Easy-to-use interfaces for supporting the semantic annotation of web tables

Sara Bonfitto[1,*], Paolo Perlasca[1] and Marco Mesiti[1]

[1]Department of Computer Science (University of Milan), via Celoria 18, 20133 Milan (MI), Italy

Abstract

In the last few years, many approaches have been proposed for the semantic annotation of Web tables according to the concepts of a domain ontology and for the semantic description of the relationships existing among the identified concepts. However, these approaches are probabilistic and they are not always able to identify the correct semantic annotation because of the heterogeneity of the table contents, the eventual presence of mistakes, and the lack of standardization. The user intervention is thus required for checking the proposed annotations, correcting mistakes, and eventually providing new ones. In this paper, we propose different easy-to-use graphical facilities for supporting the user in this activity when dealing with web tables presenting a complex structure and syntactic and semantic mistakes. Different semantic annotation techniques can be integrated into the web application that produces results according to the data structures that are discussed in the paper. A usability analysis was conducted to assess the quality of the provided graphical tools.

Keywords

Table Understanding, GUIs for Web tables, Graphical representation of semantic description, Usability analysis

1. Introduction

Web tables [1] are essential sources of information that can be exploited for conducting different kinds of analysis and predictions. However, they are designed for being interpreted by humans, are heterogeneous, and do not follow any standard format or notation. Some tables, usually denoted *1 Dimensional tables* [2], are organized as relational tables where column values are homogeneous, while others are highly heterogeneous presenting columns with different types of information at different granularity levels. Moreover, errors can occur in their content which makes harder the characterization of their content and introduces inconsistencies. Extracting the semantics from these tables and making them machine-understandable knowledge is an interesting research field referred to as *table understanding* [2] that encompasses the identification and segmentation of the table from the source document, the discrimination of its cells' role (either the header or content cells), the structural organization of the header cells, and their interpretation.

For what concerns the semantic interpretation of the table content, many approaches take into account a knowledge graph (KG) and the corresponding conceptual model (usually expressed in terms of a domain ontology O) and face the problems of: i) *column type identification* (*CTI*), i.e. the identification of the type of each column in terms of concepts in O or simple types; and, ii) *relation discovery* (*RD*), i.e. the identification of relations in O that bind the concepts/columns occurring in the table. For the first problem, many methods have been proposed for annotating the table columns with simple types (e.g. [3, 4, 5, 6, 7, 8]) or classes and properties of an ontology (e.g. [9, 10, 11, 12, 13]). We also proposed an approach [14] based on the use of decision trees for the identification of column types of tables extracted from spreadsheets that might present erroneous values. For relation discovery, the identification of the direct relation between entities is the main focus (e.g. [15, 16, 17]). Only recently, the possibility of identifying relations passing through other entities has been considered [18, 19, 20, 21] (e.g. two actors can be related by means of the film in which they have played). In these approaches, a graph, reporting all possible plausible relations involving the concepts identified in the table, is generated. Then, the edges of the graph are weighted (either by considering the specificity of the relations in the ontology and/or considering the relation frequencies in KG) and a tree with minimal weight is extracted that better represents the relations among the table columns. Recent approaches for link prediction [22, 23], which exploit the embedding of the KG in a vector space, can be applied to web tables.

Even if the approaches proposed for facing these two problems are promising, the proposed annotations can contain errors and user intervention is required for their checking, for correcting mistakes, and in case, for providing new annotations. In this paper, different graphical facilities are proposed that can be exploited for the visual representation of the results of the prediction tasks and for supporting the user in easily checking and modifying

Dataplat'23: 2nd International Workshop on Data Platform Design, Management, and Optimization, March 28, 2023, Ioannina, Greece

*Corresponding author.

sara.bonfitto@unimi.it (S. Bonfitto); paolo.perlasca@unimi.it (P. Perlasca); marco.mesiti@unimi.it (M. Mesiti)

0000-0002-9883-5561 (S. Bonfitto); 0000-0001-6674-2822 (P. Perlasca); 0000-0001-5701-0080 (M. Mesiti)

the predicted annotations. For what concerns CTI, the proposed interfaces allow showing errors occurring in columns (i.e. values that do not adhere to the column type), identifying more than one annotation for the same column, annotating the string components with different ontology properties. For what concerns RD, we consider the possibility of identifying a semantic description (in the same spirit of [18, 19, 20, 21]) and propose graphical tools for completing the semantic description and changing concepts and properties automatically determined. A usability test has been conducted on the proposed visual interfaces with good appreciation from our volunteers.

By means of the data structures that our interfaces rely on, our web application can integrate different CTI and RD approaches. In the examples presented in the paper we refer to the CTI approach developed in [14] and the RD approach developed in [21]. However, other approaches can be easily integrated.

In the remainder, Section 2 introduces the data structures for tables, types, ontology, and semantic description that are exploited from our interfaces. Section 3 shows the interfaces developed in the context of CTI. Section 4 shows the interfaces developed for the result of RD and for their modification. Section 5 shows the usability test and the obtained results. Finally, concluding remarks are reported in Section 6.

