Journal of open research software

MantisTable UI: A Web Interface for Comprehensive Semantic Table Interpretation Curation and Management

SOFTWARE METAPAPER

MARCO CREMASCHI
FABIO D'ADDA
SARA NOCCO
*Author affiliations can be found in the back matter of this article

ubiquity press

ABSTRACT

Semantic Table Interpretation (STI) is essential for data analysis and knowledge graph construction. However, current solutions lack accessible interfaces for managing STI workflows. *MantisTable UI* has been developed as a modular, web-based platform designed to simplify STI processes through an intuitive interface and an extensible plugin system supporting data transformation and export. The tool enables annotation visualisation and integration of custom modules. Usability testing indicated high effectiveness and user satisfaction. Being open-source and publicly available, *MantisTable UI* offers a practical solution for enhancing semantic annotation and facilitating the semantic enrichment of tabular data.

CORRESPONDING AUTHOR:
Marco Cremaschi
Department of Informatics, Systems and Communication – DISCo, University of Milano-Bicocca, Milan, Italy
marco.cremaschi@unimib.it

KEYWORDS:
Semantic Table Interpretation; Tabular Data Annotation; Knowledge Graph; KG construction; KG completion; Open Source Semantic Tools

TO CITE THIS ARTICLE:
Cremaschi M, D'Adda F, Nocco S. 2025 MantisTable UI: A Web Interface for Comprehensive Semantic Table Interpretation Curation and Management. *Journal of Open Research Software*, 13: 34. DOI: https://doi.org/10.5334/jors.583

(1) OVERVIEW

INTRODUCTION

In the digital era, a significant volume of structured data about a wide range of domains is embedded in tables on the Web. These tables hold enormous potential for data analysis and machine learning algorithms applications, including classification, clustering, filtering, and retrieval [14]. However, tabular data is neither easily query-able nor semantically interpretable due to their implicit or visual structures, which make them designed for a straightforward human understanding [8]. For computers to effectively interpret, combine, and leverage this data, it is essential to make the semantics explicit by identifying and annotating entities in the cells, their types, and the relation between them [8].

Given this premise, the tasks of the Semantic Table Interpretation (STI) constitute a powerful method to annotate elements in a table. By STI, it is intended to produce semantically annotated tables by matching information in Knowledge Graph (KG) to tabular data. STI enables KG construction and KG completion, that is, respectively, the organisation of triples of information in the form of (head entity, relation, tail entity) or <subject, predicate, object>, and the process of completing incomplete triples [22]. STI has gained increasing interest among researchers, as can be demonstrated by the organisation of the International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) challenge. This initiative has been proposed since 2018 and repeated annually, with new datasets, approaches, and related Gold Standards (GSs) released yearly to benchmark STI systems.

The research on STI tasks has led to creating and refining a set of components to implement and continuously improve all the processes involved in constructing KGs. However, despite many STI approaches have been developed over the last years, none of them is equipped with a Graphical User Interface (GUI) that currently allows users to dynamically execute the STI process. Therefore, we stress the need for a tool supported by an intuitive GUI to handle the entire set of functionalities required for STI, including the export step, namely the operation of producing the Resource Description Framework (RDF) triples from annotated tables.

This paper discusses the implementation of a new tool with a graphical interface designed for managing the input and output of a STI approach through APIs. We will be referring to this new tool with "*MantisTable UI*",[1] with its name derived from the set of developed STI approaches, namely MantisTable V [1], which is, in turn, an extension and improvement of MantisTable SE [4], and MantisTable [8].

The main functionalities of *MantisTable UI* are (i) the ability to manage and visualise the input and output of STI approaches through APIs, (ii) the possibility of exporting RDF triples from the annotated tables, and (iii) the availability of a plugin system to extend the default *MantisTable UI* capabilities, providing additional functionalities such as lexicalisation, visualisation, and export in user-defined formats.

To clarify the functionalities of *MantisTable UI*, a definition of STI is provided. In its most agreed and complete formalisation, the STI process considers two inputs: i) a relational table, which is usually assumed to be *well-formed and normalised* (*i.e.*, the table has a grid structure, where the first row may contain the table headers and any other row contains values), as in Figure 1; and ii) a reference *KG* with its vocabulary (*i.e.*, a set of symbols denoting concepts, datatypes, properties, instances, also referred to as *entities* in the following) as in Figure 2. The output of the STI process is a semantically annotated table, *i.e.*, a table where its elements, typically values, columns, and column pairs, are annotated with symbols from the KG vocabulary. The exact specification of the annotations expected as the output of the STI process may differ in the proposed approaches. Here we discuss a canonical definition of a *semantically annotated table* to provide a first understanding of key STI tasks, inspired by the SemTab Challenge, where the annotation process has been better formalised with a community-driven effort.

To discuss this canonical definition, we use the example reported in Figure 3.

Name	Coordinates	Height	Range
Mont Blanc	45°49'58"N, 6°51'54"E	4808	M. Blanc massif
Hohtälli	45°59'20"N, 7°48'10"E	3275	Pennine Alps
Cervino	45°58'35"N, 7°39'31"E	4478	Pennine Alps

Input Data

Figure 1 Example of a well-formed relational table.

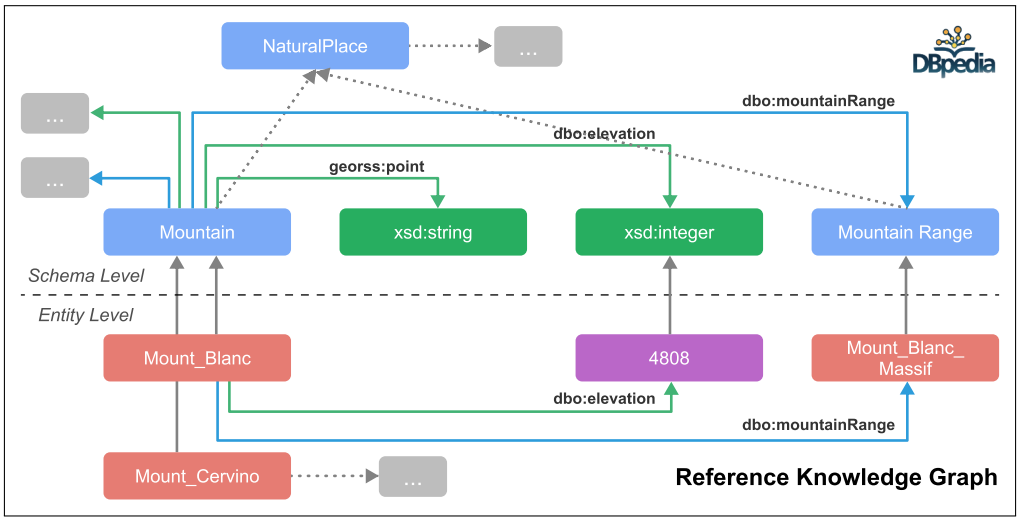

Figure 2 A sample of Knowledge Graph.

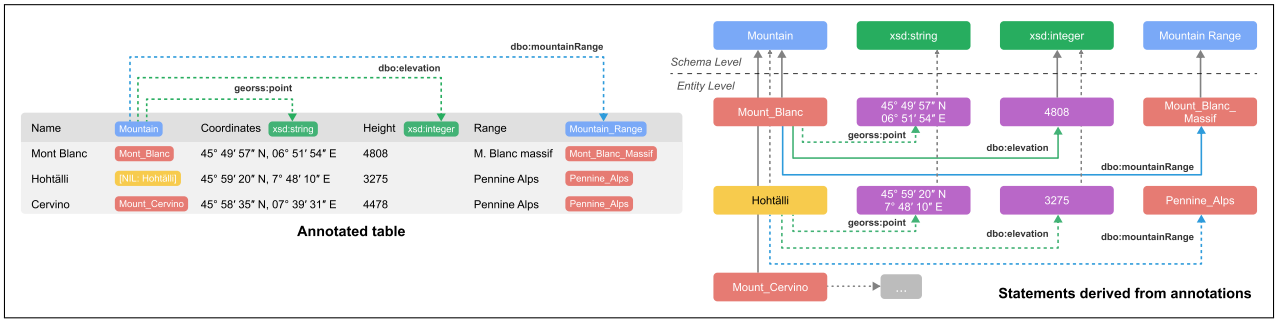

Figure 3 Example of an annotated table.

