JenTab Meets SemTab 2021's New Challenges

Nora Abdelmageed[1−3 [0000−0002−1405−6860]] and Sirko Schindler[1,3 [0000−0002−0964−4457]]

[1] Heinz Nixdorf Chair for Distributed Information Systems
[2] Michael Stifel Center Jena, Germany
[3] Friedrich Schiller University Jena, Germany
{nora.abdelmageed, sirko.schindler}@uni-jena.de*

Abstract. While tables are a rich source of structured information, their automated use is oftentimes prevented by the inherent ambiguity contained within. Issues ranging from mere typos over inconsistent naming conventions to homonymy among values pose substantial barriers to exploiting this source of knowledge. Although the Semantic Web can alleviate many of these issues, the actual annotation process remains challenging. To foster new ideas and the improvement of existing approaches, the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) since 2019 hosts yearly competitions allowing systems to present their current capabilities. Datasets of different origins and characteristics highlight the various challenges present in this area. In this paper, we report on the evolution of our system, "JenTab", during SemTab2021. We re-designed the system architecture, optimized individual modules, and developed various pipelines to target specific challenges posed throughout the challenge. JenTab is among the top 5 systems in the first two rounds of SemTab2021. The results demonstrate JenTab's flexibility and its ability to quickly address new challenges.

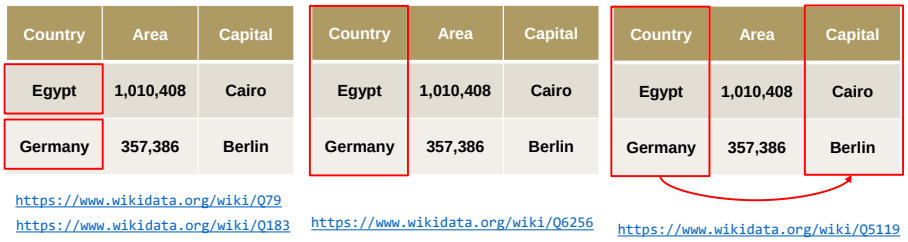
Keywords: Entity Linking · Cell Entity Annotation · Column Type Annotation · Column-Column Property Annotation · Semantic Table Annotation

1 Introduction

Tables are an essential tool when it comes to structuring large amounts of information. Nevertheless, they are hardly machine-interpretable in their raw form and are thus hidden from many automated processes. The annotation of regular tables with concepts from the Semantic Web faces various challenges, including misspellings, abbreviations, and the general ambiguity of the free text.

Over time, different approaches have been developed to cope with these issues and provide a semantic layer on top of common tables. The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[4] offers a forum for state-of-the-art systems to compare against one another and provides them with various datasets to

[4] http://www.cs.ox.ac.uk/isg/challenges/sem-tab/



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

Fig. 1: SemTab tasks summary [1].

challenge their capabilities. In its third year, it features a series of three rounds. Each round consists of a variety of raw tables. Such tables to be annotated with concepts either from Wikidata [13] or DBpedia [7].

The annotation tasks themselves are split into three areas, namely Cell Entity Annotation (CEA), Column Type Annotation (CTA), and Column Property Annotation (CPA). Given a data table and a target Knowledge Graph (KG), CEA links a cell to an entity within the KG (cf. Figure 1a). CTA is the task of assigning a semantic type (e.g., a class) to a column (cf. Figure 1b). Finally, CPA assigns a suitable semantic relation (predicate) from the KG to individual column pairs (cf. Figure 1c).

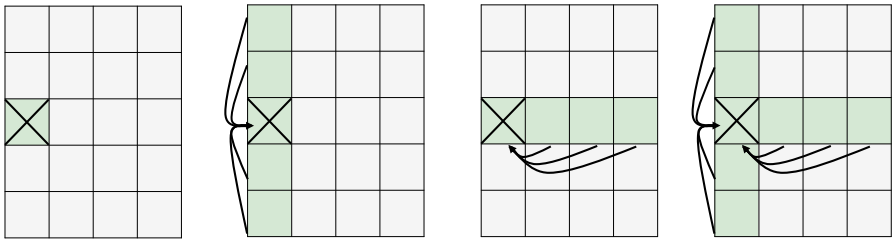
In this paper, we present the evolution of JenTab as a reaction to the new challenges introduced by SemTab2021. We have adopted the system to cover multiple Knowledge Bases (KBs) and explored various pipeline configurations based on the issues encountered. Moreover, we have enhanced the preprocessing steps to handle domain-specific datasets within the challenge. Finally, we optimized many components to improve the overall processing time.