2. Preliminaries

A web table T is a triple $\langle Col, Rows, Ann\rangle$, where Col denotes the list of column names $[col_1,\ldots,col_j,\ldots,col_m]$ (when available, otherwise the symbol ? is used to denote its absence), and $Rows = \{row_1,\ldots,row_n\}$ is the set of table rows (each row row_i, $1 \leq i \leq n$, is a list of values $row_i = [val_{i,1},\ldots,val_{i,j},\ldots,val_{i,m}]$, one value for each column identified in the column schema). Ann is a partial function that represents the type/annotation associated with each value and column name of T. The type annotation can be: a basic/domain-specific type, a mixed/union type, or the property p of a concept C of an Ontology O (denoted $\langle C, p\rangle$). Basic types include `integer`, `Boolean`, `decimal`, `date`, whereas domain-specific types can be, for example, `Social Security Number (SSN)`, `VAT`, `currency`, `email`, `province`, `zip code`. Mixed types are record-types associated with a set of patterns for extracting the record components from strings, and *union types* for representing the occurrence of different types of values in a column. A domain Ontology O contains a set of concepts $\mathcal{C} = \{C_1,\ldots,C_n\}$ and relationships $\mathcal{R} = \{(C_1, r, C_2)|C_1, C_2 \in \mathcal{C}, r \in R\}$, where R is the set of relation names. Concepts can be organized in an inheritance hierarchy: $C_1 \sqsubseteq C_2$ denotes that C_1 is sub concept of C_2. Each concept can have associated basic properties taken from a set $\mathcal{P} = \{(p_1, D_1),\ldots,(p_m.D_m)\}$, where D_i is the basic type of the values of property name p_i; the properties of a concept C include those specifically defined for C and those inherited from ancestors of C.

A semantic description for a table T is a graph SD representing the mapping between the columns of T and the "meta-instances" of the concepts in O. We talk about meta-instances instead of concepts of O because SD can contain different instances of the same concept, and we need to discriminate them. Formally, a semantic description for a table $T = \langle Col, Rows, Ann\rangle$ is a graph $SD = (U_{Cs}, U_T, E_R, E_T)$, where: U_{Cs} is a set of nodes representing meta-instances of the concepts in $\mathcal{C}$; $u_C^j \in U_{Cs}$ denotes a vertex corresponding to the j^{th} occurrence of the concept C; U_T is a set of nodes corresponding to the columns in T ($|U_T| \leq |Col|$); $E_R \subseteq U_{Cs} \times R \times U_{Cs}$ represents the relationships among concepts in U_{Cs}; $E_T \subseteq U_{Cs} \times P \times U_T$ denotes the properties associated with the columns of T.

3. Web Table Visualization with Column Type Annotation

Once one of the CTI approaches is applied, a table $T = \langle Col, Rows, Ann\rangle$ is generated and annotated with a set of types and possibly with pairs $\langle C, p\rangle$ of the Ontology O. In some cases, table columns that uniquely identify an instance of the concept (e.g. SSN of a person) can be discriminated from those that are simple characteristics of the identified concept (e.g. `gender` of a person). More than a single type can be used for annotating the values of a single column. Whenever the frequency of cells of a given type is above a given threshold, the type is added to the union type identified for the column. Conversely, if the frequency is below the threshold, the cell is considered an error. Moreover, CTI approaches can also extract annotations for sub-components of strings, thus a record type can be extracted according to a given pattern (introduced as mixed type in Section 2).

Sometimes CTI approaches are not able to identify the pair $\langle C, p\rangle$ for all the table columns. Columns that did not receive the annotation are named *unmatched* and need to be properly handled by our graphical interfaces.

In this section, the interfaces for showing T with the type annotations and for updating them are introduced.

3.1. Main interface

Figure 1 shows the main interface we have developed for showing a table T representing information about invoices for the payment of taxes. The invoice can be related to a person or a company (with the relations

#	Name/Company	Identfier	date of birth	Address	ZIP code	Country	tax	penalty
	mixed1, text	SSN/VAT	date,error	mixed1, text	ZIP, error	country	decimal	decimal
1	Peter Arnold	CE 06 91 43 C	14/01/1992	3293 Abbey Rd London	NW8 0AE	United Kingdom	5.67	113.32
5	Collier	14 - 5537560		4300 Maple Lane, Gaden	35901	Alabama		
6	James Green	MR 75 33 03 A	300579	343 Oxford Rd Reading	RG301AY	United Kingdom	2.67	53.21
7	Danielle Gray Green	NM 00 98 06 B	27/04/1988	Num. 310 of Church Rd - Bristol	BS5 8AJ	United Kingdom	3.6	72.11
8	Glover	64-8163968		2918 Green Road, Alexandria	22304	Vermont	1107	97.09

