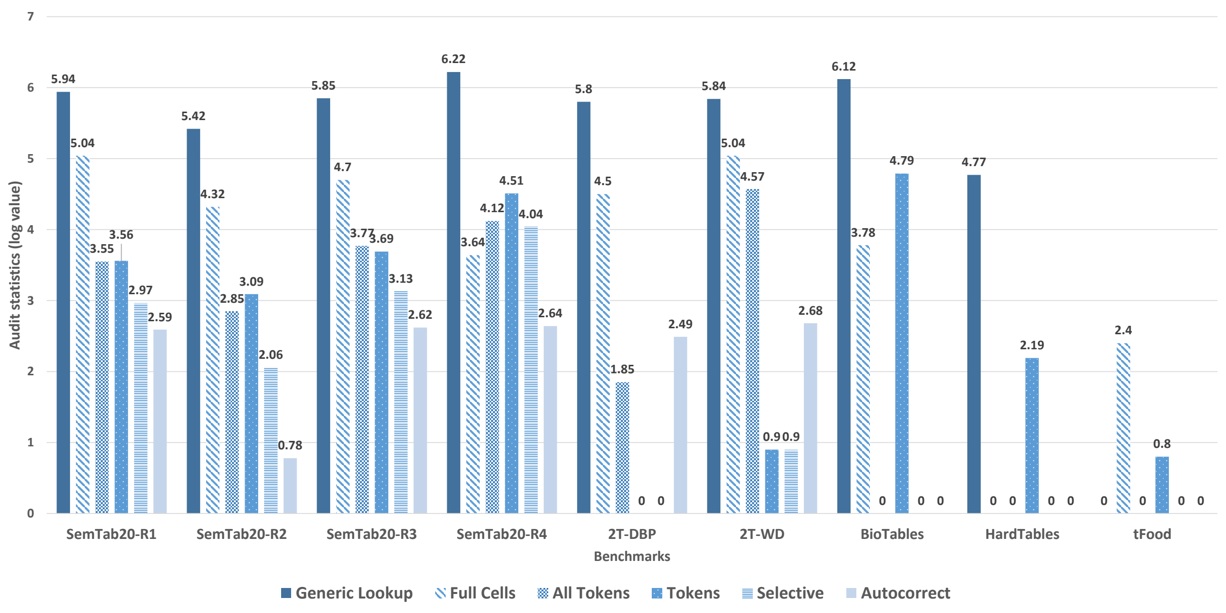


Fig. 10. Audit statistics for CEA creation. y-axis is the *log* scale of the solved cells.

CTA Selection Figure 14 describes the distribution of CTA selection methods with LCS being the dominant strategy. It shows continuous success in solving the task for all benchmarks, especially BioTables and HardTable. Both benchmarks did not need any backup solutions to select the CTA candidate at all. For BiodivTab, Figure 15 highlights the distribution of CTA selection strategies. The dominant method in the Wikidata version is the LCS where we activated the "2Hops" mode. However, the primary method in the DBpedia version is "Majority Vote". We observe that LCS successfully finds more solutions than the other relying less on backup strategies or tiebreakers. The same prioritization from the CEA selection is also applied in CTA. The dominant method, e.g., LCS or "Majority Vote", is invoked more frequently due to its higher priority during execution. Other backup strategies try to solve the remaining columns if previous methods failed.

5.5. *Accuracy Scores*

In the following, we compare JenTab to existing state-of-the-art systems tackling STA tasks to demonstrate its effectiveness[25].

SemTab 2020 (cf. Table 3). This represents the first participation of JenTab in SemTab. We did not participate in R1's CPA task since the corresponding functions were not yet implemented. However, we continued development after the challenge and report a complete list of scores here. In this benchmark, JenTab demonstrated high scores across all rounds, especially in the CPA task. JenTab is achieved third place for the given benchmarks and without complex dependencies or a KG dump.

HardTables (cf. Table 4). JenTab kept its high performance and rank during 2021's edition of this benchmark. However, during the second round in 2022, it achieved lower scores. A reason could be that HardTables during this challenge's edition is way more complicated compared to previous instances. A higher level of ambiguity has been introduced causing all top participants, not only JenTab, to achieve lower scores.

Tough Tables (2T) (cf. Table 5). It first appeared during R4 of SemTab 2020 and used Wikidata as the target KG. In 2021, it appeared again with an updated ground truth from Wikidata. In addition, the authors added DBpedia as an additional target KG. The scores by JenTab and others systems are relatively lower than those on AG datasets. The main reason for these low scores is the high level of ambiguity that could confuse even a human [9] of the table

[25]SemTab challenge results are used for comparison.

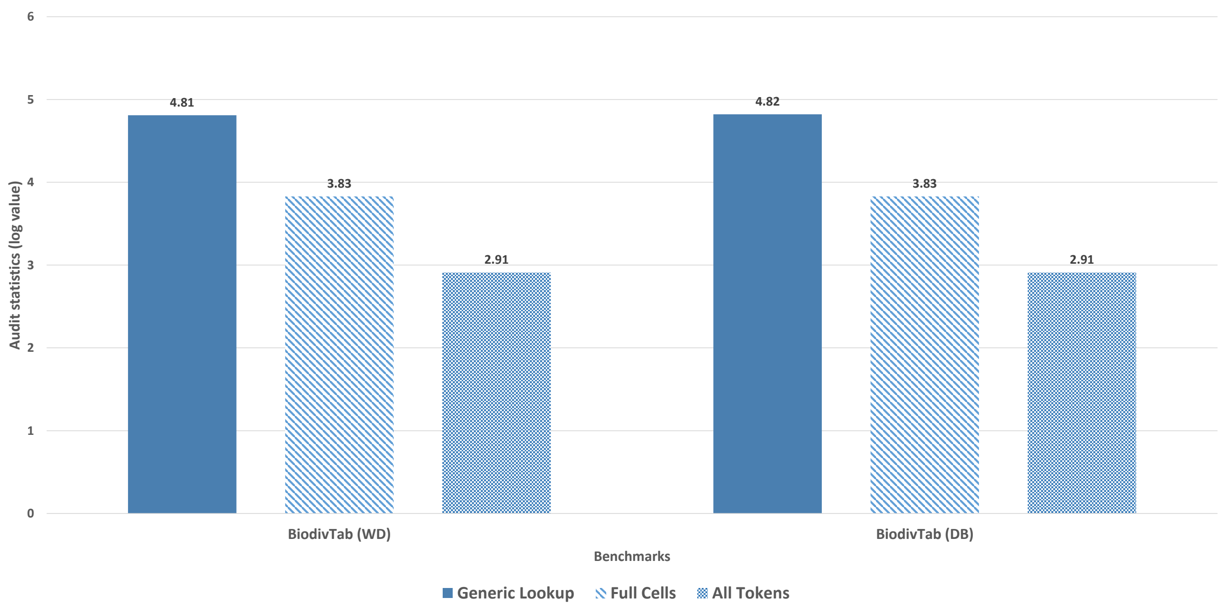


Fig. 11. Audit statistics for CEA creation (**BiodivTab**). y-axis is the *log* scale of the solved cells.

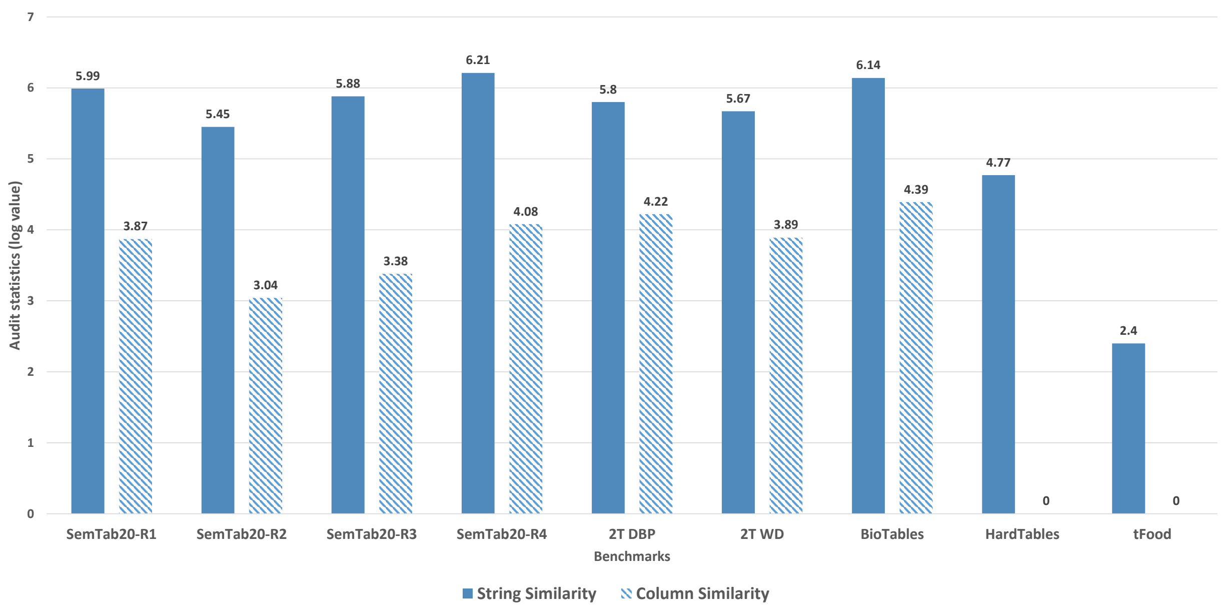


Fig. 12. Audit statistics for CEA selection. y-axis is the *log* scale of the solved cells.

cells. Another reason is the excessive amount of artificial spelling mistakes added to the entries. In 2020, JenTab was still more sensitive to the latter issue than other systems. We continuously developed JenTab to tackle the problem of misspellings. For CEA, over the years, JenTab gained much higher scores on this dataset from 37% to slightly above 80%, given Wikidata as a target KG.