The remainder of this paper is structured as follows. Section 2 summarizes the basic ideas and outlines JenTab's approach. Section 3 describes the new capabilities and changes made to address SemTab2021. Section 4 lists the challenges encountered in this year's datasets and the ways they are addressed in JenTab. Section 5 discusses our results for all three rounds of SemTab2021. Finally, Section 6 concludes the paper and gives an overview of our future work.

2 Background

This section provides an overview of the general approach JenTab follows. We briefly discuss the various table contexts exploited during the annotation process. Finally, we outline the default configuration of our pipeline. For more details, kindly refer to previous publications [1, 3].

JenTab follows a Create, Filter, and Select (CFS) pattern: It starts with creating a large pool of candidates or possible solutions for each table component (cells, columns, and column pairs). Subsequently, this set of candidates is iteratively filtered using various characteristics of the different table contexts and the current state of the annotation process. For example, an already determined CTA-solution may be used to remove CEA-candidates which are not an instance of the selected column type. Finally,



(a) Cell (b) Column (c) Row (d) Row-Column

Fig. 2: Possible contexts for resolving and disambiguating annotations for subject cells only [3]. Arrows indicate information used.

Fig. 3: Abstract view of the `pipeline_full`.

once the list of candidates is reduced, solutions for each task can be selected based, e.g., on their string similarity to the initial cell values.

As noted before, JenTab relies on different table contexts to reduce the number of considered candidates. These contexts are illustrated in Figure 2.

- The *Cell Context* relies on nothing else but the given value of each cell (cf. Figure 2a).
- The *Column Context* aggregates all cells of a single column and assumes that they represent the same characteristic of their respective tuples (cf. Figure 2b).
- The *Row Context* combines all cells of a single row and considers them as different aspects of the same entity (cf. Figure 2c)
- The *Row-Column Context* combines both the Row and Column context and thus represents all the direct information known for a single cell (cf. Figure 2d)

Figure 3 gives an abstract overview of JenTab's default pipeline, `pipeline_full`. We use 5 phases that each contains a series of modules. In the first phase, we create the initial set of candidates using only the cell context. The following component is a series of filtering modules leveraging both row and column context. The third phase represents our first attempt at selecting solutions. Afterwards, we turn our eye to cells without solutions and try to find candidates by exploiting row and column contexts. Finally, the last phase covers our last resort strategies for selecting a solution. For the complete configuration of our default pipeline, kindly refer to previous publications that provided a comprehensive overview of the system alongside a detailed evaluation of this pipeline [1, 3].

3 JenTab New Design

In this section, we discuss the current system architecture of JenTab, our newly developed annotation modules, and the various pipeline configurations we have developed.

3.1 System Architecture

Figure 4 illustrates the current design of JenTab. It is a simplified version of the previously used architecture [3]. We give an overview of the current components of the system, followed by a list of changes applied throughout SemTab2021.

Manager - The central service. Assigns tables to be processed to the actual Runners and collects results, errors, and audit records.
Runner - Worker node. Handles the communication between Manager and Solver(s).
Solver - Core service. Performs the preprocesing steps and executes the actual pipeline on an assigned table.
Wikidata Proxy and **DBpedia Proxy** - Encapsulate all interactions with the individual KGs, e.g., Wikidata [13] and DBpedia [7] for SemTab2021.
Generic Lookup - Holds a precomputed mapping to the target KG for the unique cell-values of the entire dataset using Jaro Winkler distance [14].
Caching Server - Centralized caching server. Reduces the number of queries issued through the proxy services by caching responses already retrieved before.