Given:

- a relational table *T* (Figure 1);
- a Knowledge Graph and its vocabulary (Figure 2).

T is annotated when:

- each column is associated with one or more types from the KG [Column-Type Annotation (CTA)]; *e.g.*, the column *Name* in Figure 1 is annotated with the type `Mountain`; the column *Height* is annotated with datatype `xsd:integer`;
- each cell in "entity columns" is annotated with an entity identifier from the KG or with *NIL*, if no entity in the KG corresponds to the cell value [Cell-Entity Annotation (CEA)]; *e.g.*, the cell *Le Mount Blanc* is annotated with `Mont_Blanc`; the cell *Hohtälli* is annotated with *NIL* since it has not yet been included in the KG;
- some pairs of columns are annotated with a binary property [Columns-Property Annotation (CPA)]; *e.g.*, the pairs composed by the columns *Name* and *Height* are annotated with `dbo:elevation`.

The result of the annotation process for the table considered in the example is shown in Figure 3. Note that each bullet point can be interpreted as a high-level STI task to be completed. Additionally, the annotations may identify new entities in the table that are not present in the reference KG (*e.g.*, `Hohtälli`).

In this context, it is important to distinguish between Named Entity (NE) and Literal (LIT) columns. NE columns contain values that can be directly linked to unique entities in a KG (*e.g.*, *Mont Blanc* → `dbr:Mont_Blanc`). These values represent real-world objects or concepts that can be unambiguously referenced. Conversely, LIT columns contain values that are not entities but literals, such as numbers, dates, or plain text strings (*e.g.*, elevations expressed as integers, coordinates, or descriptive labels). In such cases, annotations specify datatypes or literal values (*e.g.*, `xsd:integer`, `xsd:string`), rather than entity identifiers. This distinction is fundamental, as STI tasks like CEA apply to NE columns, while LIT columns are primarily handled through datatype recognition and are involved in CPA when paired with NE columns.

Having clarified the concept of STI, we proceeded to identify the tools with GUI developed over the last few years that perform the annotation process. In the state-of-the-art (SOTA), there are twelve STI tools with a user interface (UI): 1) OpenRefine,[2] 2) Karma[3] [11], 3) Trifacta,[4] 4) TableMiner+ [13], 5) Odalic,[5] 6) ESKAPE [16],

7) MantisTable[6] [5], 8) DataGraft+ASIA[7] [9, 18], 9) MTab tool [15], 10) MAGIC[8] [20], 11) SemTUI[9] [17], and 12) DAGOBAH UI[10] [19].

An inspection of the current accessibility and usability of the listed tools assessed that the great majority of them are currently no longer available. OpenRefine is the only one that can be successfully accessed and utilised. It has a graphical user interface and is available in more than 15 languages, and its main features include Wikidata reconciliation, Wikidata editing, and data augmentation. Odalic was installed via Docker image, which permitted the selection of two inputs: (i) a table of data in CSV format and (ii) one or more KG. The latter can be chosen among ADEQUATe, DBpedia, German DBpedia, Local KB, and OpenData, which can be combined to improve the annotation results. It is also possible to import/add a new KG. Nonetheless, STI process-related features are not working, which marks this second tool as accessible but not functional. On the other hand, Trifacta is now Alteryx,[11] an AI Platform for Enterprise Analytics. Indeed, Trifacta was acquired by the computer software company Alteryx in January 2022. Two months later, Alteryx announced its Analytics Cloud Platform, which it updated to integrate Designer Cloud powered by Trifacta.

Concerning the other nine listed STI tools, all of them are currently unreachable, *i.e.*, their installation cannot be completed or, if they are successfully installed, their GUI cannot be reached. Our previous MantisTable tool is currently deprecated as it was tightly coupled with the homonym approach, which was successively modified in an extensive way without leading to any interface updates.

As it is not possible to directly check the availability of the original functionalities of the entirety of the developed STI tools, due to the impossibility of accessing or using some of them, the following analysis derives from the information provided in each tool's respective paper.

Table 1 provides a comparison of the described tools based on some key functionalities: (i) table import, (ii) table import via API, (iii) import of ontologies, (iv) definition of personalised ontologies, (v) semi-automatic annotation/human-in-the-loop (HITL), (vi) semantic annotation suggestions, (vii) auto-complete support for the STI process, (viii) STI subtasks (CEA, CTA, and CPA), (ix) table manipulation, (x) automatic table extension, (xi) graphical visualisation of the annotations, (xii) auto-save of current user workspace, (xii) export mapping, (xiii) RDF triples export, and (xiv) open source. Moreover, it reports the release year for each tool. None of the tools includes all the key aspects required for the semantic enrichment of tabular data, with SemTUI equipped with most of the selected functionalities. It was impossible to determine the functionalities handled by ESKAPE, due to the unavailability of the tool and insufficient information in the relative papers.

Considering the scarcity of working tools capable of handling key STI features, we further emphasise the necessity of building a novel tool that can simplify the management of STI processes throughout an intuitive GUI. Decisions on the functionalities to be included derive from the SOTA requirements analysis, and their application in *MantisTable UI* is explained in the following Section.

IMPLEMENTATION AND ARCHITECTURE

From the analysis of the SOTA, it is clear that developing a comprehensive tool managing the input and output of a STI approach can be significantly relevant to any user, necessitating an accessible framework to make the annotation process effective. In this Section, the following elements are presented: (i) *Requirements fulfilment* illustrates the key STI requirements depicted in Table 1 as they are integrated into *MantisTable UI*, (ii) *Architecture* offers an explanation of the architecture of the implemented framework, and (iii) *MantisTable UI in Practice* provides an overview of the interaction with the platform and of the functioning of its plugin system.

(i) Requirements fulfilment

The examination of the SOTA aimed at identifying a set of requirements that a tool managing STI processes should ideally be endowed with, to fully encapsulate the entire semantic enrichment experience. The fulfilment of these requirements guided the structuring of *MantisTable UI*. Table 2 maps each requirement to the corresponding *MantisTable UI* implementation, when applicable.

Import of tables, visualisation of annotations, and auto-save are already fully supported by *MantisTable UI* v1.0. Additionally, export functionalities and automatic table extensions are implemented in the platform via the export and transformation plugins. The plugin system is explained below. Integrating additional features using plugins allows users to customise outputs independently from the *MantisTable UI* core system, ensuring stability.

Other requirements, such as the import and definition of personalised ontologies, as well as subject column detection, are not considered for direct integration within *MantisTable UI* since they must be directly managed by the STI approach rather than within the platform itself. The STI approach must also implement the core STI tasks represented by CEA, CTA, and CPA for Named Entity (NE) and Literal (LIT) columns and visualise within *MantisTable UI*. Semi-automatic annotation with human-in-the-loop (HITL) is supported through manual editing of annotations.

Future development is planned for the import of tables via API, the integration of annotation suggestions, and auto-complete support.[12]

Table manipulation is not supported, marking it as the only requirement not met by *MantisTable UI*.