BioTables (cf. Table 6). JenTab showed consistent scores and ranked among the top systems. This benchmark introduced new obstacles for all participants. Initially, we failed to run our `pipeline_full` due to several time-out issues. However, we developed our default pipeline and managed to execute it on this benchmark and reach competitive scores.

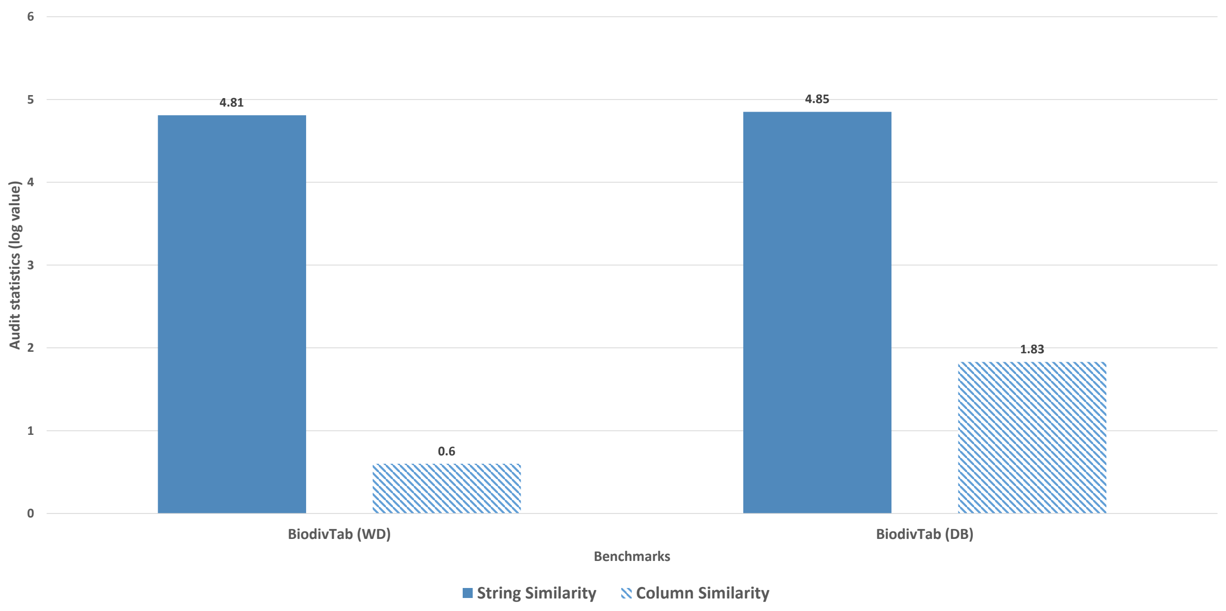


Fig. 13. Audit statistics for CEA selection (**BiodivTab**). y-axis is the *log* scale of the solved cells.

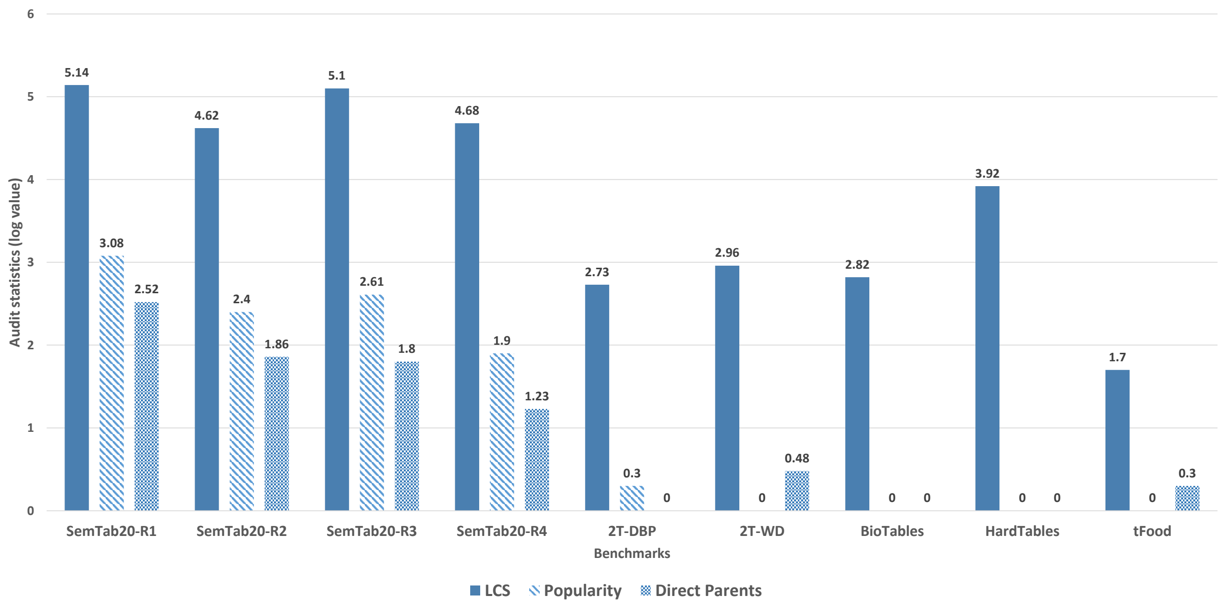


Fig. 14. Audit statistics for CTA selection. “2Hops” mode. y-axis is the *log* scale of the solved cells.

BiodivTab (cf. Table 7). JenTab, achieved the best score for the CEA task. However, the best CTA score was achieved by KEPLER [43]. We gained low scores of 60%/10% in 2021 and 55%/41% in 2022 (CEA / CTA). In contrast, for the synthetic dataset, HardTables 2021, DAGOBAH achieved the maximum F1-score 97.4%/99% (CEA / CTA). These low scores are due to the unique benchmark characteristics differing from the traditional AG and general-domain benchmarks.

tFood (cf. Table 8). We asked the authors of DAGOBAH [66] and s-elbat [46] to solve this dataset beyond the scope of SemTab 2023 to compare all systems. We gained our lowest scores on this benchmark as the case for other systems. We need to investigate in depth concrete examples of this dataset to determine the root cause. Possible

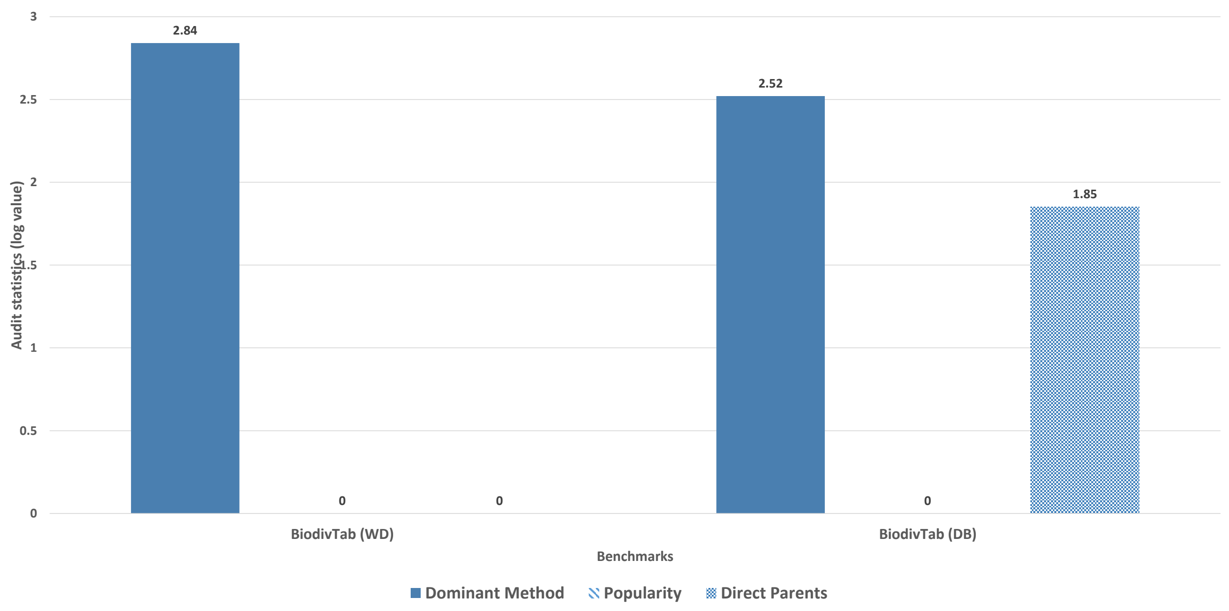


Fig. 15. Audit statistics for CTA selection (**BiodvTab**). “2Hops” mode. y-axis is the *log* scale of the solved cells.

reasons are either the dataset is too hard to solve for STA systems, i.e., lacking enough context to disambiguate table cells and columns, or the dataset contains errors and lacks data quality.