In the following, we list the changes compared to the previous year's system [3].

1. The formerly separate services Type Prediction and Clean Cells are now included within the Solver service itself. This reduced the amount of data transferred between individual nodes and thus sped up the execution time.
2. We added the DBpedia Proxy service to access the corresponding KG as a new source this year. SemTab2021 used DBpedia as a target KG in some of its challenges. The addition was reasonably straightforward as this new service resembles the corresponding Wikidata Proxy structure for the most part. Only queries that fetch entities, types, and properties had to be rewritten to use the live edition of the DBpedia SPARQL query endpoint[5]. The Solver service can now be configured to use either DBpedia or Wikidata as the target KG.
3. Previously, each instance of our services maintained its cache server. We employ a distributed setup of multiple instances of each service, which leads to duplicated efforts across the system. To reduce the inherent redundancy and increase the impact of the caching mechanism, we switched to a centralized caching server. This maximizes cache hits across all instances and thus reduces necessary queries to the live endpoints. Overall, this decreased the required processing time and made us less reliant on the live lookup and endpoint services of Wikidata/DBpedia.
4. The Generic Lookup was optimized and can now be relied upon without any other auto-correction strategies needed.

[5] `https://dbpedia.org/sparql`

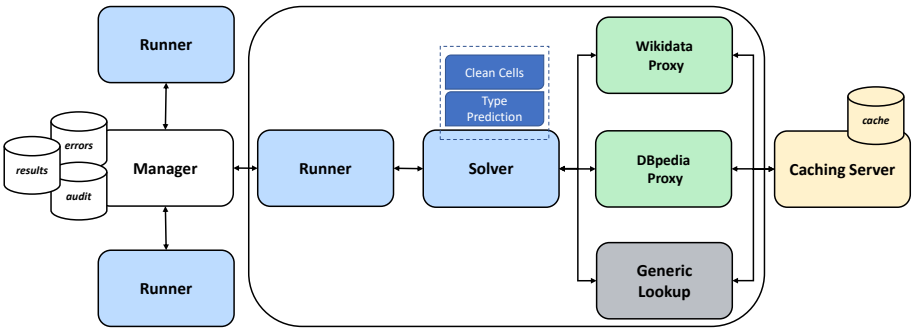


Fig. 4: JenTab: Current system architecture.

3.2 Modules

During SemTab2021 we revised some of the previously created modules (see Section 3.2 in [3]). For the most part, this included improving their runtime characteristics like execution time or memory consumption, fixing bugs, and considering edge cases that became apparent while trying to solve individual tracks' challenges. In addition to the modules developed previously, we added one more module:

`filter_topCpa` Similar to the previously available `filter_unmatchedCpa`, this filter considers the number of cells in the same row that can be matched to a specific subject candidate. Contrary to its predecessor, however, this new filter will determine those candidates that match their respective rows best. Subsequently, all other subject candidates have to be either of a compatible type or match at least as many cells. All other candidates will be removed.

3.3 Various Pipelines

Based on the original JenTab pipeline [1] 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. More details on the rationale behind each variation will be given in subsequent Section 4 that discusses the evolution of JenTab throughout this year's challenge.

`pipeline_full` This pipeline is the most powerful and the original implementation of JenTab. For more details, we kindly refer to previous publications [1, 3].

`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`.

(a) Clean table values. (b) Matched values in the noisy table

Fig. 5: Key table solution broadcast.

`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_keyTables` This pipeline works on datasets that are grouped into a key-value pair data structure. It represents clean tables without a large amount of artificially introduced misspellings. It solves those tables once using `pipeline_full` and subsequently applies the found solutions to associated tables in the same group. Solutions will be applied by identifying similar keys as shown in Figure 5.

`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 uses `filter_topCpa` to determine the most likely CEA and thus indirectly CTA-candidates. The latter is then used while applying `generate_cea_by_col` 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.

`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`.