FUNCTIONALITIES	OPENREFINE	KARMA	TRIFACTA	TABLEMINER+	ODALIC	ESKAPE	MANTISTABLE	DATAGRAFT	MTAB	MAGIC	SEMTUI	DAGOBAH	MANTISTABLE UI
Import & Export													
Import of tables	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y
Import of tables via API	N	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	F
Import of ontologies	N	Y	N	N	N	–	N	Y	N	N	N	N	O
Export mapping	N	Y	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
Export RDF triples	Y	Y	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
Annotation & Assistance													
Definition of personalised ontologies	N	N	N	N	N	–	N	Y	N	N	N	N	O
Semi-automatic annotation/HITL	N	Y	N	Y	Y	–	Y	N	Y	Y	N	Y	N
Annotation suggestions	N	Y	N	N	N	–	Y	Y	N	Y	Y	N	F
Auto-complete support	Y	Y	N	N	N	–	Y	Y	N	N	Y	N	F
Subject column detection	N	Y	N	Y	Y	–	Y	N	Y	N	Y	Y	O
CEA	Y	N	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
CTA	N	N	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
CPA (NE columns)	N	N	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
CPA (LIT columns)	N	N	N	Y	Y	–	Y	Y	Y	Y	Y	Y	I
Table management & Visualisation													
Table manipulation	Y	Y	Y	N	N	–	Y	Y	N	N	Y	N	N
Automatic table extension	Y	N	Y	N	N	–	N	Y	N	Y	Y	N	I
Visualisation of annotations	N	N	N	N	N	–	Y	Y	Y	Y	Y	Y	Y
Auto-save	Y	Y	Y	N	N	–	N	Y	N	N	N	N	Y
Open source & metadata													
Open Source	Y	Y	N	Y	Y	–	Y	Y	N	Y	Y	N	Y
Release year	**2010**	**2012**	**2012**	**2016**	**2016**	**2017**	**2019**	**2019**	**2021**	**2021**	**2022**	**2022**	**2024**

Table 1 Comparison of semantic table interpretation tools based on key functionalities.
Legend: Y = available; N = not available; – = not verifiable. In the "MantisTable UI" column: F = planned future support; I = implemented by the STI approach and visualised in the UI; O = out of scope for the UI.

REQUIREMENTS	*MantisTable UI*
Import of tables	Supported
Import of tables via API	Future development (v2.x)
Import of ontologies	Not considered as it should be directly integrated into the STI approach
Definition of personalised ontologies	Not considered as it should be directly integrated into the STI approach
Semi-automatic annotation/HITL	Manual editing of the annotations (future development v1.2)
Annotation suggestions	Future development (v1.3)
Auto-complete support	Future development (v1.3)
Subject column detection	Not considered as it should be handled by the STI approach
CEA, CTA, and CPA (NE columns and LIT columns)	Implemented by the STI approach and visualised in *MantisTable UI*
Table manipulation	Not supported
Automatic table extension	Implemented via transformation plugins
Visualisation of annotations	Supported
Auto-save	Supported
Export mapping	Implemented via export plugins
Export RDF triples	Implemented via export plugins

Table 2 List of requirements for *MantisTable UI*.

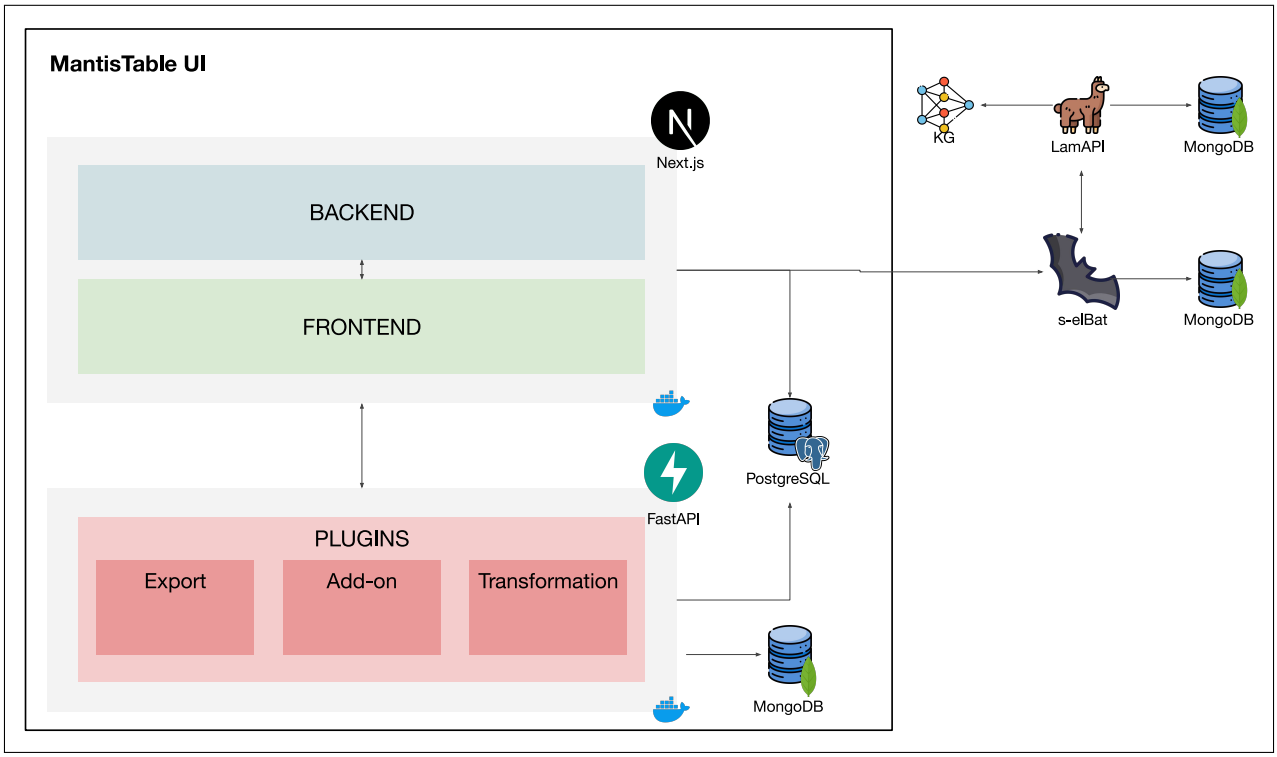

Figure 4 MantisTable UI Architecture.

(ii) Architecture

This Section illustrates the architecture of *MantisTable UI* along with all the technologies used to build the entire system.

The architecture includes multiple independent software modules that cooperate by exchanging data, resulting in a cohesive system that provides comprehensive functionality and allows for scalability, flexibility, and easier maintenance. Figure 4 shows the different software components implemented and deployed using Docker[13] containers. The core module has been integrated into a web application developed with T3 Stack,[14] a TypeScript full-stack framework. T3 Stack is based on Next.js[15] which is a popular React framework that provides server-side rendering, static site generation, and other advanced

features like API routes, image optimisation, and file-based routing.

A description of each component/module of the architecture will be provided below.