Figure 1: Extracted table with associated type annotations

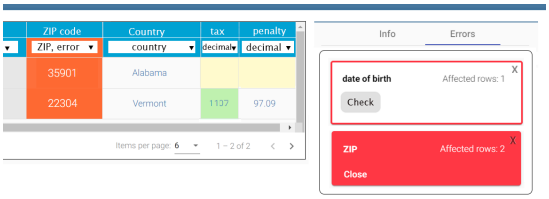

Figure 2: Error panel in the Web application

holder and owner). For each holder or owner, the associated address can be the residential address (in the case of people or the headquarter in the case of a company). The invoice contains taxes that the associated person or company should pay along with the penalty. Each cell and column is associated with the predicted type annotation. The first line reports the column schema and it is followed by a drop-down menu containing the inferred type annotations for each column. If more than one annotation is reported in a single column, this means that each annotation is a member of a union type, which is represented using different background colors for distinguishing their instances, such as in the second column of the table in Figure 1 where the occurrences of SSN and VAT are distinguished using two different shades of green. Similarly, the presence of mixed types is represented using different text colors for each component of the pattern. If a column presents a single annotation, the background remains white, if a value is missing, the cell background color is yellow. The usage of different colors can help the users in the process of checking the type predictions and the empty values (in some cases a value should be provided) and performing error corrections. A cell is considered an error when its type annotation is not compliant with the ones identified for the column; in this case, the cell background color and the column type background are marked in red. Facilities are provided for showing only rows presenting values in a column of a given type (for easily checking and correctly them).

When the number of rows contained in a document is high, it can be difficult to detect all the red cells; for this reason, we provide an error panel, on the right part of the screen, that summarizes the issues that need to be solved. An example of the panel is shown in Figure 2. When the user clicks the `check` button on one of the tabs in the panel, only the rows presenting the error are shown in the main interface. Once the errors are corrected, the corresponding panel tab is removed.

3.2. Modification of type annotations

Since the CTI approach can produce false positives or negatives (e.g. a ZIP code that has been labeled as Integer), specific GUIs have been developed for supporting the user in modifying the predicted annotations and easily applying the modification to the entire column or subset of cells. In the modification process, the user should be supported in the specification of pairs $\langle C, p \rangle$ instead of basic or domain-specific types that can be obtained through the CTI approach. Indeed, users can easily identify the domain concept to be associated with the column and thus improve the semantic description of the column. The user is also supported in the modification of the value of a cell when it contains errors. Consider for example the red cell in the `date of birth` column in Figure 1. It contains a value not compliant with the column type since the separators of the date are missing. In this case, the user can fix the mistake by directly editing the cell.

For performing bulk modifications on the values, a specific interface has been developed. For each column, the interface groups the occurrences of the identical string of the column, followed by the number of occurrences. Similar strings are clustered together relying on the edit distance and then reported together in the interface. In this way, it is easier to visually detect the errors and correct them with the aim of obtaining a homogeneous representation of the same kind of information. The user can edit the single value, and the proposed modification is applied to all the occurrences (note that when corrections lead to a value already present in the column, the two rows are collapsed).

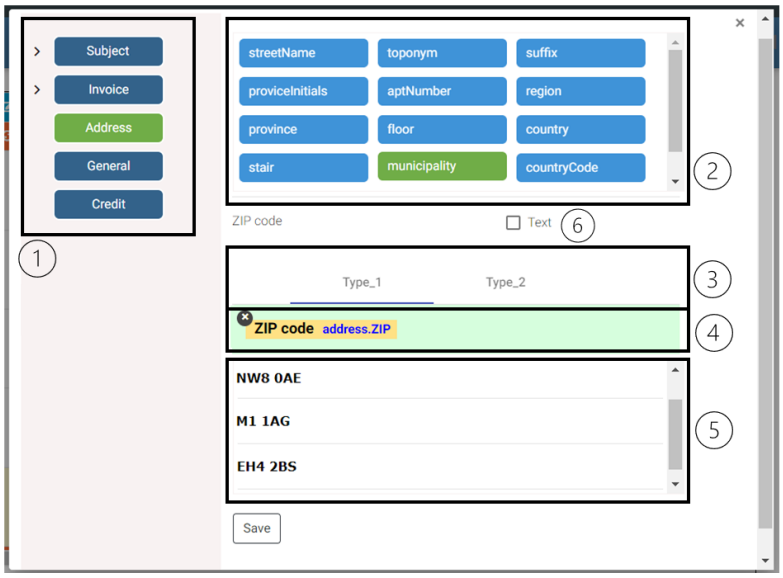

Figure 3: Data types editing

The interface in Figure 3 was developed for easily changing the type of an entire column or a subset of its values when column annotations are not correctly identified. The interface can be activated on a single cell, which becomes the current target of the modification, or on the entire column. Through this interface: i) the inferred data type can be modified into a new type, or into a $\langle C, p\rangle$ pair of the domain ontology; ii) a mixed type can be created or modified. The interface is organized into 5 areas. In (1) the hierarchy of concepts available in O is reported along with basic types (collected under the button `General`). The user can select one of the available concepts, and the corresponding properties are reported in (2) (when the `General` button is pressed, the basic types are reported). In (3), when the interface is activated on a target cell, a single type is reported (the value type), otherwise, the components of the union types specified for the column are reported. In this way, the type can be changed for each component of the union type. In (4) it is reported the target value or the column name and is highlighted with the current type for the column. The user can remove the current labeling (by clicking on the x button on the top left corner of the string) and apply a new $\langle C, p\rangle$ pair. In (5) values of the same type present in the column are reported and the user can select those to which the type modification should be applied (`all` is the default behaviour). The user can also decide to select the "text" checkbox reported in (6) to unify undesired union types (e.g. decimal and integer) and to treat the whole column as an instance of a single type. Then, the user can select the new type to be assigned to all values.