4 JenTab Evolution

In this section, we describe JenTab's evolution during SemTab2021's three rounds. For each round, we discuss the provided tables, required tasks, and the target KG followed by our attempts and the used pipelines to tackle these challenges.

4.1 Round 1

In the first round, two datasets are given – each consisting of 180 tables. The tasks included only CEA and CTA - no CPA-solutions were required. Wikidata and DBpedia are the targets KGs.

The most crucial challenges about these datasets are ambiguity and noise. In addition, tables comprised more rows than most datasets of the past years. The datasets have been reported to be adapted versions of the Tough Tables dataset [8].

Initially, JenTab supported only Wikidata as a target KG as used and tested in the SemTab2020. However, the design was intended to support various KGs from the beginning. We have implemented `DBpedia_Proxy` that encapsulates the DBpedia lookup API[6] and the SPARQL queries necessary. Thus, with a simple configuration, the core `Solver` service can easily be changed to retrieve mappings from DBpedia instead of Wikidata and vice versa. In our first trial, we used the DBpedia spotlight [9] API[7] along with the DBpedia API to generate more cell candidates. We followed the cell lookup pipeline from IDLab 2019 [12]. In our case, we found a large number of false positives. Consequently, we decided against using DBpedia spotlight and limited the lookup to the core API only.

Ambiguity is a real challenge in the given tables. Since CPA-solutions were not required in this round, no corresponding targets were given. However, we believe column types alone are not sufficient as the only semantic context during the filter stages. Thus, we have added CPA-targets on our own and evaluated the impact during various runs with and without these CPA-targets. This resulted in `pipeline_no_cpa` as the first variation of our core pipeline, `pipeline_full`.

The massive amount of noise was another challenge during this round. For example, table `00E2H310` about actors contained more than 12K rows while overall just describing a set of 5 unique cells (actor names). Here, we have manually grouped the tables by similarity and solved only the cleanest ones using `pipeline_full`. The resulting solutions were subsequently transferred to their noisy counterparts. This approach resulted in `pipeline_keyTables` for the Wikidata dataset.

4.2 Round 2

In the second round, two new datasets are given. On the one hand, *HardTables* contains 1750 tables similar to previous tasks. On the other hand, *BioTables* features 110 tables from the biomedical domain. Both datasets required all three tasks – CEA, CTA, and CPA – and were based on Wikidata as the target KG.

A unique characteristic of BioTables is its rather wide tables with an average number of columns and rows of 6 and $\sim$ 2500, respectively. Initially, our `pipeline_full` was not able to cope with these characteristics and returned timeouts for many tables.

So in a first attempt, we developed `pipeline_essential` to cope with BioTables. It consists of the core parts of `pipeline_full` but resorts to filtering by properties only once. As this yielded mixed results, we turned our eye again to the implementation of individual modules. After close inspection and some restructuring within the modules, we were able to improve the performance of multiple modules up to the point that allowed us to run `pipeline_full` also on this dataset.

HardTables featured an increased level of ambiguity again. We even encountered examples of tables where we, as humans, could not determine proper solutions. However, with no particular angle on this kind of challenge, we employed our regular `pipeline_full` once again. Throughout several experiments, it continuously provided the best results.

[6] We use the live lookup API of DBpedia (`http://lookup.dbpedia.org/api/search/PrefixSearch`).

[7] `https://api.dbpedia-spotlight.org/en/candidates`

4.3 Round 3

In the third and final round this year, three datasets from different domains were given. For the general domain, *HardTables* contained 7207 tables with CEA, CTA, and CPA tasks required based on Wikidata as the target KG. Another general domain dataset is *GitTables* including 1101 tables with only the CTA task being required [10]. Unlike the other datasets in this round, it used DBpedia and schema.org as target KGs. Finally, *BiodivTab* [6] is provided as a domain-specific dataset. It was comprised of 50 tables inspired by biodiversity research data and provided CEA- and CTA-tasks using Wikidata.