1. **Frontend Module**: the module is responsible for the user interaction during table annotation. The technologies of the T3 Stack that have been integrated into this module are: (i) *TypeScript*,[16] a strongly typed programming language that builds on JavaScript, providing static typing to catch errors early in the development process; (ii) *Tailwind CSS*,[17] a utility-first CSS framework that provides low-level utility classes to build responsive and modern UIs quickly. Tailwind CSS promotes rapid development and allows for highly customisable and consistent styling across the application.
2. **Backend Module**: the backend module is fundamental as it interacts with requests from the frontend (which are made by the user), and then performs all the operations behind the scenes. The main role of this component is to collect requests from the frontend, save the data in the database, and interact with the execution of plugins. In this module, the technologies used from T3 Stack are: (i) *tRPC*,[18] a TypeScript-based remote procedure call framework which allows the building of type-safe APIs. It provides end-to-end type safety, enabling the developer to define the APIs in a single place and get full type inference across the stack; (ii) *Drizzle*,[19] an ORM tool for *Node.js* and *TypeScript* that provides a type-safe database client. It simplifies database interactions by offering an intuitive schema definition language and powerful query capabilities, and it supports popular databases such as PostgreSQL and MySQL.
3. **Plugins Module**: the last component of the architecture is represented by the *Plugins Module*, which allows the application to load plugins. The implementation of the plugins is separate from the T3 Stack and has been developed using the FastAPI[20] backend framework. Plugins are written in Python as it provides a vast ecosystem of libraries, especially for data manipulation, transformation, and export tasks. Libraries like *pandas*, *numpy*, and *scipy* make data handling and manipulation straightforward and efficient. Python was also chosen for its simplicity, readability, and versatility. When selected, each plugin is loaded in an isolated environment and executed independently.
4. **Databases**: Uploaded tables and their respective annotations are permanently stored in a *PostgreSQL*[21] database, which is perfectly integrable in the *Next.js* system, and enables to retrieve tables efficiently. On the other hand, the *Plugins* service relies on *Mongo*[22] database to store plugin information.

The above explained architecture and technologies allow for application-level parallelism and simplify server deployment. The GUI introduced in this paper aims to complete the deployment of all the discussed tools, enabling the STI process, providing the possibility to graphically explore annotations for every phase and easily execute the export step, as well as the plugin of the Python modules as zip files.

The source code for *MantisTable UI* is freely available[23] and AGPL licensed, enabling users to access, modify, and contribute to the platform. By providing open access to the codebase and detailed documentation,[24] we encourage collaboration and innovation within the community, allowing developers to tailor the tool to their specific needs, integrate new features, and enhance its overall functionality through the plugin system.

5. **External Modules**: Table annotation is performed by **s-elBat** and **LamAPI**:

- ***LamAPI***[25] provides fast and structured access to Wikidata, offering full-text search, entity relationships, and literal values. It ingests and indexes Wikidata dumps in MongoDB and Elasticsearch, enabling efficient lookup and analysis for downstream applications such as entity linking and table annotation;
- ***s-elBat***[26] performs entity linking on tabular data by mapping table cells to Knowledge Graph entities. Its pipeline includes column classification, candidate retrieval via **LamAPI**, feature extraction, and final prediction. It supports high-quality semantic labelling, human-in-the-loop validation, and data enrichment.

(iii) *MantisTable UI* in Practice

MantisTable UI integrates a default STI approach: s-elBat [2, 6], whose name comes from taBle-s and Semantic Entity Linking to BAtch Table. s-elBat is a supervised and unsupervised STI approach that incorporates various strategies to address all STI challenges. Based on the experience acquired from those solutions and their involvement in the SemTab Challenge, it integrates a feature vector-based entity disambiguation (ED) approach. This approach combines both heuristic and machine learning techniques to achieve precise disambiguation results. *MantisTable UI* also allows users to integrate their approaches via API interaction. The API must consist of two primary endpoints:

1. *Semantic Table Interpretation*, which processes a given table and performs semantic interpretation;
2. *Get Table Annotations*, which returns the table and its annotations.

Information about endpoints can be set in the configuration file.[27]

Having illustrated the default STI approach and the possibility of integrating external approaches with an API, an explanation of how to use *MantisTable UI*[28] in practice is provided in this subsection, mainly focusing on the novel plugin system.

Figure 5 *MantisTable UI* Landing Page.

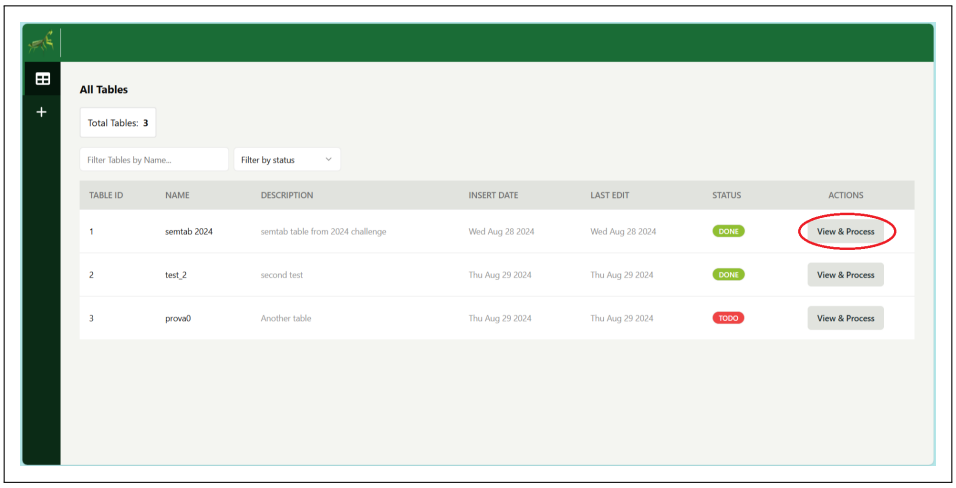

Figure 6 *MantisTable UI* Imported tables – Homepage. Highlight on the "View & Process" button.

In particular, a description of each page constituting the GUI and an overview of its functionalities is provided below:

- **Landing page**: displays the *MantisTable UI* logo and a short description of the tool. Via the "get started" button, users can access the homepage from which they can either move to a window to visualise the imported tables or go to another window where new table import can be executed (Figure 5);
- **Imported tables – Homepage**: in this window, a table reports information about each of the already imported tables, including ID, name, description, insert date, last edit date, and status (one between "DONE", indicating the table is already annotated, and "TODO", indicating the opposite). Tables can be filtered by name and status. A "View & Process" button associated with each of the imported tables guides users to the "View & Process" page (Figure 6);
- **Table import – Homepage**: allows the import into the platform of new, not yet annotated tables. For each new insertion, a name and a description of table content are required before selecting and uploading the file (Figure 7). *MantisTable UI* currently supports the import of tables in CSV format only, meaning tables in every other format must be converted to CSV before insertion into the platform. New tables can be visualised in the imported tables homepage described above;
- **View & Process page**: if a table has already been processed and is therefore associated with the status "DONE" in the homepage, users can find it in the "View & Process" page (Figure 8). Visualising the results of the CTA, CEA and CPA tasks is possible. The resulting entity IDs associated with each mention are displayed in the original table, and by clicking on them, users are redirected to the corresponding Wikidata entity pages. If the selected table has not yet been processed, it is possible to proceed with the annotation task via the corresponding button and to check the status of the process, eventually refreshing the page. Successively, users can visualise the annotation result as described above, as well as

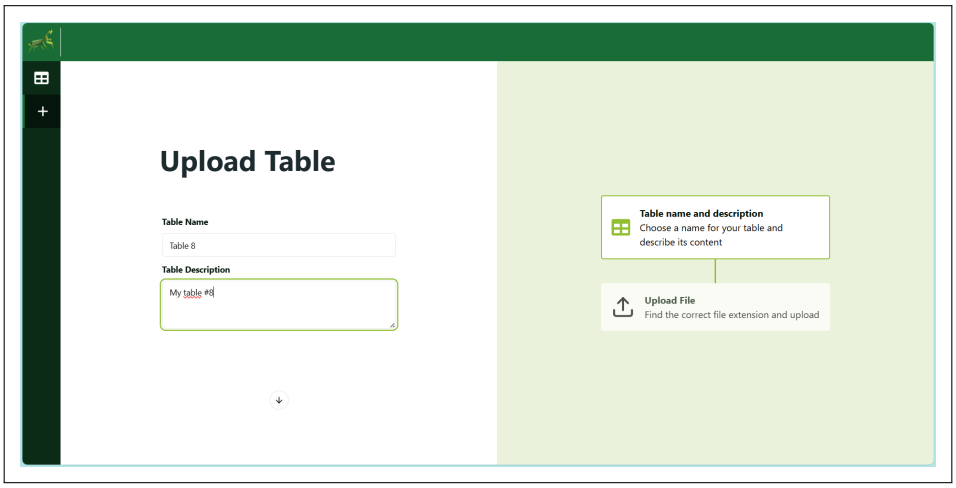

Figure 7 *MantisTable UI* Table import – Homepage.