Example 1. *Two errors occur in the* `ZIP code` *column in Figure 1. Through the interface in Figure 3, the user highlights* `Type_2` *and substitutes the errors with* `Address.ZIP`. *Moreover,* `Type_1` *can be modified in* `Address.ZIP` *leading to a single column type.* □

The possibility of modifying types according to the concepts contained in the domain ontology can also introduce some issues that need to be properly managed.

Example 2. *Consider the column* `Name/Company` *in Figure 1 that the CTI approach has typed* `union(mixed_1, text, company)`, *where the structure associated with* `mixed_1` *is* `rec(name, surname)`. *The value* `Danielle Gray Greeen` *is of type* `text` *and can be changed with the mixed type* `mixed_2` *whose structure is* `rec(Person.name, Person.surname)`. *So, a more complex type than the one expected is generated.* □

To face this issue, a re-writing system based on rules [24] has been developed for the simplification of the type expression after the modifications applied by the user. The re-writing rules express correspondence between simple types and $\langle C, p\rangle$ pairs of the domain ontology occurring in the same table column. Once the re-writing rules are applied, the union-type components presenting the same structures are compacted. The union type is finally transformed into a simple type when a single component is identified. In the previous example, the application of the re-writing system leads to the type `union(mixed_1, company)`, where the record type associated with `mixed_1` is `rec(Person.name, Person.surname)`.

Figure 4: Pattern definition for column `Address`

3.3. Identification of a mixed type

Even if different approaches for the extraction of concepts from texts have been proposed [25], the identification of sub-components of a mixed type is quite hard to be handled automatically, especially when errors and variability in the pattern occur. Our interfaces support the user in the specification and modification of mixed types.

Example 3. *Consider the column* `address` *in Figure 1 and suppose the ML algorithm was able to identify the type* `mixed_1` *for some of its values. The others are marked of type* `text` *and we can see that they follow two specific patterns. These patterns can be manually detected on a single instance and applied to all the others.* □

The interface for the identification of mixed types is similar to the one presented in Figure 3 but it works on specific cell values that are reported in (4). Once the user has selected the property of a concept (in this case the `municipality` of an `Address`), he/she can highlight the part of the string of such a type. This behaviour applied to all the components will lead to the situation reported in the top part of Figure 4. In this way, we identify the terminal and non-terminal symbols that form our pattern. The non-labeled items are considered terminal symbols, while the labeled items are exploited for the generation of the pattern. Note that the `void` symbol can be applied for skipping variable parts of the string. Once the labeling is complete, the user can check if the generated pattern can be applied to other strings occurring in the same column that adhere to the same pattern (the instances in (5) that follow the pattern are highlighted). When the user tries to apply the labeling to other strings, the interface in Figure 5 is shown. The top part of the figure reports the labeled string, whereas the left panel reports strings that do not present the same pattern and the right panel contains the strings that have been re-written according to the identified pattern. The user can check the correctness of the applied pattern in the right panel and move to the left one those erroneously annotated. Moreover, he/she can take note of the strings in the left panel because they require the specification of different patterns or the identification of different types.

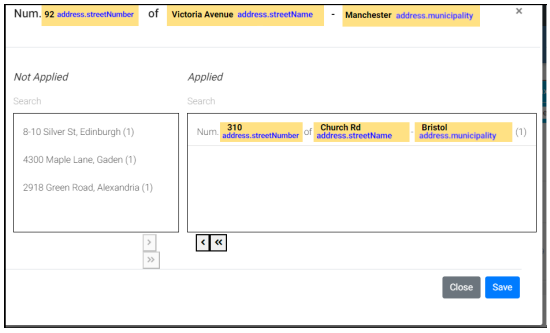

Figure 5: Application of a pattern to cells of the same column

In our example, the pattern can be applied only to two strings. For the remaining two strings of type text, the pattern in the bottom part of Figure 4 should be specified on one of them and applied to the other.

4. Relation discovery

As a result of the RD task, a graph SD can be generated. Its vertices correspond to the concepts that occur in the table or concepts induced by the presence of relationships with the table concepts. Moreover, graph edges are predicted by the adopted ML algorithm. Besides that, nodes representing the table columns are also included in the graph and are associated with the concepts by means of the corresponding properties.