The structure of tables in HardTables changed compared to previous datasets. This time, most tables contained only a single object column and multiple numeric ones. This structure makes some of the modules in `pipeline_full` obsolete as they try to infer subject candidates from other candidates found in the same row. Nevertheless, the row-context seemed more critical than the column-context in this dataset. Combined, these facts led to the development of `pipeline_numeric` and subsequently `pipeline_conditional` that uses the traditional `pipeline_full` as a backup if the initial attempts fail. The new `pipeline_numeric` emphasizes candidates that match their row over those that match other candidates in the same column. As evaluating the column-context first is computationally cheaper, `pipeline_full` used the column-context before the row-context. However, within this dataset, a reversal of this order proved more effective.

GitTables posed new challenges, in particular regarding the structure of tables and targets. Unlike other datasets where the subject column had always been the first column in a table, here, it could be located anywhere within a table. In another deviation from past datasets, proper CTA solutions could also involve properties. So far, CTA-solutions were always given by classes whereas the corresponding properties were subject to CPA. Other issues surrounding GitTables were of technical nature: In particular, the sparsity of cells in some columns provided hard challenges to our pipelines. Some columns for which CTA were requested even contained no cell value at all. Consequently, we approached GitTables as follows.

1. Parse the given targets to identify a) *subject-columns* as a the first column of a table that has a corresponding CTA-target, b) *object-columns* as all remaining columns having a CTA-target but not being a subject-column.
2. Reduce the `CTA-targets` to those subject-columns. We expect them to be annotated by semantic types.
3. Since we can solve neither CTA nor CPA without also tackling CEA, we artificially added `CEA-targets` for the subject-columns.
4. We established the `CPA-targets` between subject-columns and the corresponding object-columns. Again, we expect them to be annotated by semantic properties.
5. We have selected `pipeline_essential` to solve GitTables.
6. The final targets' solutions are given by a combination of CTA and CPA solutions.

Besides the default configurations that retrieve types from DBpedia, we have supported schema.org as well. We have configured the DBpedia_Proxy to filter the retrieved types to those from schema.org only and discard all others. To the best of our knowledge, there are no other means to access schema.org mappings dynamically, e.g., using a standalone lookup API.

The domain-specific dataset of BiodivTab also provided new challenges. In particular, there is only very limited context to disambiguate among possible candidates. Furthermore, we encountered the following challenges:

- Most of the data are numerical, measured in research laboratories, and thus could not be mapped to the target KGs.
- Abbreviations are extensively used for taxon names making it difficult to derive proper candidates. For example, *C.glauca* matches 48 plant species in Wikidata as of 14. October 2021.
- Multiple entities contained within a single cell: A column containing a researcher's name may also include the research institute at the same time.
- Values appear in condensed formats. We encountered species names represented in a structure like `species:abc sub:xyz`.
- Chemical elements are commonly given by their corresponding symbols. It may be seen as a special case of the abbreviations mentioned above.

We devised a custom treatment for some of the issues encountered. We adapted our preprocessing to cope with special formats like species names. To mitigate the impact of abbreviations, we constructed dictionaries for both taxons as well as chemical elements 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. We have tackled the nested entities problem by trying to parse individual cells and only retaining the first part. For example, *David Eichenberg (University of Halle-Wittenberg)* would be converted to only *David Eichenberg*. We used `pipeline_no_cpa` for two reasons: First, no CPA-targets were required. Second, we were not able to determine properties from Wikidata that would have connected subject candidates with the remaining values in their rows.

5 Experiences and Results

Table 1 shows an overview for the given datasets across all rounds. Only the HardTables-tasks in R2 and R3 resemble the characteristics of the challenges provided in previous years. All other tasks provide fewer but substantially larger tables. This growth in size can largely be attributed to an increased number of rows, whereas the number of columns remained relatively constant. BiodivTab and GitTables are the exceptions here as they also feature a two to three times increase in the number of columns. Overall, this shifted the focus towards CEA and put less emphasis on both CTA and CPA tasks. Furthermore, some datasets did not require all tasks to be solved: R1 and the BiodivTab dataset of R3 required no CPA solutions, whereas the GitTables dataset of R3 included only CTA-targets. To leverage all of JenTab's modules, we created missing targets most of the time. The respective number of targets are provided within Table 1.