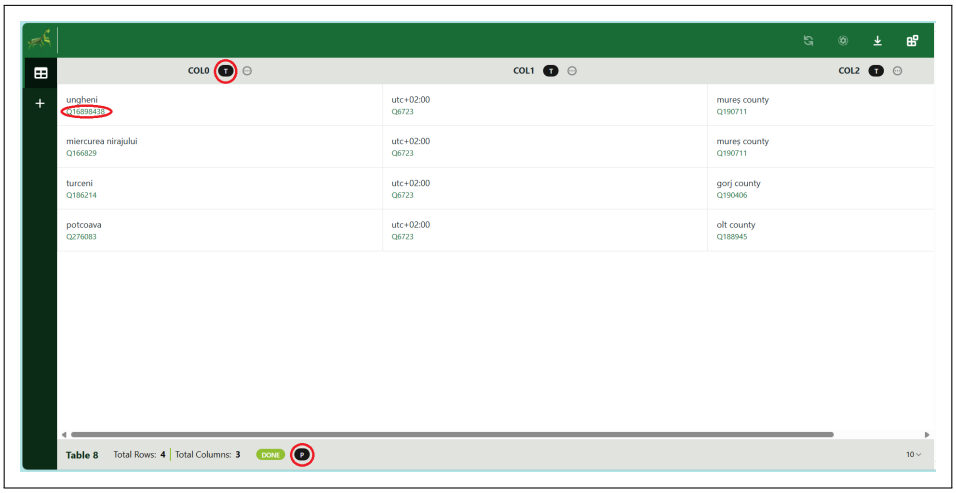

Figure 8 *MantisTable UI* View & Process Page. Highlight on some of the CTA, CEA and CPA result icons.

the number of rows and columns constituting the table.

Following the annotation step, it is possible from the same page to finalise the process by exporting the annotated table. Concrete examples of exported files are provided in the repository and archive to facilitate reproduction. *MantisTable UI* already allows the export in some formats (JSON, N3, trig, ttl, XML). Yet, as part of its plugin system, it offers the possibility to import specific customer-built export modules as .zip files, enabling the export of tables in a customised format to expand *MantisTable UI* default functionalities. This option is handled and performed via the "Manage Plugin" button (Figure 9), which allows importing the modules. Once the plugin is imported, it can be visualised and eventually selected among the available plugins (Figure 10), as will be described below.

PLUGIN SYSTEM

To enhance its capabilities, *MantisTable UI* is designed to be an extensible tool through the use of plugins. Three main types of plugins can be integrated: (i) export plugins, (ii) add-on plugins, and (iii) transformation plugins. A brief overview of their functionalities and use case examples are provided in what follows:

1. **Export Plugins**: they enable *MantisTable UI* to export data and annotations in user-defined formats.[29] Hence, they are responsible for the possibility of exporting tables to custom file formats (*i.e.*, TTL, JSON, XML, N3, TRIG, N-TRIPLES),[30] generating reports of table content or other document formats, or providing custom serialisation of table data for integration with other systems;
2. **Add-on Plugins**: they use semantic annotations to process table data, providing additional functionalities that can enhance the interpretability and presentation of the data.[31] For instance, the lexicalisation plugin,[32] named *MantisTableX* [7], is a powerful tool in improving the interpretability of tables, as it generates natural language descriptions of table content. Other use cases for add-on plugins include visualising table data in charts or graphs and

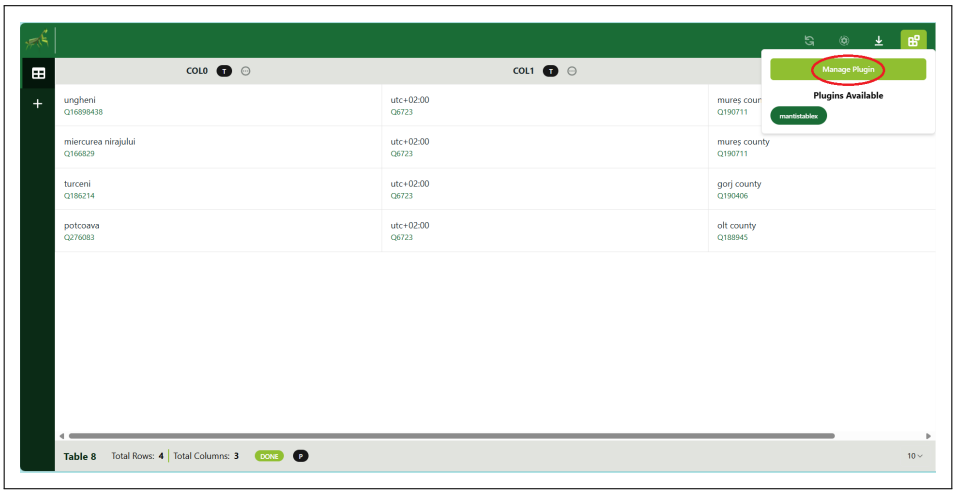

Figure 9 *MantisTable UI* – "Manage Plugin" option.

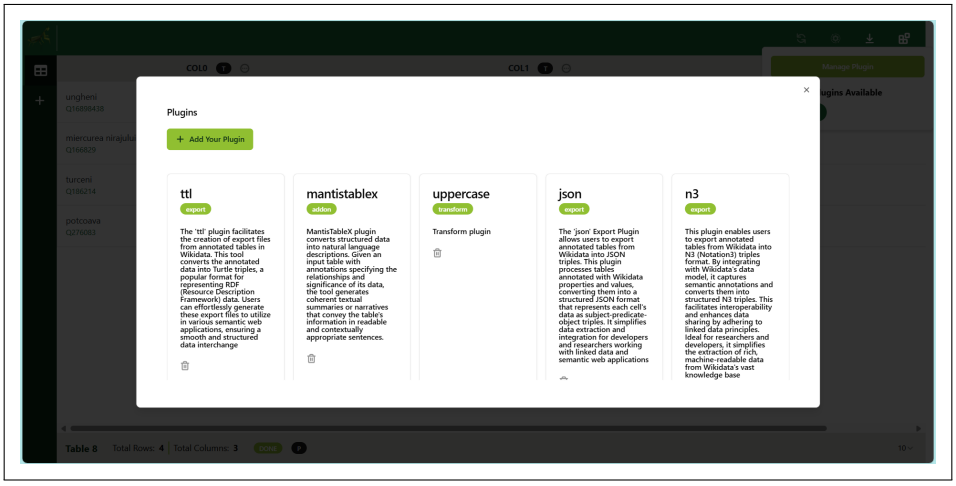

Figure 10 *MantisTable UI* – Plugins list.

integrating with external semantic services for enriched data processing;