Starting from SD, a graphical representation can be devised and reported in the main canvas of Figure 6 for being checked and approved by the user. The green nodes (representing meta-instances) are laid out in the top part of the canvas, whereas, light blue nodes (representing the table columns) are in the lower part of the canvas. Edges between instance nodes (i.e. E_R) and edges between instance nodes and terminal nodes (i.e. E_T) are represented in the same way (labeled arrows) because their meaning is easily understandable from the context. The label on the edges is the relation/property name. For each light blue node in the graph, a single incoming edge is present if the column has a single basic type (e.g. the `zip` column). Multiple incoming edges can be present when the light blue node represents a mixed or union-type column. For example, the `Address` column is of type mixed and three incoming edges are present (for representing the properties `streetName`, `streetNumber`, and `municipality`). Moreover, the `SSN/VAT` column is an example of column of type union and two incoming edges are present (one representing the `SSN` property of the instance-node $u^1_{\texttt{Person}}$ and the other representing the `VAT` identifier of the instance-node $u^1_{\texttt{Company}}$. We have decided to maintain this simplified representation for

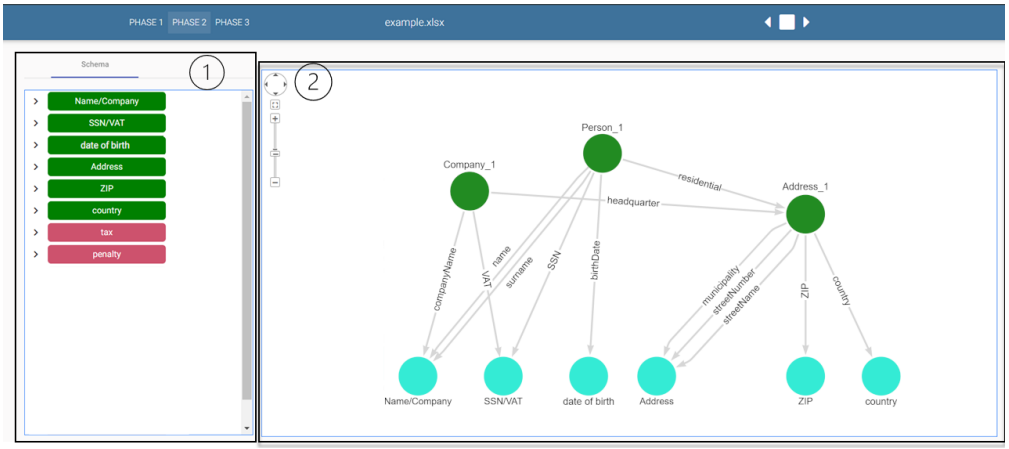

Figure 6: Interface for working with the graphical representation of the semantic description

keeping simple the illustration. Isolated nodes (i.e. terminal nodes without incoming edges) are not included in our graph representation.

The left panel (1) contains buttons corresponding to the table column. We exploit a double representation of the table columns (buttons in the left panel and light blue nodes in the central panel) because they are used for checking the correctness of the semantic values associated with each column and for adding missing annotations to the unmatched columns. Moreover, graphical edges are used for verifying the connections among the components and modifying/adding new ones.

The buttons in the left panel can be colored in two ways: *green*, i.e. the associated column has been already included in SD; *pink*, i.e. the associated column is not yet included in SD. By clicking on the arrow positioned on the left side of the button, it is possible to show the data type associated with that column (single type or union of types). By right-clicking on the button itself it is possible to specify a new pair $\langle C, p \rangle$ of the domain ontology for each data type of the column.

In the remainder, we discuss the operations that can be invoked on the two parts of the interface.

4.1. Visual operations on table columns

The following operations can be invoked on the table columns reported in the left panel for the correction of errors or in the definition of new nodes:

1. *Association of properties.* It allows the specification of properties to unmatched columns.
2. *Modification of properties.* It allows changing the current association of properties for a column.
3. *Removal of properties.* It removes the semantic concept associated with the column.

Operation 1 can be invoked on unmatched columns (i.e. pink buttons) and used for including them in SD in two steps. First, the identification of the properties that represent the column content in the ontology concepts is specified. Then, the instance nodes in SD (or new nodes that need to be added in SD), to which the properties can be associated, must be defined.

Example 4. *Consider the unmatched* `tax` *column in Figure 6. When Operation 1 is invoked on it, an interface similar to the one in Figure 3 is shown. In this case, the property* `taxes` *associated with the concept* `Invoice` *on the left bar is used for annotating the entire column. At the end of the operation, since no node in SD represents an instance of* `Invoice`*, the node $u^1_{\texttt{Invoice}}$ is introduced in SD with the node u_T for representing the table column* `tax`*. The edge $(u^1_{\texttt{Invoice}}, \texttt{taxes}, u_T)$ is included in E_T.* □

Whenever the chosen concept is already present in SD, an interface is shown to the user for deciding if the identified property should be associated with one of the meta-nodes in SD or a new one should be included. In this way, it is possible to distinguish the presence of different instances of the same concept.

Example 5. *Consider the situation of the previous example, and suppose that the table column* `penalty` *is now semantically annotated with* `Invoice.penalty`*. Since node $u^1_{\texttt{Invoice}}$ is already included in SD, a panel is shown to the user for deciding if the property should be associated with $u^1_{\texttt{Invoice}}$ or a new meta-instance should be created.* □

Regardless of the number of instances, after the introduction of a new concept, the system identifies the relations existing between the newly inserted element and the other concepts in the ontology. If a single relation is present, it is automatically added to SD.