Spelling mistakes and artificial noise are common challenges across SemTab2021's datasets. A generic lookup is our primary strategy for tackling this crucial issue. This lookup is created ahead of time. Due to the resources required for comparing cell values against all labels (and aliases) within Wikidata or DBpedia For this, we extract the unique values from all tables of a dataset and match those against the labels of the respective KG using an optimized Jaro-Winkler Similarity implementation based on [11] and a threshold for minimum similarity of 0.9. Table 2 illustrates the results of this approach. For most datasets, more than 89% of unique values could be matched

Table 1: SemTab2021 dataset. KGs: DBpedia (DBP), Wikidata (WD), and schema.org (SCH). Targets created by JenTab are marked with a star (*).

	R1		R2		R3		
	2T DBP	2T WD	BioTables WD	HardTables WD	BiodivTab WD	GitTables DBP & SCH	HardTables WD
Tables	180	180	110	1,750	50	1,101	7,207
Avg. Rows # (± Std Dev.)	1,080 ±2,798	1,080 ±2,798	2,448 ±193	17 ±8	259 ±743	58 ±95	8 ±5
Avg. Cols # (± Std Dev.)	5 ± 2	4 ± 2	6 ± 1	3 ± 1	24 ± 13	16 ± 12	2 ± 1
Avg. Cells # (± Std Dev.)	4125 ±10947	3952 ±10129	14605 ±2338	55 ±32	4589 ±10862	690 ±1159	20 ±15
CEA #	663,655	636,185	1,391,324	47,439	31,467	75,161* & 23,915*	58,948
CTA #	539	535	656	2,190	613	2,516 & 720	7,206
CPA #	359*	355*	546	3,835	NA*	1,687* & 444*	10,694

Table 2: Generic Lookup: Unique labels and ratio of resolved labels per round.

Rounds	Dataset	Target	Unique Labels	Unmatched	Matched	Matched (%)
R1	2T	Wikidata	69,980	7,072	62,908	89.89%
R1	2T	DBpedia	66,340	7,172	59,168	89.19%
R2	HardTables	Wikidata	249,625	600	249,025	99.76%
R3	HardTables	Wikidata	47,809	944	46,865	98.03%
R3	GitTables	DBpedia	37,780	21,253	16,527	43.75%

up to a 99.76% success rate for R2's HardTables. Only GitTables falls short, with less than half of the values being matched successfully. This may be attributed to the fact that the cell values here were not intended to be matched to the KG in the first place (no CEA targets were provided). We have not created a generic lookup for BiodivTab in Round 3, since our approach here relies on dictionaries for taxons and chemical elements - the prevalent type of cell values in this dataset.

Table 3 summarizes our results for R1 and R2 as reported by AICrowd[8]. R3 did not rely on AICrowd, and thus scores are unknown as of the time of writing. In the following, we discuss our results for each round. While R1 did not include CPA-targets as part of the challenge, we created such targets in order to use `pipeline_full`. This approach improved the results considerably. Further improvements were achieved by using `pipeline_keyTables` as described before. The mixed results indicate that JenTab is still challenged by the amount of noise introduced in these datasets. The 2T dataset

[8] `https://www.aicrowd.com/challenges/semtab-2021`

Table 3: Primary, secondary scores, and Ranks for JenTab. F1 - F1 Score, Pr - Precision, AF1 - Average F1 Score, and APr - Average Precision.

			CEA			CTA			CPA		
Rounds	Dataset	Target	F1	Pr	Rank	AF1	APr	Rank	F1	Pr	Rank
Round 1	2T	DBpedia	0.607	0.669	3^{rd}	0.460	0.468	1^{st}	NA	NA	NA
Round 1	2T	Wikidata	0.457	0.520	3^{rd}	0.697	0.697	2^{nd}	NA	NA	NA
Round 2	HardTables	Wikidata	0.966	0.967	4^{th}	0.914	0.917	4^{th}	0.996	0.997	2^{nd}
Round 2	BioTables	Wikidata	0.857	0.858	4^{th}	0.835	0.843	5^{th}	0.899	0.899	2^{nd}

includes a vast number of artificially created misspellings that severely impact the performance of JenTab. We have tackled the ambiguity, especially in the R2 HardTables using `pipeline_full`. Our highest scores are achieved at CPA task. Thanks to our fuzzy matching technique [1], we managed to capture a wide range of properties.