3. **Transformation Plugins**: they allow complex data transformations to modify table structures and reshape data to meet specific analysis needs. Data transformation also includes the performance of computations for operations such as normalisation of data entries, as well as data cleaning, which is essential to improve data quality.[33] *MantisTable UI* uses *regular expressions*[34] to capture the type of the columns. In this way, plugin developers can constrain a plugin's applicability. Some transformation plugins are already integrated into *MantisTable UI*: *Uppercase*,[35] a simple plugin that allows the user to convert the contents of an entire column into uppercase, and *Reverse Geocoding*,[36] for translating latitude and longitude coordinates into a human-readable address for geolocalisation purposes. This plugin uses the OpenStreetMap reverse-geocoding service.[37]

To ensure security and stability, the plugins are executed in isolation within a Docker container. This approach guarantees that each plugin operates in an independent environment, preventing interference between plugins and the core *MantisTable UI* system. Each implemented plugin should allow configuration through environment variables or configuration files, and should be completed by clear documentation. A detailed guideline on the usage and implementation of the plugin system can be found in the *MantisTable UI* plugin documentation.[38]

QUALITY CONTROL

MantisTable UI has been tested at the following levels:

- **Unit testing**: Core modules of *MantisTable UI*, as well as the backend (FastAPI) and frontend (Next.js) components, have been covered with unit tests. These focused on validating individual functions, plugin interfaces, and API endpoints. Python unit tests were implemented using pytest,[39] while frontend units were tested using Jest;[40]
- **Integration testing**: Integration tests verified interactions among *MantisTable UI* components: frontend UI, backend API, and databases (PostgreSQL

and MongoDB). These tests ensured the full data flow worked as expected—from table import and annotation to plugin execution and export. Tests were executed within the Dockerised environment to reflect production-like deployment;

- **Efficiency and usability testing**: Usability tests were conducted to assess the UI, plugin loading experience, and overall workflow. Feedback from these sessions was used to improve UI clarity, error messages, and documentation. More details about this testing are provided below.

EFFICIENCY AND USABILITY TESTING

The efficiency and usability of the tool have been tested following a standard evaluation process consisting of administering to a group of participants, (i) a set of specific tasks, and (ii) a user experience questionnaire. Moreover, the *Think-aloud* method [21] was involved to gain real-time insights into users' interaction with the platform.

The execution of the specific activities was evaluated based on the "Task Success Rate". This metric identifies effectiveness, which is defined as the accuracy and completeness with which users can achieve specified goals in particular environments.[41]

The formula for "Task Success Rate" is shown below:

$$\text{Task Success Rate} = \frac{\text{Number of successfully completed tasks}}{\text{Total number of subtasks undertaken}} \times 100 \quad (1)$$

where a task was considered to be successfully completed if participants achieved its corresponding goal, suggesting successful navigation and interaction with the platform. The tasks were scored as follows: 0 for not completed tasks, 1 for tasks completed with difficulty, and 2 for tasks completed successfully. The obtained scores for the tasks were summed and divided by the maximum possible score. Successively, the result was converted into a percentage value.

The tasks which were assigned to the users are the following:

- **Task 1**: Importing a new table (provided in advance to the participants);
- **Task 2**: Annotating the imported table;
- **Task 3**: Identifying the column type annotation;
- **Task 4**: Exporting the table in .xml format;
- **Task 5**: Lexicalising the annotated table via the *MantisTableX* plugin;
- **Task 6**: Importing a custom plugin (provided in advance to the participants);

After completing the specific tasks, the System Usability Scale (SUS) questionnaire was employed as a validated tool to measure user satisfaction. The SUS is a user experience questionnaire comprising ten items, each scoring on an adjusted Likert scale ranging from 0 to 4. It is valued for its simplicity, reliability, and versatility in assessing various systems and interfaces [3]. The measured metric, "Satisfaction", was computed following [3]: the adjusted item scores were summed and then multiplied by 2.5. The total score ranges from 0 to 100, with higher values indicating higher satisfaction. Generally, a score below 68 is considered below average [12], a score exceeding 68 is regarded as good, whereas a score over 85 is deemed excellent [10].

The choice of this type of procedure to assess the efficiency and usability of *MantisTable UI* relies on the intention of providing both a quantitative evaluation, given by the Task Success Rate, as well as a qualitative evaluation, resulting from the *Think-aloud* method and the SUS questionnaire. Together, they offer a more comprehensive overview of the usability of the tool. Indeed, the latter evaluation method constitutes an essential resource for recommendations and general feedback, while the Task Success Rate metric returns a more objective assessment of the platform's usability.

Participants' recruitment consisted of the selection of a pool of 24 subjects in the age range 22–58. Out of the 24 subjects, 12 of them were "non-expert users", meaning that they had no experience in the domains of semantics, KG or data integration. The others were considered expert users, as they were aware of semantics and related topics and working with data from this field. Classification into one of the two categories was based on a preliminary questionnaire completed by each participant regarding their familiarity with the semantics field. Each participant worked individually, using their laptop to complete the usability testing, and had prior access to the online documentation.

Task Success Rate

As previously explained, the metric used for evaluating the effectiveness of *MantisTable UI* was the Task Success Rate score, which indicates whether a particular aspect of a product needs improvement in its usability.

Table 3 reports the task success scores, revealing varied levels of success across different tasks. While Task 1 (importing a new table) was easily completed by all the participants, Task 3 (identifying the column type annotation) registered the lowest success rate, amounting to 77.58%. The other four scores ranged between 81.03% and 94.82%. Overall, users quickly comprehended and completed most of the tasks. The main difficulty observed in completing Task 3 was the failure of participants to identify in the "P" icon the button revealing the column type annotation. A similar issue was encountered in Task 2 (annotating the imported table), denoted by the second lowest score (81.03%). Indeed,

#	TASK SUCCESS RATE
Task 1	100%
Task 2	81.03%
Task 3	77.58%
Task 4	94.82%
Task 5	86.2%
Task 6	91.37%

Table 3 Task Success Rates scored by the participants.

some users did not recognise the button responsible for the annotation task, while others struggled to identify it, eventually completing the task after random attempts of clicking the available icons. Despite these errors involved only non-expert users, they unveiled the need to modify the design, making the buttons' purpose explicit throughout the appearance of text while hovering over their icons.

For each participant, the evaluator kept track of the time employed to complete each task. A substantial difference in the completion time was measured between expert and non-expert users, with shorter completing intervals for expert users and reduced ranges of variation. Results are displayed in Figure 11. An exception is constituted by Task 1 (importing a new table) and Task 2 (annotating the imported table), whose relatively large ranges of variation for expert users, in line with the ranges of variation registered for non-expert users, could be attributed to the fact that participants were interacting for the first time with the platform and needed some time to become familiar with it, as demonstrated by the shorter completion times achieved in the successive tasks (as well as by the success rate scores). Moreover, it must be noted that Task 3 (identifying the column type annotation) has the shortest range of variation and completion times for expert users, indicating they quickly identified the button responsible for the CPA task. Task 4 (exporting the table in .xml format), *i.e.*, the one registering the second highest Success Rate, has overall the shortest completion times among non-expert users. Although all the users completed Task 1 (with a Success Rate of 100%), it took them longer compared to completing Task 4 (which registered a lower Success Rate), as they needed to gain familiarity with the platform.

User Experience Questionnaire and General Feedback

Following the testing sessions, participants provided valuable feedback on various aspects of the application. Overall, positive feedback was collected regarding the user-friendly interface and ease of navigation. Expert users particularly praised the platform's utility in executing STI processes. Constituting the only type of negative feedback, many of the participants highlighted the necessity of adding explanatory text when hovering some of the icons in the *View & Process* page, such as the icon to perform the annotation process, as it would reduce the time to execute for the first time this task and therefore improve usability. This suggestion already emerged as a consequence of the implementation of Task 2 and Task 3 and was also stressed by expert users despite their ease in completing Task 3.