General Overview We give an overview of JenTab scores on both AG (cf. Figure 16), and 2T (cf. Figure 17) datasets between 2020 to 2022. For the AG datasets, the average F1-score for the three tasks – CEA, CTA, and CPA – are 0.91, 0.93, and 0.97 respectively. For the 2T dataset, the average F1-score for both CEA and CTA tasks is 0.54. This shows JenTab's sensitivity for high levels of artificial noise. In 2022, CEA results have significantly improved due to a sophisticated cleaning module for such datasets.

5.6. Runtime Performance

During JenTab's development, we used regular laptops for execution, e.g., two core i7 machines, with 16GB and 8GB of memory. Starting from 2021, we hosted added a virtual machine with 256 cores and 64GB of memory. We set up three different experiments to test the effect of CTA selection strategies for the SemTab 2020 benchmark. Besides low scores, the Multi-Hops strategy is also computationally expensive. Table 9 shows the processing time for all four rounds with the number of used runners for each mode setting of the CTA task. Close inspection revealed that the execution time is largely dominated by the responses of Wikidata servers and thus beyond our control. The execution was time-scoped, i.e., an upper limit for the time per table was set. This allowed the system to converge faster compared to the initial implementation of JenTab in 2020, with, e.g., R4 showing a more than 50% reduction in time. Intermediate results are cached across rounds, saving time and lowering the number of requests to external services. Our modular approach allows us to scale the number of runners based on available resources and speed up the overall process.

We excluded the “Multi-Hops” from further use and limit ourselves to “P31” and “2Hops”. Table 10 shows the runtime of JenTab during its participation in 2021 and 2022. This table shows the configuration yielding the best scores. For 2021, the setup of CTA is “2Hops”, but for 2022 benchmarks, we selected “P31”. The 2020 benchmark is the average of the corresponding setting's four rounds from the previous table. We summarize JenTab's performance in Figure 18. We point out the continuous enhancements developed to reduce the required resources across its years of development.

Table 3

SemTab 2020 top participants' scores (**AG-datasets**). F1 - F1 Score, Pr - Precision, R - Recall, AF1, APr, and AR - Approximate version of F1 Score, Precision, and Recall respectively.

		CEA		CTA		CPA	
Round	System	F1	Pr	AF1	APr	F1	Pr
R1	MTab [27]	0.987	0.988	0.885	0.884	0.971	0.991
R1	LinkingPark [39]	0.987	0.988	0.926	0.926	0.967	0.978
R1	SSL [40]	0.936	0.936	0.861	0.860	0.943	0.943
R1	bbw [31]	NA	NA	NA	NA	NA	NA
R1	DAGOBAH [29]	0.922	0.944	0.834	0.854	0.914	0.962
R1	JenTab	0.968	0.969	0.962	0.965	0.984	0.988
R2	MTab [27]	0.995	0.995	0.984	0.984	0.997	0.997
R2	LinkingPark [39]	0.993	0.993	0.984	0.985	0.993	0.994
R2	SSL [40]	0.961	0.961	0.966	0.966	0.973	0.973
R3	bbw [31]	0.892	0.960	0.914	0.929	0.991	0.992
R2	DAGOBAH [29]	0.993	0.993	0.983	0.983	0.992	0.994
R2	JenTab	0.975	0.975	0.965	0.965	0.984	0.984
R3	MTab [27]	0.991	0.992	0.976	0.976	0.995	0.995
R3	LinkingPark [39]	0.986	0.986	0.978	0.979	0.985	0.988
R3	SSL [40]	0.906	0.906	0.913	0.913	0.815	0.815
R3	bbw [31]	0.954	0.974	0.960	0.966	0.949	0.957
R3	DAGOBAH [29]	0.985	0.985	0.974	0.974	0.993	0.994
R3	JenTab	0.967	0.967	0.955	0.959	0.981	0.987
R4	MTab [27]	0.993	0.993	0.981	0.982	0.997	0.997
R4	LinkingPark [39]	0.985	0.985	0.953	0.953	0.985	0.985
R4	SSL [40]	0.833	0.833	0.946	0.946	0.924	0.924
R4	bbw [31]	0.978	0.984	0.980	0.980	0.995	0.995
R4	DAGOBAH [29]	0.984	0.985	0.972	0.972	0.995	0.995
R4	JenTab	0.974	0.974	0.945	0.93	0.993	0.994

6. Discussion and Limitations

JenTab is easily extensible and its components are highly configurable. For example, it provides an easy way to change the target KG and enable various settings for solving the STA tasks. Currently, it contains six pipelines that we developed based on various dataset characteristics. In addition, it supports annotations from both Wikidata and DBpedia. JenTab adopts a distributed approach that leverages the capabilities of the individual client machines. This allows for a scalable framework that can process large-scale tabular data benchmarks. We tested JenTab on more than $156,000$ tables in the scope of the SemTab challenge. JenTab has a reasonable amount of open-source dependencies and is well-documented. In addition, it supports an easy-to-use execution via Docker containers, as seen in Listing 1.

Listing 1: JenTab docker commands

```
docker-compose -f docker-compose.manager.yml up
docker-compose -f docker-compose.yml up
docker run --network="host" runner
```

JenTab won the second prize in the “Usability track” by IBM research in 2021. In addition, JenTab is the only participant obtaining the “Artificate Availability Badge” by SemTab 2022. This badge is awarded for systems that

Table 4
JenTab & SOTA scores (**HardTables** 2021-2022). F1 - F1 Score, Pr - Precision, R - Recall, AF1, APr, and AR - Approximate version of F1 Score, Precision, and Recall respectively.

			CEA		CTA		CPA	
Year	Round	System	F1	Pr	AF1	APr	F1	Pr
2021	R2	MTab [28]	0.985	0.985	0.977	0.977	0.997	0.998
2021	R2	DAGOBAH [30]	0.975	0.975	0.976	0.976	0.996	0.996
2021	R2	Magic [41]	0.836	0.947	0.757	0.681	0.865	0.954
2021	R2	Kepler-aSI [43]	0.975	0.975	0.894	0.931	0.915	0.989
2021	R2	JenTab	0.966	0.967	0.914	0.917	0.996	0.997
2021	R3	MTab [28]	0.968	0.968	0.984	0.984	0.993	0.993
2021	R3	DAGOBAH [30]	0.974	0.974	0.990	0.990	0.991	0.995
2021	R3	Magic [41]	0.641	0.721	0.687	0.687	0.788	0.936
2021	R3	Kepler-aSI [43]	NA	NA	0.244	0.244	NA	NA
2021	R3	JenTab	0.940	0.940	0.942	0.942	0.992	0.992
2022	R1	KGCODE-Tab [65]	0.893	0.916	0.942	0.944	0.906	0.918
2022	R1	DAGOBAH [66]	0.954	0.955	0.975	0.975	0.984	0.99
2022	R1	s-elbat [46]	0.945	0.964	0.951	0.957	0.983	0.989
2022	R1	JenTab	0.945	0.946	0.938	0.940	0.975	0.986
2022	R2	KGCODE-Tab [65]	0.856	0.875	0.968	0.971	0.916	0.943
2022	R2	DAGOBAH [66]	0.904	0.905	0.96	0.96	0.931	0.97
2022	R2	s-elbat [46]	0.825	0.875	0.859	0.878	0.931	0.96
2022	R2	JenTab	0.751	0.758	0.836	0.881	0.872	0.921

Table 5
JenTab and existing systems scores (**2T** dataset). F1 - F1 Score, and Pr - Precision. AF1, and APr - Approximate version of F1 Score, Precision respectively.

