A Web Tool for the Semantic Integration of Heterogeneous and Complex Spreadsheet Tables

Sara Bonfitto[1], Luca Cappelletti[1], Elena Casiraghi[1], Paolo Perlasca[1], Fabrizio Trovato[2], Giorgio Valentini[1] and Marco Mesiti[1]

[1]*Dep. of Computer Science, Università di Milano, Via Celoria 18, 20133 Milano*
[2]*Area s.r.l., Via Torino 10/B, 12084 Mondovì*

Abstract
The acquisition and integration of data contained in spreadsheet tables is a complex task because they do not impose any regular structure on the organization of the data, or constraints on valid values. Moreover, mistakes can occur due to the passage from a format to another one or misspelled words in the original sources. The automatic extraction of their content, interpretation and integration is thus a complex task. In this paper, we outline the characteristics of a semi-automatic, interactive tool conceived for creating a knowledge base by extracting semantic information from heterogeneous spreadsheets.

Keywords
Heterogeneous Spreadsheet Tables, Semantic Table Interpretation, User Interfaces, Machine Learning

1. Introduction

The table understanding problem consists in the meaningful extraction of information from tabular data [1] contained in different kinds of files (HTML, pdf, csv, spreadsheets). As reported in a recent survey [2], many approaches have been proposed for the *localization* of the table(s) within a file, the *segmentation* of its cells, each with an eventually different size, the automatic identification of their *functional* role, in terms of either data cell or access key (i.e. the column headers or later stubs) and of their *structural* role (i.e. the identification of hierarchies on the access keys), and the *interpretation* of the table content. The interpretation of a table usually regards the transformation of its content in terms of well-known data models (like the relational model or RDF). Nowadays, different approaches for table understanding have been proposed that exploit machine learning (ML) techniques for inferring the table metadata by exploiting domain Ontologies and (automatically or manually) tagged knowledge bases (e.g. [3, 4, 5, 6]).

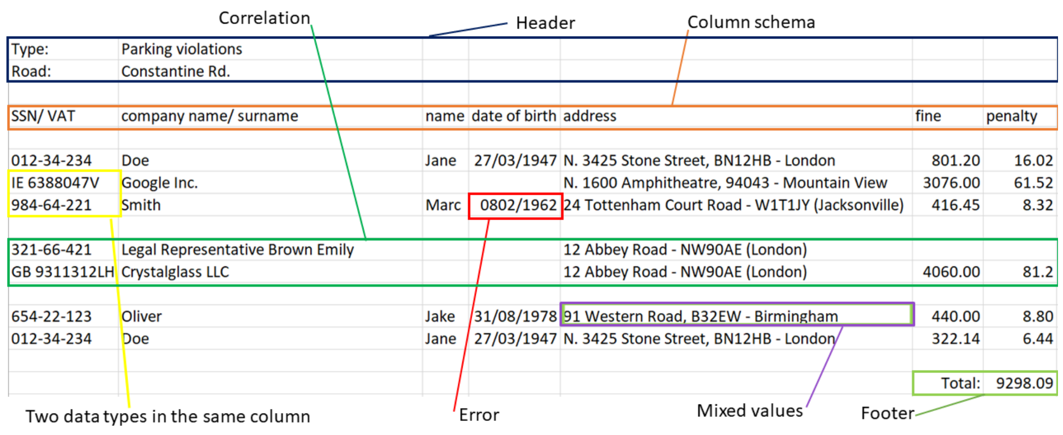
We have faced the table understanding problem in the context of a research project with an Italian debt collection agency. The agency daily receives spreadsheets in different formats (`csv`, `xls`, `xlsx`) from local authorities (e.g. municipalities, tax agency), each containing batches

SEBD 2021: The 29th Italian Symposium on Advanced Database Systems, September 5-9, 2021, Pizzo Calabro (VV), Italy
sara.bonfitto@unimi.it (S. Bonfitto); luca.cappelletti@unimi.it (L. Cappelletti); elena.casiraghi@unimi.it (E. Casiraghi); paolo.perlasca@unimi.it (P. Perlasca); fabrizio.trovato@team1994.it (F. Trovato); giorgio.valentini@unimi.it (G. Valentini); marco.mesiti@unimi.it (M. Mesiti)
0000-0002-9883-5561 (S. Bonfitto); 0000-0002-1269-2038 (L. Cappelletti); 0000-0003-2024-7572 (E. Casiraghi); 0000-0001-6674-2822 (P. Perlasca); 0000-0002-5694-3919 (G. Valentini); 0000-0001-5701-0080 (M. Mesiti)