Regarding the post-test administered SUS questionnaire, the average score of 71.82 (SD = 5.66) suggests the platform is perceived to be highly usable. The calculated scores ranged from 63.5 (satisfying usability) and 81 (good usability), indicating a consistent perceived usability among the 24 subjects. However, the lowest score denoted the necessity of refining certain design elements, as suggested by the feedback obtained through the *Think-aloud* method and the task completion. As a consequence, these problematic elements were successively fixed.

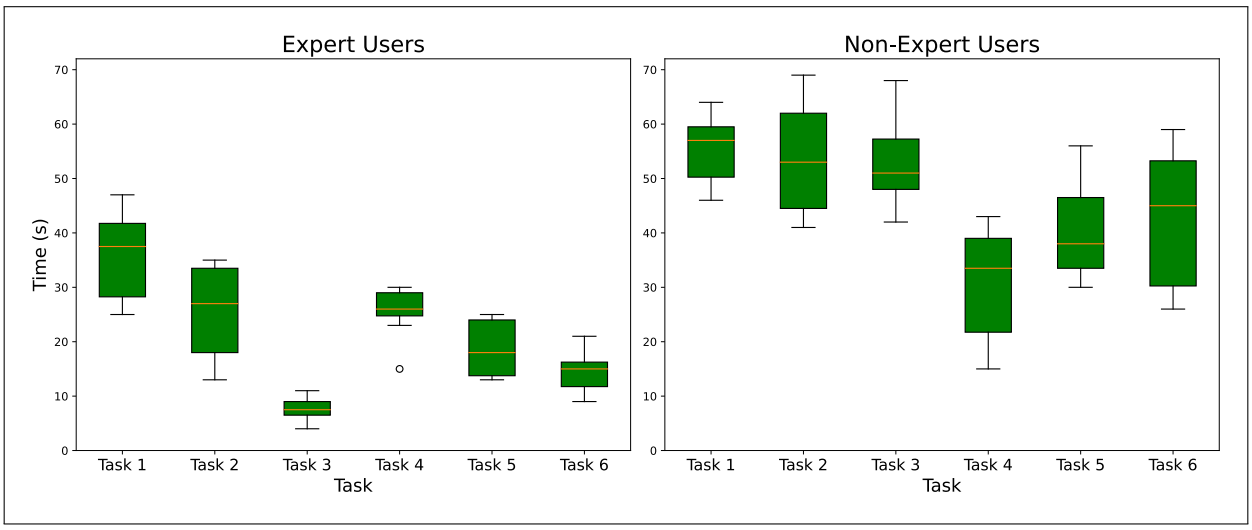

Figure 11 Difference in task completion time between expert and non-expert users.

(2) AVAILABILITY

OPERATING SYSTEM

MantisTable UI is compatible with Linux, macOS and Windows via Docker.

PROGRAMMING LANGUAGE

- ***MantisTable UI* Platform**: Typescript (Version: 5.4.2);
- **Plugin Backend API**: Python (Minimum version: 3.9).

ADDITIONAL SYSTEM REQUIREMENTS

- **Memory**: Minimum 4GB RAM;
- **Disk Space**: Minimum 2GB free space;
- **Processor**: Dual-core CPU;
- **Input Devices**: Keyboard, Mouse;
- **Output Devices**: Standard Monitor;
- **Docker Engine (minimum version 20.10)**;
- **Web browser (Chrome recommended)**.

DEPENDENCIES

- **Frontend Platform**: Next.js (v14 or later), Tailwind CSS (v3 or later), and PostgreSQL (v14 or later);
- **Plugin Backend**: FastAPI (v0.78 or later), MongoDB (v6.0 or later);
- **Processor**: Dual-core CPU;
- **System**: Docker Compose (v2.0 or later);
- **Plugin System**: Supports Python plugins loaded dynamically. Plugin interface specification defined in the backend documentation.

LIST OF CONTRIBUTORS

1. Fabio D'Adda – University of Milano-Bicocca;
2. Sara Nocco – University of Milano-Bicocca;
3. Marco Cremaschi – University of Milano-Bicocca.

SOFTWARE LOCATION

Archive

Name: Zenodo
Persistent identifier: https://doi.org/10.5281/zenodo.15482303
Licence: Creative Commons Attribution 4.0 International
Publisher: Fabio D'Adda
Version published: v1
Date published: 21/05/25

Code repository

Name: Github.
Persistent identifier: unimib-datai.github.io/mantistable-ui-docs
Licence: AGPL-3.0 license
Date published: 04/09/24

LANGUAGE

English

(3) REUSE POTENTIAL

This paper presents *MantisTable UI*, a novel web-based interface supporting the STI process and providing an intuitive GUI to assist users in executing all the steps provided by a STI approach. To address the significant gap in SOTA solutions, this tool was developed in response to the current lack of user-friendly interfaces for managing STI tasks. A detailed description of the independent modules constituting the *MantisTable UI* architecture is presented, highlighting the substantial advantages of scalability, flexibility, and easier maintenance of the architecture guaranteed by such a modular framework. The plugin system constitutes a powerful instrument to extend the default *MantisTable UI* features from data transformation to the export phase. The efficiency and usability tests demonstrated the expected ease of use of the tool, with overall positive scores on the administered usability tasks and appreciative feedback on the clarity and usefulness of the platform.

The release of *MantisTable UI* significantly impacts the field of STI as it constitutes the only tool that supports STI provided with a currently operating GUI. *MantisTable UI* is free, open, and accessible on the Web, making the entire STI process attainable to a broad range of users. The tool's plugin system further enhances its utility, expanding the functionalities in many data life cycle phases. The extensive usability demonstrates the uptake of our work beyond our research group, providing a valuable resource for diverse applications and encouraging widespread adoption within the research community.

Future developments of the platform involve implementing those requirements identified with the SOTA analysis, which have not been integrated into *MantisTable UI* version 1.0. These include the import of tables via API, the integration of annotation suggestions, and auto-complete support. Updates on the latest functionalities will be uploaded on the official *MantisTable UI* documentation website.

NOTES

1 unimib-datai.github.io/mantistable-ui-docs.
2 openrefine.org.
3 usc-isi-i2.github.io/karma.
4 alteryx.com/about-us/trifacta-is-now-alteryx-designer-cloud.
5 github.com/odalic.
6 bitbucket.org/disco_unimib/mantistable-tool/src/master.
7 datagraft.io.
8 github.com/IBCNServices/MAGIC.
9 github.com/I2Tunimib.
10 developer.orange.com/apis/table-annotation.
11 alteryx.com.
12 github.com/unimib-datAI/mantistable-ui/issues.

13 docker.com.
14 create.t3.gg.
15 nextjs.org.
16 typescriptlang.org.
17 tailwindcss.com.
18 trpc.io.
19 orm.drizzle.team.
20 fastapi.tiangolo.com.
21 postgresql.org.
22 mongodb.com.
23 github.com/unimib-datAI/mantistable-ui.
24 unimib-datai.github.io/mantistable-ui-docs.
25 unimib-datai.github.io/lamapi-docs/.
26 unimib-datai.github.io/s-elbat-docs/.
27 unimib-datai.github.io/mantistable-ui-docs/docs/getting-started/configuration#external-sti-services.
28 mantistable.datai.disco.unimib.it.
29 github.com/unimib-datAI/mtu-plugin-export-sample.
30 github.com/unimib-datAI/mtu-plugin-export.
31 github.com/unimib-datAI/mtu-plugin-addon-sample.
32 github.com/unimib-datAI/mtu-plugin-mantistablex.
33 github.com/unimib-datAI/mtu-plugin-tranformation-sample.
34 docs.python.org/3/library/re.html.
35 github.com/unimib-datAI/mtu-plugin-uppercase.
36 github.com/unimib-datAI/mtu-plugin-reverse-geocoding.
37 nominatim.openstreetmap.org/ui/.
38 unimib-datai.github.io/mantistable-ui-docs/docs/plug-in.
39 pypi.org/project/pytest.
40 jestjs.io.
41 International Organisation for Standardization. ISO 9241-11:2018 Ergonomics of Human-System Interaction – Part 11: Usability: Definitions and Concepts 2018.