Figure 7: Link definition among meta-instances

Table 1
Operations on the graphical representation of SD

#	Operation	On	Description
1	edit property	node	the properties of a blue node is modified
2	delete	node	a node is removed
3	insert relation	node	a new relation between nodes is inserted
4	update	edge	source or destination of a node is modified
5	delete	edge	an edge is removed

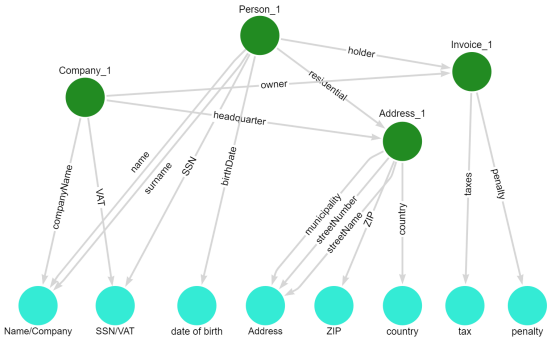

Figure 8: Final SD

Operation 2 is used for changing the already associated semantic annotation to a column (i.e. it is invoked on a green button). Besides changing the semantic annotation, this operation also allows changing the instance node to which the properties are associated (if needed). By invoking Operation 3 on a green button, the existing semantic annotation is removed along with the corresponding nodes in the graphical representation.

4.2. Visual Operations on SD

Table 1 shows the operations that can be executed on the graphical representation of SD. Some of them (1 and 2) can be invoked on light blue nodes and produce the same effect as the corresponding operations that can be applied to the buttons on the left sidebar. Operation 3 can be invoked on an instance node and allows the introduction of a new link with another instance node. The inserted links must be coherent with the domain ontology so that, for each pair of nodes, only existing relations in the correct direction can be added.

Example 6. *Consider the unmatched columns* `tax` *and* `penalty` *that have been associated with the instance node* $u^1_{\texttt{Invoice}}$. *This instance node should be linked with its holder or its owner (that can be a person or a company). These bindings are realized by means of the interface in Figure 7. The interface shows the lists of relation names (ingoing and outgoing) that can be exploited for the nodes of this concept by taking into account the instance nodes in the current SD and the constraints of the domain ontology. The user can select the correct relation and insert it in the graphical representation of SD that is updated accordingly. In this case, the interface is used two times for including two links (relation* `holder` *and* `owner`*) for representing the relation with the person and the company.* □

Operation 4 of Table 1 can be invoked on a node of the graphical representation of SD with the aim of modifying the name of the relation between two nodes or one of the nodes connected by the link. Finally, Operation 5 allows the deletion of an edge occurring in SD.

Table 2
Tasks identified for the usability test

#	goal	time	success	failure
1	mixed type	7 min	Definition of a mixed type and application of the labeling to other strings through the "apply" function	Lack of the pattern definition or application of the new procedure every time
2	errors	5 min	Detection and correction of the errors on values/types through the error panel	The errors are not corrected and the error panel is not exploited
3	bulk editing	3 min	Rows are updated in a single operation	Rows are updated one at a time
4	management of unmatched columns	7 min	The user specifies a concept and a property for each unmatched column.	One or more columns have not been associated with a concept and property of the domain ontology
5	connections among concepts	5 min	The user identifies the correctness of the existing links, adds the missing ones and modifies the wrong ones	The final SD is not complete or the links are wrong
6	new instance	4 min	the user is able to define a new instance for a concept	the user uses the same instance for multiple occurrences

Figure 8 shows the final SD that can be realized by means of our tool that describes the different kinds of data that can be extracted from our running example. This semantic description can then be used for translating the table content as RDF triples.

5. Experimental results

We organized a usability test of the Web application. The aim of this test is to evaluate if the users can smoothly interact with the application and use the provided tools, what level of knowledge in computer science is needed, and check the existence of critical aspects that should be fixed or improvements to be applied. This test is composed of three parts: first, the user watches a video that introduces him/her to the problem and shows the system usage. Then, some tasks are assigned to be carried out on specific files. Finally, the user fills out a questionnaire about his experience with the system, containing: i) personal information (age, gender, level of instruction) and technical abilities (computer skills in general, knowledge of operating systems, skills in the use of spreadsheets, ...); and, ii) users' opinions about the assigned tasks and their complexity iii) users' opinions about the functionalities of the proposed tool. The questions are rated using a Likert scale (from "strongly disagree" to "strongly agree").

We selected 20 participants, 12 males and 8 females, 60% of them were between 21 and 23 years old, 20% between 24 and 26 years old and the remaining ones were more than 26. Most of the users were recruited among personnel and students of the department of computer science of the University of Milan and therefore they have good technical skills. However, they are not involved in this project and they have little knowledge of the domain. Only a small part of the participants (50%) feels confident with Excel. Most of the students are currently attending their bachelor's degree, therefore they have only a high school diploma. Users have an average knowledge of different operative systems and use a computer or a laptop mostly for working or studying.

Table 2 reports the tasks that we have identified for checking the main functionalities of our system. Each task requires the processing of a spreadsheet that is specifically created for the purpose of the task and whose content can be easily understood also by non-expert of the domain. Specifically, two spreadsheets have been designed for pointing out the issues that each task was intended to address. Even if these spreadsheets correspond to real documents of our domain, their content has been anonymized for preserving user privacy. For each task, Table 2 reports the main goal, the time required for completing the task and when the task can be considered successfully completed or completely a failure. Tasks 1, 2, and 3 are used for evaluating the usability of interfaces developed for CTI, while the remaining tasks are used for evaluating the interfaces in handling the result of RD.