		Wikidata				DBpedia			
		CEA		CTA		CEA		CTA	
Year	System	F1	Pr	AF1	APr	F1	Pr	AF1	APr
2020	MTab [28]	0.907	0.907	0.885	0.884	NA	NA	NA	NA
2020	LinkingPark [39]	0.810	0.811	0.926	0.926	NA	NA	NA	NA
2020	SSL [40]	0.198	0.198	0.861	0.860	NA	NA	NA	NA
2020	bbw [31]	0.863	0.927	NA	NA	NA	NA	NA	NA
2020	DAGOBAH [40]	0.412	0.749	0.834	0.854	NA	NA	NA	NA
2020	JenTab	0.374	0.541	0.574	0.626	NA	NA	NA	NA
2021	DAGOBAH [30]	0.923	0.923	0.832	0.832	0.945	0.946	0.422	0.424
2021	AMALGAM [42]	0.658	0.791	0.476	0.422	NA	NA	NA	NA
2021	Kepler-aSI [43]	0.194	0.760	0.464	0.481	0.110	0.644	0.027	0.133
2021	Magic [41]	NA	NA	NA	NA	0.184	0.506	0.159	0.628
2021	JenTab	0.457	0.520	0.697	0.697	0.607	0.669	0.460	0.468
2022	KGCODE-Tab [65]	0.905	0.913	0.543	0.546	0.827	0.830	0.480	0.484
2022	DAGOBAH [66]	0.945	0.946	0.409	0.409	NA	NA	NA	NA
2022	s-elbat [46]	0.937	0.938	0.366	0.366	0.789	0.808	0.373	0.375
2022	JenTab	0.802	0.807	0.346	0.356	0.572	0.792	0.234	0.290

Table 6

JenTab & SOTA scores (**BioTables** 2021). F1 - F1 Score, Pr - Precision. AF1, and APr - Approximate version of F1 Score, and Precision respectively.

			CEA		CTA		CPA	
Year	Round	System	F1	Pr	AF1	APr	F1	Pr
2021	R2	MTab [28]	0.985	0.985	0.977	0.977	0.997	0.998
2021	R2	DAGOBAH [30]	0.975	0.975	0.976	0.976	0.996	0.996
2021	R2	Magic [41]	0.836	0.947	0.757	0.681	0.865	0.954
2021	R2	Kepler-aSI [43]	0.975	0.975	0.894	0.931	0.915	0.989
2021	R2	JenTab	0.966	0.967	0.914	0.917	0.996	0.997

Table 7

JenTab & SOTA scores (**BiodivTab** 2021-2022) [5, 25]. F1 - F1 Score, Pr - Precision, AF1, and APr - Approximate version of F1 Score and, Precision respectively.

			CEA		CTA	
Year	Target	System	F1	Pr	AF1	AP1
2021	Wikidata	MTab [28]	0.522	0.527	0.123	0.282
2021	Wikidata	Magic [41]	0.142	0.192	0.10	0.253
2021	Wikidata	DAGOBAH [30]	0.496	0.497	0.381	0.382
2021	Wikidata	mantisTable [37]	0.264	0.785	0.061	0.076
2021	Wikidata	KEPLER [43]	NA	NA	0.593	0.595
2021	Wikidata	JenTab	0.602	0.611	0.107	0.107
2022	DBpedia	KGCODE-Tab [65]	0.910	0.910	0.870	0.870
2022	DBpedia	DAGOBAH [66]	NA	NA	0.620	0.620
2022	DBpedia	s-elbat [46]	NA	NA	0.06	0.06
2022	DBpedia	JenTab	0.550	0.610	0.420	0.412

Table 8

JenTab & SOTA scores (**tFood** 2023) . F1 - F1 Score, Pr - Precision, AF1, and APr - Approximate version of F1 Score and, Precision respectively.

	CEA		CTA		CPA	
System	F1	Pr	AF1	APr	F1	Pr
DAGOBAH [66]	0.012	0.069	0.305	0.422	0.013	0.176
s-elbat [46]	0.012	0.067	0.339	0.553	0.034	0.085
TSOTSA [48]	0.045	0.041	0.031	0.031	0.285	0.285
Kepler-aSI [45]	-	-	0.105	0.105	0	0
JenTab	0.01	0.052	0.320	0.424	0.031	0.176

Table 9

Execution time for different setups (SemTab2020 benchmark).

	R1		R2		R3		R4	
Mode	Days	Runners	Days	Runners	Days	Runners	Days	Runners
P31	0.5	4	2.5	4	1.5	6	2	4
2Hops	1	4	1.2	4	2	4	1.1	8
Multi Hops	1	4	1.5	4	2.5	6	3.5	6

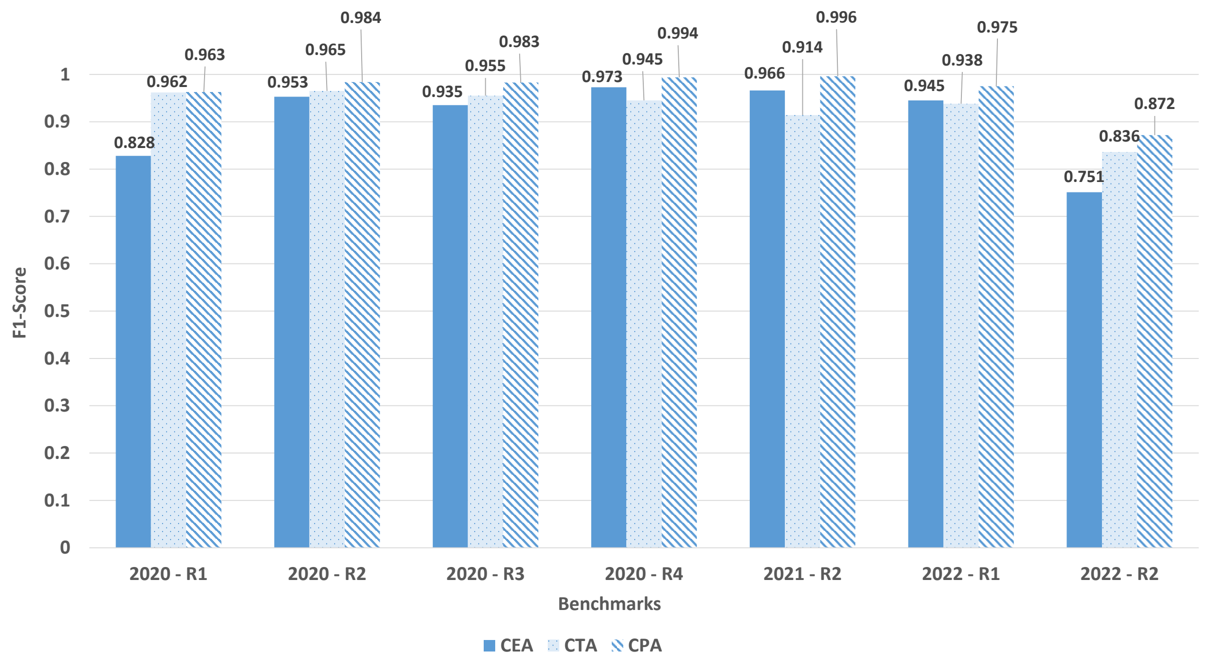


Fig. 16. JenTab F1-scores Automatically Generated (AG) datasets [2020-2022].

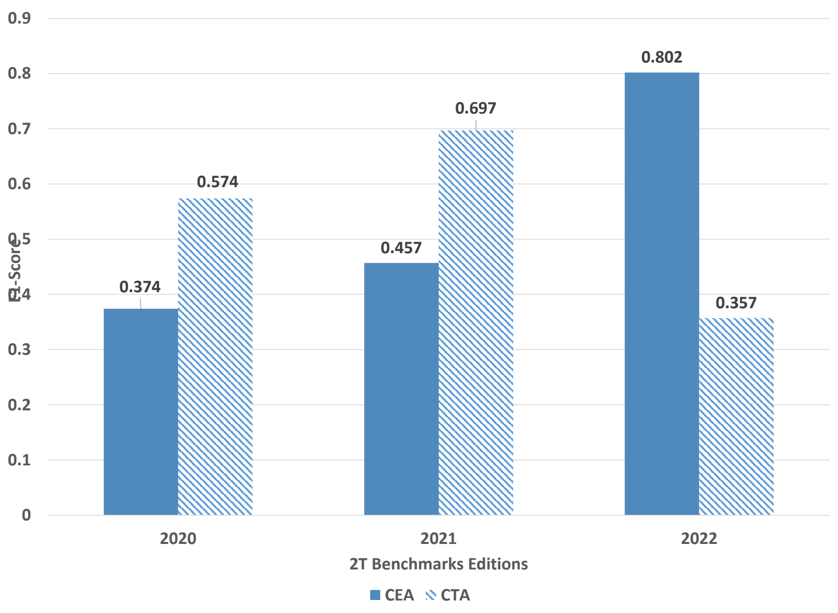


Fig. 17. JenTab F1-scores Tough Tables (2T) datasets [2020-2022].

are open-source data and code, well-documented, and have open-source and reasonable dependencies. The open-source data includes the pre-computed lookup and the system solutions. We released both of them to be publicly available to enable further analysis by future work. For example, Avgardro et al. [67] used the output of our system in a deeper quality check framework. JenTab demonstrated effectiveness in this work compared to other state of the art. Further, Chaves-Fraga and Dimou [68] used our artifacts to compare fully automatic systems versus the declarative mapping rules.