FUNDING INFORMATION

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

COMPETING INTERESTS

The authors have no competing interests to declare.

AUTHOR CONTRIBUTIONS

Marco Cremaschi: Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.
Fabio D'Adda: Conceptualization, Formal analysis, Investigation, Methodology, Resources, Software.
Sara Nocco: Validation, Visualization, Writing – original draft, Writing – review & editing.

AUTHOR AFFILIATIONS

Marco Cremaschi orcid.org/0000-0001-7840-6228
Department of Informatics, Systems and Communication – DISCo, University of Milano-Bicocca, Milan, Italy

Fabio D'Adda orcid.org/0009-0006-0995-0661
Department of Informatics, Systems and Communication – DISCo, University of Milano-Bicocca, Milan, Italy

Sara Nocco orcid.org/0009-0000-6086-8213
Department of Informatics, Systems and Communication – DISCo, University of Milano-Bicocca, Milan, Italy

REFERENCES

1. **Avogadro R, Cremaschi M.** 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 (SemTab 2021), Vol. 3103 of CEUR Workshop Proceedings.* CEUR-WS.org; 2021. pp. 79–91.
2. **Avogadro R, D'Adda F, Cremaschi M.** Feature/vector entity retrieval and disambiguation techniques to create a supervised and unsupervised semantic table interpretation approach. *Knowledge-Based Systems.* 2024;304:112447. DOI: https://doi.org/10.1016/j.knosys.2024.112447
3. **Brooke J.** SUS: A 'quick and dirty' usability scale. In: Jordan PW, Thomas B, Weerdmeester BA, McClelland IL, editors. *Usability Evaluation in Industry.* London, UK: Taylor & Francis; 1996. pp. 189–194.
4. **Cremaschi M, Avogadro R, Barazzetti A, Chieregato D, Jiménez-Ruiz E.** 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), Vol. 2775 of CEUR Workshop Proceedings.* CEUR-WS.org, Cham; 2020. pp. 75–85.
5. **Cremaschi M, Avogadro R, Chieregato D.** 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), Vol. 2553 of CEUR Workshop Proceedings.* CEUR-WS.org; 2019. pp. 15–24.
6. **Cremaschi M, Avogadro R, Chieregato D.** 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 (SemTab@ISWC 2022), Vol. 3320 of CEUR Workshop Proceedings.* CEUR-WS.org; 2022. pp. 59–71.
7. **Cremaschi M, Barbato JA, Rula A, Palmonari M, Actis-Grosso R.** What really matters in a table? insights from a user study. In: *2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*; 2022. pp. 263–269. DOI: https://doi.org/10.1109/WI-IAT55865.2022.00045
8. **Cremaschi M, De Paoli F, Rula A, Spahiu B.** A fully automated approach to a complete semantic table

interpretation. *Future Generation Computer Systems*. 2020;112:478–500. DOI: https://doi.org/10.1016/j.future.2020.05.019

9. **Cutrona V, Ciavotta M, De Paoli F, Palmonari M.** Asia: a tool for assisted semantic interpretation and annotation of tabular data. In: *Proceedings of the ISWC 2019 Satellite Events, Vol. 2456 of CEUR Workshop Proceedings*. CEUR-WS.org; 2019. pp. 209–212.
10. **Ferreira JM, Acuña ST, Dieste O, Vegas S, Santos A, Rodríguez F, Juristo N.** Impact of usability mechanisms: An experiment on efficiency, effectiveness and user satisfaction. *Information and Software Technology*. 2020;117:106195. DOI: https://doi.org/10.1016/j.infsof.2019.106195
11. **Gupta S, Szekely P, Knoblock CA, Goel A, Taheriyan M, Muslea M.** Karma: A system for mapping structured sources into the semantic web. In: *The Semantic Web: ESWC 2012 Satellite Events, Vol. 7540 of Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg; 2015. pp. 430–434. DOI: https://doi.org/10.1007/978-3-662-46641-4_40
12. **Lewis JR, Sauro J.** Item benchmarks for the system usability scale. *Journal of Usability Studies*. 2018;13(3):158–167.
13. **Mazumdar S, Zhang Z.** Visualizing semantic table annotations with tableminer+. In: *Proceedings of the ISWC 2016 Posters & Demos Track, Vol. 1690 of CEUR Workshop Proceedings*. CEUR-WS.org; 2016. p. 88.
14. **Narducci F, Palmonari M, Semeraro G.** Cross-lingual link discovery with tr-esa. *Information Sciences*. 2017;394–395: 68–87. DOI: https://doi.org/10.1016/j.ins.2017.02.019
15. **Nguyen P, Yamada I, Kertkeidkachorn N, Ichise R, Takeda H.** Semtab 2021: Tabular data annotation with mtab tool. In: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2021), Vol. 3103 of CEUR Workshop Proceedings*. CEUR-WS.org; 2021. pp. 92–101.
16. **Pomp A, Paulus A, Jeschke S, Meisen T.** Eskape: Platform for enabling semantics in the continuously evolving internet of things. In: *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*. San Diego, CA, USA: IEEE Computer Society; 2017. pp. 262–263. DOI: https://doi.org/10.1109/ICSC.2017.45
17. **Ripamonti M, De Paoli F, Palmonari M.** *Semtui: a framework for the interactive semantic enrichment of tabular data*. arXiv preprint arXiv:2203.09521. 2022;abs/2203.09521.
18. **Roman D, Dimitrov M, Nikolov N, Putlier A, Sukhobok D, Elvesæter B, Berre A, Ye X, Simov A, Petkov Y.** Datagraft: Simplifying open data publishing. In: *The Semantic Web: ESWC 2016 Satellite Events, Vol. 9989 of Lecture Notes in Computer Science*. Cham, Switzerland: Springer International Publishing; 2016. pp. 101–106. DOI: https://doi.org/10.1007/978-3-319-47602-5_21
19. **Sarthou-Camy C, Jourdain G, Chabot Y, Monnin P, Deuzé F, Huynh V-P, Liu J, Labbé T, Troncy R.** Dagobah ui: A new hope for semantic table interpretation. In: *The Semantic Web: ESWC 2022 Satellite Events, Vol. 13384 of Lecture Notes in Computer Science*. Hersonissos, Greece: Springer International Publishing; 2022. pp. 107–111. DOI: https://doi.org/10.1007/978-3-031-11609-4_20
20. **Steenwinckel B, De Turck F, Ongenae F.** 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 ISWC 2021, Vol. 3103 of CEUR Workshop Proceedings*. CEUR-WS.org, Virtual event; 2021. pp. 68–78.
21. **Weichbroth P.** Usability testing of mobile applications: A methodological framework. *Applied Sciences*. 2024;14(5):1792. DOI: https://doi.org/10.3390/app14051792
22. **Zamini M, Reza H, Rabiei M.** A review of knowledge graph completion. *Information*. 2022;13(8). DOI: https://doi.org/10.3390/info13080396

TO CITE THIS ARTICLE:
Cremaschi M, D'Adda F, Nocco S. 2025 MantisTable UI: A Web Interface for Comprehensive Semantic Table Interpretation Curation and Management. *Journal of Open Research Software*, 13: 34. DOI: https://doi.org/10.5334/jors.583

Submitted: 21 May 2025 **Accepted:** 24 November 2025 **Published:** 11 December 2025