Almost all the volunteers (70%) were able to complete the assigned tasks within the specified time limits. The others would have been able to complete the task with additional time. A good fraction (70% of the users) thought that the assigned tasks were easy and enough intuitive.

For *task 1*, most of the individuals (85%) were able to specify a mixed type through the interface. All of them used the "apply" button to label all the mixed types in a single column. The main reason for the failure of this task was the choice of the wrong interface (they selected the interface for the modification of column type instead of the one for modifying the cell type).

For *task 2*, 75% of the individuals used the error panel and the general impression about its usefulness is very positive (from partially to strongly agree). The users that

did not exploit the error panel, tried to increase the size of table pagination to identify the errors. In these cases, the identification and correction of the errors required more time. Concerning this task, only 28% of users had trouble in distinguishing errors occurring on the data type (i.e. the component of a union type was not identified by the used CTI approach) from errors occurring on the data (i.e. a date is written without separators).

For *task 3*, most of the individuals (86%) were able to use the bulk editing functionality and all of them thought it speeds up the editing process. The remaining part did not notice the error occurring within the data (usually an additional letter in the name of a city) and they corrected it by editing the data type.

For *task 4* and *task 5*, 90% of the individuals completed the job in just 10 minutes. Only 10% of them had some trouble remembering the procedures to complete *task 5*.

For *task 6*, 20% of the users needed to watch again the training videos to complete the assignment correctly. The additional time required for watching videos were not counted in the total time of the task completion. The fact that all users, possibly after watching again the training video, have completed the tasks correctly highlights a possible difficulty for a novice user to learn the various procedures rather than apply them.

We tested user satisfaction in using the developed interfaces to support users in solving both CTI and RD issues. For the interfaces developed for CTI, 95% of the users agreed that the application is easy-to-use and intuitive and 85% of them declared that they did not have problems during the error correction process. The greater difficulties were related to the understanding of the specific domain; most of the users did not know the meaning of the concepts of the domain ontology and tried to identify the most suitable one. Moreover, the application provides a lot of functionalities and the user needs time to gain confidence in the system.

For the interfaces developed for RD, 85% of the users agreed that the interfaces are easy-to-use and intuitive whereas the remaining ones expressed a neutral position. The greatest uncertainties concerned the application of the operations on the graphical representation of SD. In particular, a few users had difficulty in recognizing or applying operations such as the insertion of new concepts or the insertion or removal of links between concepts. Half of the users declared that they had to remove an edge because it was not correct. Only 16% of the users could not connect all the graph nodes because they did not have enough knowledge about the domain (e.g. they did not know that a company can be the invoice holder).

In conclusion, the usability test suggests that although some aspects of the application could be improved, for example by adding contextual help possibly supported by short videos, the overall opinion is that the system is intuitive and easy to use.

6. Concluding remarks

In this paper we have discussed different user interfaces that can be exploited after the application of CTI and RD approaches for correcting the automatic predicted annotations and thus improving the semantic description of web tables. The developed interfaces allow, in many cases, the specification of a single modification and its propagation to other values in the same column that follows the same type or the same pattern. Once the semantic description has been generated and validated, it can be exploited for the translation of the table content in a KG representation, thus obtaining a meaningful representation of the table content.

The problem of supporting the user in the interpretation of table content was initially faced in Karma [18]. Our approach deeply extends this work by considering a more sophisticated data model both for the CTI and RD interfaces. Our semantic description allows the management of tuples of different types that need a different knowledge representation. Moreover, interfaces for the semantic labeling of columns and for extracting patterns for strings are new contributions of this paper.

A usability test has been run on the graphical interfaces for assessing their facility of use. Our results show that almost all users believe the application is easy-to-use and intuitive. Some more efforts should be devoted to improving the interfaces for handling the semantic description and for showing the results of the modifications on the table data. We are currently working on providing further facilities for supporting the user in this activity.

The work discussed in this paper can be extended in several directions. Even if we have focused on developing graphical interfaces for supporting the CTI and RD tasks, also entity linking approaches [26] can be used for table understanding and specific interfaces can be included in our system for their management. Moreover, once the semantic description is obtained, it can be exploited for the creation of KGs reporting the table content [21]. Specific interfaces can be also developed for supporting the user in obtaining this result and for the management of duplication and for fusing together alternative representations of the same entity. We would like also to collect the user modifications on the automatically generated annotations provided by CTI and RD approaches and use them for tuning the underline approaches. Finally, we would like to use the proposed interfaces for the construction of biological knowledge graphs [27].

Acknowledgments

This research was supported by the "National Center for Gene Therapy and Drugs based on RNA Technology", PNRR-NextGenerationEU program [G43C22001320007].

References

[1] M. J. Cafarella, et al., Webtables: Exploring the power of tables on the web, Proc. VLDB. 1 (2008) 538–549. doi:10.14778/1453856.1453916.

[2] S. Bonfitto, E. Casiraghi, M. Mesiti, Table understanding approaches for extracting knowledge from heterogeneous tables, WIREs Data Mining Knowl. Discov. 11 (2021). doi:10.1002/widm.1407.