with thousand of invoices to be rescued. These spreadsheets are large, heterogeneous, and do not follow any standard format or notation. Moreover, the labels used for the column headers, if not missing, do not belong to any pre-defined dataset and sometimes are not informative of the column content. Finally, spreadsheets often contain mistakes that need to be fixed. In this context, usual approaches for table understanding are not successful due to the following two reasons. First, they mainly rely on the homogeneity of the types of data contained in the columns of the identified table (i.e. columns containing only person names, or only date of births), and therefore exploit rules maximizing coherence in their cell segmentation methods, as well as in their structural and functional analysis algorithms; this is impossible in our scenario, where heterogeneity often characterizes both rows, columns, and even cells. In practice, it often happens that both rows and columns contain different types of information (e.g. the name of a person and the name of a company in the same column) and the information of a single invoice is contained in contiguous table rows. Moreover, each cell often contains complex string contents, which express multiple information (e.g. street number, street name, ZIP code and city can be extracted from the string "N. 3425 Stone Street, BN12HB - London"). Finally, current approaches rarely consider the presence of syntactic mistakes (e.g. the SSN number that does not fit the length constraint, or a misspelled city name) and semantic mistakes (e.g. a city in the wrong region, or the amount of a debt that does not match its detailed voices).

We believe that the development of an automatic technique for addressing all the mentioned issues, even if exploiting sophisticated ML techniques, would be unpractical. Therefore, we introduce a semi-automatic approach that combines data management approaches with ML techniques and intelligent user-interfaces to reduce the human effort in the extraction, cleaning and semantic characterization of the information contained in the invoices and their integration in a single and common knowledge base. Our human-in-the-loop approach heavily relies on the user-interaction to tune the prediction system depending on the feedback obtained while processing new spreadsheets. Easy-to-use graphical interfaces are thus fundamental for correcting mistakes and improving the overall performance of the system. To reach this goal, we propose the adoption of a three-phase approach. Phase I (Section 2) is responsible for the spreadsheet cleaning, the identification of the column types and the syntactic error correction. Phase II (Section 3) aims at creating a semantic characterization of the table content to be extracted from the spreadsheet and relies on the use of a domain Ontology (DO) and, when possible, a Knowledge Base (KB). Phase III (Section 4) relies on the construction of a KB containing the extracted information.

In the paper we use as running example the spreadsheet in Fig. 1 reporting traffic tickets which is organized in three parts: title lines, a table of data, and footer notes. Below the heading rows, a row contains the access keys to the columns (*column schema*). Column SSN/VAT contains the code identifying either individual or companies (it is an example of heterogeneous column containing values of two types). On the other side, the address column contains complex (string) contents expressing multiple information; a proper string parser must be defined to correctly extract all such information. Blanks and semi-blank rows can occur in the main table. Semi-blank rows usually contain totals or aggregated data. While some blank rows are sometimes used for aesthetic reasons, some others for delimiting correlated rows. This is the case of the two rows marked with a green border that represent an invoice with its the legal representative that must pay the invoice. Last but not least, different kinds of errors can occur in the spreadsheet.


Correlation
Header
Column schema
Two data types in the same column
Error
Mixed values
Footer

Type:	Parking violations					
Road:	Constantine Rd.					
SSN/ VAT	company name/ surname	name	date of birth	address	fine	penalty
012-34-234	Doe	Jane	27/03/1947	N. 3425 Stone Street, BN12HB - London	801.20	16.02
IE 6388047V	Google Inc.			N. 1600 Amphitheatre, 94043 - Mountain View	3076.00	61.52
984-64-221	Smith	Marc	0802/1962	24 Tottenham Court Road - W1T1JY (Jacksonville)	416.45	8.32
321-66-421	Legal Representative Brown Emily			12 Abbey Road - NW90AE (London)		
GB 9311312LH	Crystalglass LLC			12 Abbey Road - NW90AE (London)	4060.00	81.2
654-22-123	Oliver	Jake	31/08/1978	91 Western Road, B32EW - Birmingham	440.00	8.80
012-34-234	Doe	Jane	27/03/1947	N. 3425 Stone Street, BN12HB - London	322.14	6.44
					Total:	9298.09

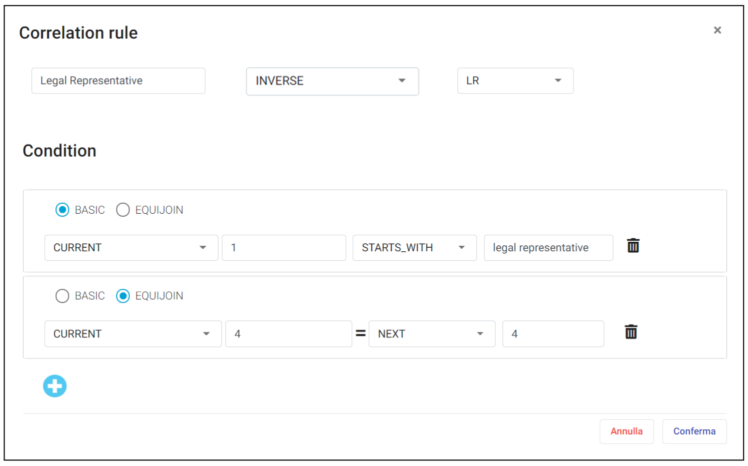
Figure 1: A spreadsheet example

2. Phase I: Extraction, Cleaning and Typing

The main purpose of this phase is to correct syntax errors occurring in the data, identifying the existing relations among table rows, and identifying basic types of each column and cell as detailed in the remainder of the section.