Table 10
Execution time for different setups (SemTab 2020-2022 benchmarks).

Mode	Year	Dataset	Target	Runner	Time
P31	2020	SemTab2020	Wikidata	4.5	1.5 Days
	2022	HardTables	Wikidata	3	11 Hours
	2022	2T	DBpedia	1	5 Hours
	2022	2T	Wikidata	1	3 Hours
	2022	BiodivTab	DBpedia	1	1 Hour
2 Hops	2020	SemTab2020	Wikidata	5	1.33 Days
	2021	HardTables	Wikidata	1	10.5 Hours
	2021	2T	DBpedia	1	1 Day
	2021	2T	Wikidata	2	11 Hours
	2021	BioTables	Wikidata	1	7 Hours
	2021	BiodivTab	Wikidata	1	1.5 Hours

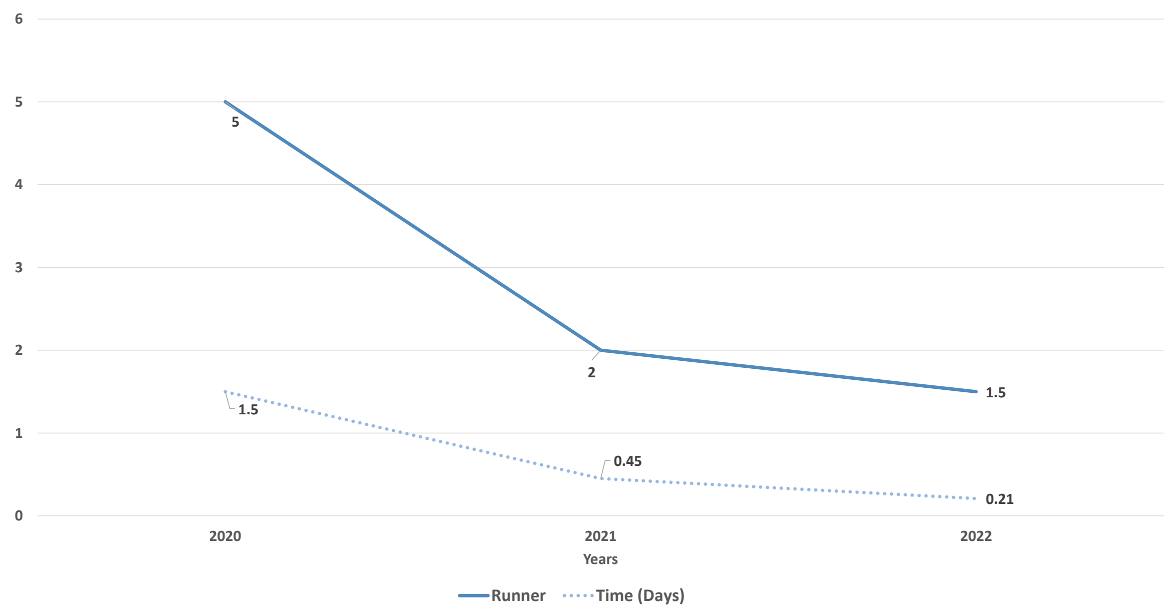
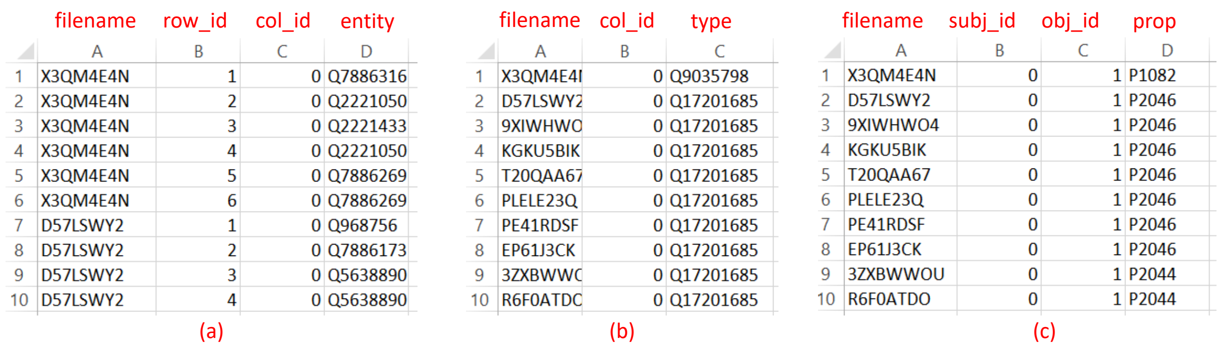


Fig. 18. Runners and runtime of JenTab [2020-2022].

JenTab provides an overview of the current processing state. Figure 20 depicts a screenshot of the centralized Manager node of JenTab. It shows how many tables are in progress, successfully completed, failed to complete, and the errors returned. It stores the results and can export them in the required format. The output format is shown in Figure 19 where Wikidata is the target KG. First, CEA results include the file/table name without extension, row id, column id (together they point to a specific cell), and the mapped entity from KG. Second, CTA results consist of file/table name, target column id, and the mapped semantic type or class from the KG. Finally, CPA output contains the file/table name, subject and object columns id, and their semantic property or relation from the KG. The Manager enables a debug feature by retrieving tables with no or incomplete annotations to help us investigate these hard cases more closely. Such features helped us identify each dataset's characteristics and facilitate error analysis. Moreover, the Manager contains audit statistics, i.e., the loads on individual system components (e.g., CEA Creation Module) making the system more transparent.

JenTab is a top performer system solving STA tasks. During its years of development, it achieved a high accuracy in general domain benchmarks. On average and given all the Automatically Generated (AG) datasets, it achieved



(a)

	filename	row_id	col_id	entity
	A	B	C	D
1	X3QM4E4N	1	0	Q7886316
2	X3QM4E4N	2	0	Q2221050
3	X3QM4E4N	3	0	Q2221433
4	X3QM4E4N	4	0	Q2221050
5	X3QM4E4N	5	0	Q7886269
6	X3QM4E4N	6	0	Q7886269
7	D57LSWY2	1	0	Q968756
8	D57LSWY2	2	0	Q7886173
9	D57LSWY2	3	0	Q5638890
10	D57LSWY2	4	0	Q5638890