[3] S. Kandel, et al., Wrangler: Interactive visual specification of data transformation scripts, in: ACM Human Factors in Computing Systems (CHI), 2011, p. 3363–3372. doi:10.1145/1978942.1979444.

[4] Trifacta, Wrangler, 2020. www.trifacta.com/.

[5] Google, Openrefine: A free, open source, powerful tool for working with messy data, 2020. Https://openrefine.org/.

[6] I. Valera, Z. Ghahramani, Automatic discovery of the statistical types of variables in a dataset, in: Proc. of Machine Learning Research, volume 70, 2017, pp. 3521–3529.

[7] T. Ceritli, C. K. I. Williams, J. Geddes, ptype: probabilistic type inference, Data Mining and Knowledge Discovery 34 (2020) 870–904. doi:10.1007/s10618-020-00680-1.

[8] Y. Yang, F. Abdelhédi, J. Darmont, F. Ravat, O. Teste, Automatic machine learning-based olap measure detection for tabular data, in: Big Data Analytics and Knowledge Discovery, Springer, 2022, pp. 173–188. doi:10.1007/978-3-031-12670-3_15.

[9] M. Pham, et al., Semantic labeling: A domain-independent approach, in: The Semantic Web Conference, Springer, Germany, 2016, pp. 446–462.

[10] N. Rümmele, Y. Tyshetskiy, A. Collins, Evaluating approaches for supervised semantic labeling, in: Workshop on Linked Data on the Web, volume 2073 of *CEUR*, Lyon, France, 2018, pp. 30–40.

[11] J. Chen, E. Jimenez-Ruiz, I. Horrocks, C. Sutton, Colnet: Embedding the semantics of web tables for column type prediction, in: Proc. of AAAI Conf. on Artificial Intelligence, volume 33, 2019, pp. 29–36. doi:10.1609/aaai.v33i01.330129.

[12] M. Hulsebos, et al., Sherlock: A deep learning approach to semantic data type detection, in: SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining, ACM, 2019, p. 1500–1508.

[13] D. Zhang, et al., Sato: Contextual semantic type detection in tables, Proc. VLDB. 13 (2020) 1835–1848. doi:10.14778/3407790.3407793.

[14] S. Bonfitto, et al., Semi-automatic column type inference for CSV table understanding, in: Proc. of 47th Int'l Conf. on Current Trends in Theory and Practice of Computer Science, SOFSEM, volume 12607 of *LNCS*, Springer, Bolzano, Italy, 2021, pp. 535–549. doi:10.1007/978-3-030-67731-2_39.

[15] G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, Proc. VLDB 3 (2010) 1338–1347. doi:10.14778/1920841.1921005.

[16] P. Venetis, et al., Recovering semantics of tables on the web, Proc. VLDB. 4 (2011) 528–538. doi:10.14778/2002938.2002939.

[17] V. Mulwad, T. Finin, A. Joshi, Semantic message passing for generating linked data from tables, in: The Semantic Web Conference, Springer, Berlin, Heidelberg, 2013, pp. 363–378.

[18] M. Taheriyan, C. A. Knoblock, P. Szekely, J. L. Ambite, Learning the semantics of structured data sources, Journal of Web Semantics 37-38 (2016) 152 – 169. doi:10.1016/j.websem.2015.12.003.

[19] G. Futia, A. Vetrò, J. C. De Martin, Semi: A semantic modeling machine to build knowledge graphs with graph neural networks, SoftwareX 12 (2020) 100516.

[20] B. Vu, C. Knoblock, J. Pujara, Learning semantic models of data sources using probabilistic graphical models, in: The WWW Conf., ACM, 2019, p. 1944–1953. doi:10.1145/3308558.3313711.

[21] S. Bonfitto, et al., A semantic approach for constructing knowledge graphs extracted from tables, Tech. Rep, Dept. Computer Science, Uni. of Milano, 2023.

[22] A. Kumar, et al., Link prediction techniques, applications, and performance: A survey, Physica A: Statistical Mechanics and Its Applications 553 (2020). doi:10.1016/j.physa.2020.124289.

[23] M. Schlichtkrull, et al., Modeling relational data with graph convolutional networks, 2017. URL: doi:10.48550/ARXIV.1703.06103.

[24] N. Dershowitz, D. A. Plaisted, Chapter 9 - rewriting, Handbook of Automated Reasoning North-Holland, 2001, pp. 535–610. doi:10.1016/B978-044450813-3/50011-4.

[25] F. Gutierrez, et al., A hybrid ontology-based information extraction system, Journal of Information Science 42 (2016) 798–820. doi:10.1177/0165551515610989.

[26] S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: A survey, ACM Trans. Intell. Syst. Technol. 11 (2020). doi:10.1145/3372117.
Marco Mesiti, Marco Notaro, Alessandro Petrini, Alex Patak, Antonio Puertas-Gallardo, Alberto Paccanaro, Giorgio Valentini, Elena Casiraghi

[27] J. Gliozzo, et al., Heterogeneous data integration methods for patient similarity networks, Briefings in Bioinformatics 23 (2022) doi:10.1093/bib/bbac207.