6 Conclusions & Future Work

In this paper, we have reported on the updates to JenTab as part of the 2021-edition of Semantic Web Challenge on Tabular Data to Knowledge Graph Matching. We discussed the changes to the system architecture. We further created a variety of pipelines, each catering to the specific requirements of the individual datasets of the challenge. JenTab remains a top-5 system among participants. Our code is publicly available [2][9]. Moreover, our precomputed generic lookup [5] and solution files [4] for each round of SemTab2021 are also publicly available.

We see various areas for further improvement. First, the binary decision of whether to keep candidates or remove them should be replaced by a scoring system that emphasizes well-supported candidates but maintains other options. A further challenge to address is the lack of targets for specific tasks either voluntarily or by lack of corresponding structure in the target KG. As witnessed in particular by the GitTables dataset, JenTab struggles in such cases. Finally, we continuously strive to improve the performance of JenTab in particular to reduce the number of timeouts received from the public endpoints of KGs.

Acknowledgment

The authors thank the Carl Zeiss Foundation for the financial support of the project "A Virtual Werkstatt for Digitization in the Sciences (P5)" within the scope of the program line "Breakthroughs: Exploring Intelligent Systems" for "Digitization - explore the basics, use applications". We would further like to thank Muhammad Abbady and Sarah Böning for the fruitful discussions throughout the challenge. Last but not least, we would thank Birgitta König-Ries her guidance and continuous feedback.

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

References

1. Abdelmageed, N., Schindler, S.: Jentab: Matching tabular data to knowledge graphs. In: SemTab@ ISWC. pp. 40–49 (2020)
2. Abdelmageed, N., Schindler, S.: fusion-jena/jentab (2021). https://doi.org/10.5281/zenodo.5584721
3. Abdelmageed, N., Schindler, S.: Jentab: A toolkit for semantic table annotations. In: Knowledge Graph Construction Workshop (KGC), ESWC (2021)
4. Abdelmageed, N., Schindler, S.: JenTab Solution Files for SemTab 2021 (Oct 2021). https://doi.org/10.5281/zenodo.5584538
5. Abdelmageed, N., Schindler, S.: JenTab_precomputed_lookup_2021 (Oct 2021). https://doi.org/10.5281/zenodo.5584447
6. Abdelmageed, N., Schindler, S., Knig-Ries, B.: fusion-jena/BiodivTab (Oct 2021). https://doi.org/10.5281/zenodo.5584180
7. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. pp. 722–735. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-76298-0_52
8. Cutrona, V., Bianchi, F., Jimnez-Ruiz, E., Palmonari, M.: Tough Tables: Carefully Evaluating Entity Linking for Tabular Data (Nov 2020). https://doi.org/10.5281/zenodo.4246370
9. Daiber, J., Jakob, M., Hokamp, C., Mendes, P.N.: Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems (I-Semantics). ACM Press (2013). https://doi.org/10.1145/2506182.2506198
10. Hulsebos, M., Demiralp, Ç., Groth, P.: Gittables: A large-scale corpus of relational tables. CoRR **abs/2106.07258** (2021), `https://arxiv.org/abs/2106.07258`
11. Keil, J.M.: Efficient bounded Jaro-Winkler Similarity based search. BTW 2019 (2019). https://doi.org/10.18420/BTW2019-13
12. Steenwinckel, B., Vandewiele, G., De Turck, F., Ongenae, F.: Csv2kg: Transforming tabular data into semantic knowledge. SemTab, ISWC Challenge (2019)
13. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (sep 2014). https://doi.org/10.1145/2629489
14. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. (1990)