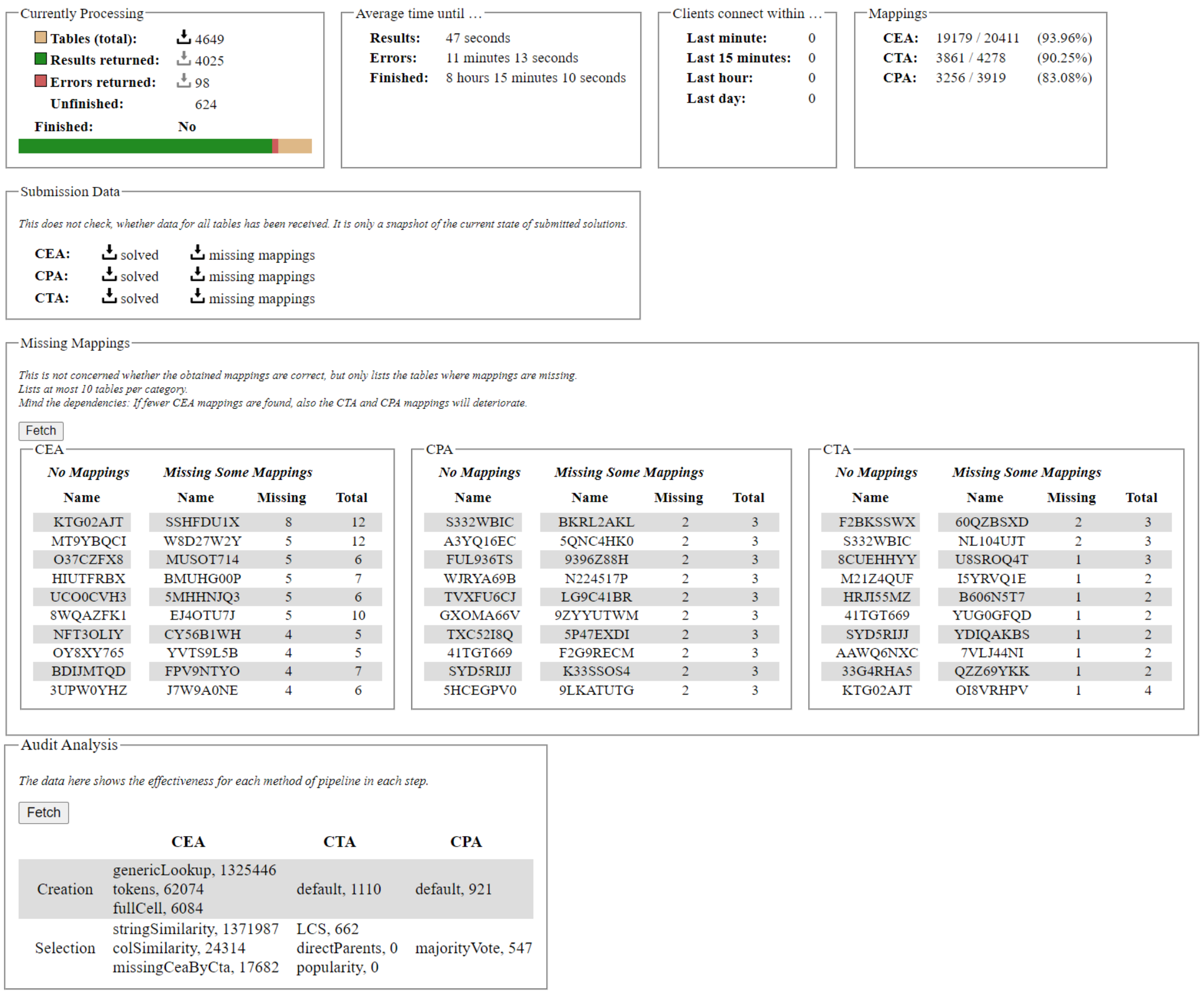
(b)

	filename	col_id	type
	A	B	C
1	X3QM4E4I	0	Q9035798
2	D57LSWY2	0	Q17201685
3	9XIWHWO	0	Q17201685
4	KGKU5BIK	0	Q17201685
5	T20QAA67	0	Q17201685
6	PLELE23Q	0	Q17201685
7	PE41RDSF	0	Q17201685
8	EP61J3CK	0	Q17201685
9	3ZXBWWC	0	Q17201685
10	R6F0ATDC	0	Q17201685

(c)

	filename	subj_id	obj_id	prop
	A	B	C	D
1	X3QM4E4N	0	1	P1082
2	D57LSWY2	0	1	P2046
3	9XIWHWO4	0	1	P2046
4	KGKU5BIK	0	1	P2046
5	T20QAA67	0	1	P2046
6	PLELE23Q	0	1	P2046
7	PE41RDSF	0	1	P2046
8	EP61J3CK	0	1	P2046
9	3ZXBWWOU	0	1	P2044
10	R6F0ATDO	0	1	P2044

Fig. 19. JenTab: Output snippet. (a) CEA, (b) CTA, and (c) CPA.



Currently Processing

▨	Tables (total):	4649
🟩	Results returned:	4025
🟥	Errors returned:	98
	Unfinished:	624
	Finished:	No

Average time until …

Results:	47 seconds
Errors:	11 minutes 13 seconds
Finished:	8 hours 15 minutes 10 seconds

Clients connect within …

Last minute:	0
Last 15 minutes:	0
Last hour:	0
Last day:	0

Mappings

CEA:	19179 / 20411	(93.96%)
CTA:	3861 / 4278	(90.25%)
CPA:	3256 / 3919	(83.08%)

Submission Data

This does not check, whether data for all tables has been received. It is only a snapshot of the current state of submitted solutions.

CEA:	solved	missing mappings
CPA:	solved	missing mappings
CTA:	solved	missing mappings

Missing Mappings

This is not concerned whether the obtained mappings are correct, but only lists the tables where mappings are missing.
Lists at most 10 tables per category.
Mind the dependencies: If fewer CEA mappings are found, also the CTA and CPA mappings will deteriorate.

Fetch

CEA

No Mappings	*Missing Some Mappings*		
Name	Name	Missing	Total
KTG02AJT	SSHFDU1X	8	12
MT9YBQCI	W8D27W2Y	5	12
O37CZFX8	MUSOT714	5	6
HIUTFRBX	BMUHG00P	5	7
UCO0CVH3	5MHHNJQ3	5	6
8WQAZFK1	EJ4OTU7J	5	10
NFT3OLIY	CY56B1WH	4	5
OY8XY765	YVTS9L5B	4	5
BDIJMTQD	FPV9NTYO	4	7
3UPW0YHZ	J7W9A0NE	4	6

CPA

No Mappings	*Missing Some Mappings*		
Name	Name	Missing	Total
S332WBIC	BKRL2AKL	2	3
A3YQ16EC	5QNC4HK0	2	3
FUL936TS	9396Z88H	2	3
WJRYA69B	N224517P	2	3
TVXFU6CJ	LG9C41BR	2	3
GXOMA66V	9ZYYUTWM	2	3
TXC52I8Q	5P47EXDI	2	3
41TGT669	F2G9RECM	2	3
SYD5RIJJ	K33SSOS4	2	3
5HCEGPV0	9LKATUTG	2	3

CTA

No Mappings	*Missing Some Mappings*		
Name	Name	Missing	Total
F2BKSSWX	60QZBSXD	2	3
S332WBIC	NL104UJT	2	3
8CUEHHYY	U8SROQ4T	1	3
M21Z4QUF	I5YRVQ1E	1	2
HRJI55MZ	B606N5T7	1	2
41TGT669	YUG0GFQD	1	2
SYD5RIJJ	YDIQAKBS	1	2
AAWQ6NXC	7VLJ44NI	1	2
33G4RHA5	QZZ69YKK	1	2
KTG02AJT	OI8VRHPV	1	4

Audit Analysis

The data here shows the effectiveness for each method of pipeline in each step.

Fetch

	CEA	CTA	CPA
Creation	genericLookup, 1325446 tokens, 62074 fullCell, 6084	default, 1110	default, 921
Selection	stringSimilarity, 1371987 colSimilarity, 24314 missingCeaByCta, 17682	LCS, 662 directParents, 0 popularity, 0	majorityVote, 547

Fig. 20. JenTab: Manager screenshot.

0.91, 0.93, and 0.97 F1-scores (CEA, CTA, and CPA). In the biodiversity domain, JenTab was less effective. It achieves 60% and 41% for CEA and CTA tasks for the BiodivTab dataset. These scores are quite low compared to AG datasets due to the unique characteristics of the biodiversity dataset.

6.1. Limitations

JenTab relies on live lookup APIs and SPARQL query endpoints of target KGs. Thus, the performance of JenTab is constrained by these external services and their usage limits. Benchmark datasets and their ground truth are created at some point in time. Hence, looking for annotations in the current and live version of the target KG can have a negative effect on accuracy scores. Using the live APIs would be recommended for live deployments of JenTab, but for benchmark datasets, an offline dump will perform better.

CTA solutions are crucial. We investigated the effect of CTA selection on the scores of other tasks and developed three strategies relying on the direct types and their parents. Our findings show that these direct parents, e.g., "P31", suit easy-to-solve, auto-generated datasets. However, we found that adding one more level to these parents "2Hops" is a better fit for more challenging datasets. We do not recommend using more hierarchy levels like "MultiHops" since this produces very general, inaccurate solutions. JenTabs uses the assumption that semantic types are derived via the direct parents or higher levels. However in real-world scenarios like BiodivTab, the most fine-grained type may be given by another property, e.g., `wd:P105` in the case of taxon-related columns. JenTab detected these columns as "taxon", e.g., `wd:Q16521`, which is correct class but not the most fine-grained solution.

7. Conclusions & Future Work

We developed a modular approach called JenTab to solve STA tasks. This framework aims to match individual components of tabular data to a KG. JenTab is easily extensible and its components are highly configurable providing an easy way to change the target KG and enabling various settings to adapt to it. JenTab provides six pipelines for both Wikidata and DBpedia. JenTab won the second prize in the "Usability track" by IBM research in 2021. In addition, JenTab is the only system participant that obtained the "Artifact Availability Badge" of SemTab 2022. JenTab provides an overview and analysis of the current processing state via its Manager GUI. During its years of development, it achieved high accuracy scores for general domain benchmarks. On average and given all the Automatically Generated (AG) datasets, it has an average 0.93 F1-score for all three tasks. However, for the biodiversity domain it has 0.51 F1-score for the two supported tasks.

7.1. Future Work

- **Pre-computed lookup as a service** The pre-computed lookup is our primary source to tackle the misspellings. Due to its high resource requirements, we build this lookup before the actual start of JenTab's pipeline. It would be essential to convert this step into a live service integrated in the pipeline execution. This would speed up deploying and running JenTab and decrease its dependencies.
- **JenTab as a service** Currently, we support two execution modes: either by local setup or via docker containers. Deploying JenTab as a public service would increase its audience and users.
- **RDF as an output** JenTab supports the required formats of the SemTab challenges. For example, for CEA task, JenTab produces a CSV file containing "filename", "row_id", "col_id", and the annotation itself (cf. Section 6). However, for generating a KG over tabular data, RDF would be a better format to use than the current CSV file.
- **Interactive GUI** Currently, we support an overview of the processing state in the "Manager" node. However, input tables and targets are pre-configured. An interactive GUI should support uploading (batches of) tables and targets and selecting a target KG. Since we provide multiple pipelines, the GUI should also provide a list of all supported pipelines for users to choose from.
- **Target inference** JenTab relies on provided targets to annotate tables. For real-life scenarios, those targets should be inferred automatically.
- **Scoring system** Currently, the most powerful pipeline in "JenTab" is the `pipeline_full` using all provided modules. However, filter functions might be too harsh on the candidates dropping, e.g., correct mappings due a high support threshold. Switching to a scoring system that preserves all candidates and keeps them until a final selection would enhance the scores of JenTab.

– **Statistics Domain** We plan to use JenTab to transform statistical data given by data providers like World-Bank[26] and eurostat[27] into KGs.

Acknowledgements

The authors thank the Carl Zeiss Foundation for the financial support of the project “A Virtual Werkstatt for Digitization in the Sciences (K3, P5)” within the scope of the program line “Breakthroughs: Exploring Intelligent Systems for Digitization - explore the basics, use applications”. We thank Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, and Vincenzo Cutrona for organizing SemTab challenge and bringing insightful challenges to the community. In addition, we thank Cornelia Fürstenau and Franziska Zander for the fruitful discussions.

References

[1] Jixiong Liu, Yoan Chabot, Raphaël Troncy, Viet-Phi Huynh, Thomas Labbé, and Pierre Monnin. From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. *Journal of Web Semantics*, 76:100761, 2022. `doi:10.1016/j.websem.2022.100761`.

[2] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, and Kavitha Srinivas. SemTab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *The Semantic Web*, pages 514–530. Springer International Publishing, 2020. `doi:10.1007/978-3-030-49461-2\_30`.

[3] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, 2014. `doi:10.1145/2629489`.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer Berlin Heidelberg, 2007. `doi:10.1007/978-3-540-76298-0\_52`.

[5] Vincenzo Cutrona, Jiaoyan Chen, Vasilis Efthymiou, Oktie Hassanzadeh, Ernesto Jiménez-Ruiz, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, Daniela Oliveira, and Catia Pesquita. Results of SemTab 2021. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 1–12. CEUR-WS.org, 2021.

[6] Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Kavitha Srinivas. SemTab2019: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching - 2019 Data Sets, October 2019. `doi:10.5281/zenodo.3518539`.

[7] Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Kavitha Srinivas. SemTab 2020: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets, November 2020. `doi:10.5281/zenodo.4282879`.

[8] Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Kavitha Srinivas. SemTab 2021: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets, November 2021.

[9] Vincenzo Cutrona, Federico Bianchi, Ernesto Jiménez-Ruiz, and Matteo Palmonari. Tough Tables: Carefully Evaluating Entity Linking for Tabular Data, November 2020. `doi:10.5281/zenodo.4246370`.

[10] Oktie Hassanzadeh, Vasilis Efthymiou, and Jiaoyan Chen. SemTab 2022: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets - "Hard Tables" R1 and R2, December 2022. `doi:10.5281/zenodo.7416036`.

[11] Nora Abdelmageed, Sirko Schindler, and Birgitta König-Ries. Biodivtab: A table annotation benchmark based on biodiversity research data. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 13–18. CEUR-WS.org, 2021.

[12] Nora Abdelmageed, Sirko Schindler, and Birgitta König-Ries. BiodivTab: Semantic table annotation benchmark construction, analysis, and new additions. In *Proceedings of the 17th International Workshop on Ontology Matching co-located with the 21st International Semantic Web Conference (ISWC 2022)*. CEUR-WS.org, 2022.

[13] Nora Abdelmageed and Sirko Schindler. Jentab: Matching tabular data to knowledge graphs. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 40–49. CEUR-WS.org, 2020.

[14] Nora Abdelmageed and Sirko Schindler. JenTab Meets SemTab 2021's New Challenges. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 42–53. CEUR-WS.org, 2021.

[26]https://www.worldbank.org/en/home

[27]https://ec.europa.eu/eurostat

[15] Nora Abdelmageed and Sirko Schindler. Jentab: A toolkit for semantic table annotations. In *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*, volume 2873. CEUR-WS.org, 2021.

[16] Nora Abdelmageed and Sirko Schindler. JenTab: Do CTA solutions affect the entire scores? . In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 21th International Semantic Web Conference (ISWC 2022), Virtual conference, October 24, 2022*, volume 3320, pages 72–79. CEUR-WS.org, 2022.

[17] Nora Abdelmageed. *Heterogeneous data to knowledge graphs matching*. PhD thesis, University of Jena, Germany, 2023. URL: https://www.db-thueringen.de/receive/dbt_mods_00058422.

[18] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge Graphs. *ACM Comput. Surv.*, 54(4):1–37, 2022. doi:10.1145/3447772.

[19] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. *Knowledge Graphs*. Number 22 in Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool, 2021. doi:10.2200/S01125ED1V01Y202109DSK022.

[20] Sören Auer, Viktor Kovtun, Manuel Prinz, Anna Kasprzik, Markus Stocker, and Maria Esther Vidal. Towards a Knowledge Graph for Science. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, pages 1:1–1:6. ACM, 2018. doi:10.1145/3227609.3227689.

[21] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Kavitha Srinivas, and Vincenzo Cutrona. Results of semtab 2020. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 1–8. CEUR-WS.org, 2020.

[22] Brend Wanders. *Repurposing and probabilistic integration of data: an iterative and data model independent approach*. SIKS dissertation series, Universiteit Twente, Jun 2016. isbn:978-90-365-4110-7, number:2016-24. doi:10.3990/1.9789036541107.

[23] Yalin Wang and Jianying Hu. *Detecting Tables in HTML Documents*, volume 2423 of *Lecture Notes in Computer Science*, pages 249–260. Springer Berlin Heidelberg, 2002. doi:10.1007/3-540-45869-7_29.

[24] Gerald Penn, Jianying Hu, Hengbin Luo, and Ryan T. McDonald. Flexible Web Document Analysis for Delivery to Narrow-Bandwidth Devices. In *Proceedings of Sixth International Conference on Document Analysis and Recognition*, ICDAR-01, pages 1074–1078. IEEE Comput. Soc, 2001. doi:10.1109/ICDAR.2001.953951.

[25] Nora Abdelmageed, Jiaoyan Chen, Vincenzo Cutrona, Vasilis Efthymiou, Oktie Hassanzadeh, Madelon Hulsebos, Ernesto Jiménez-Ruiz, Juan Sequeda, and Kavitha Srinivas. Results of SemTab 2022. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 21st International Semantic Web Conference (ISWC 2022).*, volume 3320, pages 1–13. CEUR-WS.org, 2022.

[26] Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. Mtab: Matching tabular data to knowledge graph using probability models. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 7–14. CEUR-WS.org, 2019.

[27] Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. MTab4Wikidata at SemTab 2020: Tabular Data Annotation with Wikidata. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 86–95. CEUR-WS.org, 2020.

[28] Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. SemTab 2021: Tabular Data Annotation with MTab Tool. In Ernesto Jiménez-Ruiz, Vasilis Efthymiou, Jiaoyan Chen, Vincenzo Cutrona, Oktie Hassanzadeh, Juan Sequeda, Kavitha Srinivas, Nora Abdelmageed, Madelon Hulsebos, Daniela Oliveira, and Catia Pesquita, editors, *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 92–101. CEUR-WS.org, 2021.

[29] Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular Data. In Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, Kavitha Srinivas, and Vincenzo Cutrona, editors, *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 27–39. CEUR-WS.org, 2020.

[30] Viet-Phi Huynh, Jixiong Liu, Yoan Chabot, Frédéric Deuzé, Thomas Labbé, Pierre Monnin, and Raphaël Troncy. DAGOBAH: table and graph contexts for efficient semantic annotation of tabular data. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 19–31. CEUR-WS.org, 2021.

[31] Renat Shigapov, Philipp Zumstein, Jan Kamlah, Lars Oberländer, Jörg Mechnich, and Irene Schumm. bbw: Matching CSV to Wikidata via Meta-lookup. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 17–26. CEUR-WS.org, 2020.

[32] Gilles Vandewiele, Bram Steenwinckel, Filip De Turck, and Femke Ongenae. CVS2KG: transforming tabular data into semantic knowledge. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 33–40. CEUR-WS.org, 2019.

[33] Avijit Thawani, Minda Hu, Erdong Hu, Husain Zafar, Naren Teja Divvala, Amandeep Singh, Ehsan Qasemi, Pedro A. Szekely, and Jay Pujara. Entity linking to knowledge graphs to infer column types and properties. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 25–32. CEUR-WS.org, 2019.

[34] Marco Cremaschi, Roberto Avogadro, and David Chieregato. MantisTable: an Automatic Approach for the Semantic Table Interpretation. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 15–24. CEUR-WS.org, 2019.

[35] Marco Cremaschi, Roberto Avogadro, Andrea Barazzetti, and David Chieregato. MantisTable SE: an Efficient Approach for the Semantic Table Interpretation. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 75–85. CEUR-WS.org, 2020.

[36] Marco Cremaschi, Flavio De Paoli, Anisa Rula, and Blerina Spahiu. A fully automated approach to a complete semantic table interpretation. *Future Gener. Comput. Syst.*, 112:478–500, 2020. doi:10.1016/j.future.2020.05.019.

[37] Roberto Avogadro and Marco Cremaschi. Mantistable V: A novel and efficient approach to semantic table interpretation. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 79–91. CEUR-WS.org, 2021.

[38] Daniela Oliveira and Mathieu d'Aquin. ADOG - Annotating Data with Ontologies and Graphs. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 1–6. CEUR-WS.org, 2019.

[39] Shuang Chen, Alperen Karaoglu, Carina Negreanu, Tingting Ma, Jin-Ge Yao, Jack Williams, Andy Gordon, and Chin-Yew Lin. LinkingPark: An Integrated Approach for Semantic Table Interpretation. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 65–74. CEUR-WS.org, 2020.

[40] Donguk Kim, Heesung Park, Jae Kyu Lee, and Wooju Kim. Generating Conceptual Subgraph from Tabular Data for Knowledge Graph Matching. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 5, 2020*, volume 2775, pages 96–103. CEUR-WS.org, 2020.

[41] Bram Steenwinckel, Filip De Turck, and Femke Ongenae. MAGIC: mining an augmented graph using ink, starting from a CSV. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 68–78. CEUR-WS.org, 2021.

[42] Rabia Azzi and Gayo Diallo. *AMALGAM: A Matching Approach to Fairfy TabuLar Data with KnowledGe GrAph Model*, volume 1366 of *Advances in Intelligent Systems and Computing*, pages 76–86. Springer International Publishing, 2021. doi:10.1007/978-3-030-72651-5_8.

[43] Wiem Baazouzi, Marouen Kachroudi, and Sami Faïz. Kepler-aSI at SemTab 2021. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 20th International Semantic Web Conference (ISWC 2021), Virtual conference, October 27, 2021*, volume 3103, pages 54–67. CEUR-WS.org, 2021.

[44] Wiem Baazouzi, Marouen Kachroudi, and Sami Faiz. Yet Another Milestone for Kepler-aSI at SemTab 2022. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2022, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference*. CEUR-WS.org, 2022.

[45] Wiem Baazouzi, Marouen Kachroudi, and Sami Faiz. Kepler-aSI at SemTab 2023. In *SemTab'23: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching 2023, co-located with the 22nd International Semantic Web Conference (ISWC)*. CEUR-WS.org, 2023.

[46] Marco Cremaschi, Roberto Avogadro, and David Chieregato. s-elBat: a Semantic Interpretation Approach for Messy taBle-s. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 21st International Semantic Web Conference (ISWC 2022), Virtual conference, October 24, 2022*, volume 3320, pages 59–71. CEUR-WS.org, 2022.

[47] Azanzi Jiomekong and Brice Albin Foko Tagne. Towards an Approach based on Knowledge Graph Refinement for Tabular Data to Knowledge Graph Matching. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2022, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference*. CEUR-WS.org, 2022.

[48] Azanzi Jiomekong and Brice Albin Foko Tagne. Exploring Naive Bayes Classifiers for Tabular Data to Knowledge Graph Matching. In *SemTab'23: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching 2023, co-located with the 22nd International Semantic Web Conference (ISWC)*. CEUR-WS.org, 2023.

[49] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.*, 3(1–2):1338–1347, 2010. doi:10.14778/1920841.1921005.

[50] Sebastian Neumaier, Jürgen Umbrich, Josiane Xavier Parreira, and Axel Polleres. Multi-level semantic labelling of numerical values. In *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings,*

Part I, volume 9981 of *Lecture Notes in Computer Science*, pages 428–445. Springer International Publishing, 2016. doi:10.1007/978-3-319-46523-4_26.

[51] Vasilis Efthymiou, Oktie Hassanzadeh, Mariano Rodriguez-Muro, and Vassilis Christophides. Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 260–277. Springer International Publishing, 2017. doi:10.1007/978-3-319-68288-4_16.

[52] Yoan Chabot, Thomas Labbé, Jixiong Liu, and Raphaël Troncy. DAGOBAH: an end-to-end context-free tabular data semantic annotation system. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 18th International Semantic Web Conference, SemTab@ISWC 2019, Auckland, New Zealand, October 30, 2019*, volume 2553, pages 41–48. CEUR-WS.org, 2019.

[53] Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, volume 33, pages 29–36. Association for the Advancement of Artificial Intelligence (AAAI), 2019. doi:10.1609/aaai.v33i01.330129.

[54] Jiaoyan Chen, Ernesto Jimenez-Ruiz, Ian Horrocks, and Charles Sutton. Learning semantic annotations for tabular data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, IJCAI-2019, pages 2088–2094. International Joint Conferences on Artificial Intelligence Organization, 2019. doi:10.24963/ijcai.2019/289.

[55] Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, KDD '19, pages 1500–1508. ACM, 2019. doi:10.1145/3292500.3330993.

[56] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A Large Public Corpus of Web Tables containing Time and Context Metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web - WWW '16 Companion*, WWW '16 Companion, pages 75–76. ACM, 2016. doi:10.1145/2872518.2889386.

[57] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: table understanding through representation learning. *SIGMOD Rec.*, 51(1):33–40, 2022. doi:10.1145/3542700.3542709.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[59] Jan Martin Keil. Efficient bounded jaro-winkler similarity based search. P-289:205–214, 2019. doi:10.18420/BTW2019-13.

[60] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017. doi:10.1162/tacl_a_00051.

[61] W. E. Winkler. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage., 1990. ERIC.

[62] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady. Akademii Nauk SSSR*, 163(4):845–848, 1965.

[63] Daniela Oliveira and Catia Pesquita. Semtab 2021 biotable dataset, October 2021. doi:10.5281/zenodo.5606585.

[64] Nora Abdelmageed, Ernesto Jimènez-Ruiz, Oktie Hassanzadeh, and Birgitta König-Ries. tFood: Semantic Table Annotations Benchmark for Food Domain, April 2023. doi:10.5281/zenodo.7828163.

[65] Xinhe Li, Shuxin Wang, Wei Zhou, Gongrui Zhang, Chenghuan Jiang, Tianyu Hong, and Peng Wang. KGCODE-Tab Results for SemTab 2022. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 21st International Semantic Web Conference (ISWC 2022), Virtual conference, October 24, 2022*, volume 3320, pages 37–44. CEUR-WS.org, 2022.

[66] Viet-Phi Huynh, Yoan Chabot, Thomas Labbé, Jixiong Liu, and Raphaël Troncy. From Heuristics to Language Models: A Journey Through the Universe of Semantic Table Interpretation with DAGOBAH. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 21st International Semantic Web Conference (ISWC 2022), Virtual conference, October 24, 2022*, volume 3320, pages 45–58. CEUR-WS.org, 2022.

[67] Roberto Avogadro, Marco Cremaschi, Ernesto Jiménez-Ruiz, and Anisa Rula. A Framework for Quality Assessment of Semantic Annotations of Tabular Data. In *The Semantic Web - ISWC 2021 - 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24-28, 2021, Proceedings*, volume 12922 of *Lecture Notes in Computer Science*, pages 528–545. Springer International Publishing, 2021. doi:10.1007/978-3-030-88361-4_31.

[68] David Chaves-Fraga and Anastasia Dimou. Declarative Description of Knowledge Graphs Construction Automation: Status & Challenges. In *Proceedings of the 3rd International Workshop on Knowledge Graph Construction (KGCW 2022) co-located with 19th Extended Semantic Web Conference (ESWC 2022), Hersonissos, Greek, May 30, 2022*, volume 3141, 2022.