Table extraction. The table is extracted by using the approach described in [7]. Essentially, for the localization of the table, we consider the density of the information contained in the table w.r.t. external data. For the functional analysis of the cells, we use their position within the table and adopt dictionaries of already processed spreadsheets for identifying the column schema. For removing blank and semi-blank rows, predefined thresholds are used on the minimal number of non-empty cells in each row w.r.t. the number of columns.

Correlation between rows. For identifying correlation among table rows, we adopt a declarative pattern-based language, allowing the user to define rules, which express its knowledge about the existence of a relationship between consecutive rows. The rules can be specified through a GUI, and the user can decide the rules to be applied in processing a specific spreadsheet. More precisely, each rule is identified by a unique name $name$, and is applied to consecutive rows, the `current` row, r_i, and the `next` row, r_{i+1}. A rule is composed by a conjunction of basic *conditions* (that check for the existence of a relationship) and an *action* that expresses the way the information from the two rows should be joined when the condition is verified. The following two basic $condition$ can be specified: 1) $r[k]\ op\ v$ (named *basic* condition) requiring that the k-th cell in the row $r \in \{\texttt{current}, \texttt{next}\}$ is compared according to the operator op with a value v, where $op \in \{=, \neq, \texttt{startwith}, \texttt{endwith}\}$; 2) $\texttt{current}[k] = \texttt{next}[k']$ (named *equijoin* condition) imposing that the k-th cell of `current` row is equal to the k'-th cell of `next` row. $action$ is specified by a tuple $(conc, rel)$ determining the way in which r_i and r_{i+1} should be concatenated ($conc \in \{\texttt{natural}, \texttt{inverse}\}$) and the kind of relationship that exists among the two rows. The relationship can be: `extra`, representing further information about the invoice; LR, when the row contains the legal representative of the invoice; `heir` when the invoice is titled to a subject that is dead and one of his/her heirs should be contacted.



Correlation rule ×

Legal Representative | INVERSE | LR

Condition

[✓ BASIC] EQUIJOIN

CURRENT | 1 | STARTS_WITH | legal representative

BASIC [✓ EQUIJOIN]

CURRENT | 4 | = | NEXT | 4

Annulla | Conferma

Figure 2: Correlation rule specification

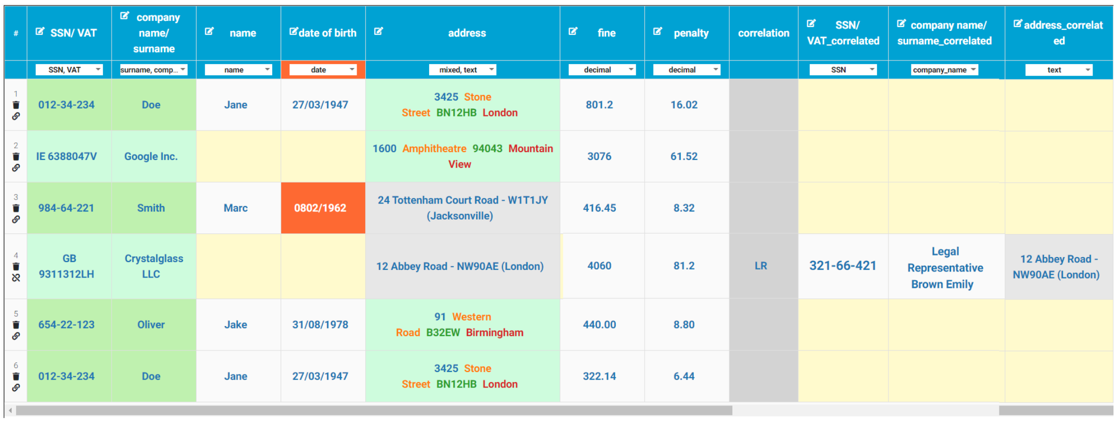
Example 1. *In our scenario, a company is represented by a Legal Representative whose data are reported in the row above the one containing the invoice. To identify this situation, the rule in Fig. 2 presents two conditions: a basic one that identifies a cell whose value starts with* `Legal Representative`*, and an equijoin to verify that the two addresses are equal. Whenever a pair of rows satisfy the condition, the first row is concatenated after the second one (inverse order).*

Rules can be specified in advance, through a graphical interface (see Fig. 2), or on the fly, by clicking on the operations on tuples loaded in the interface in Fig. 3 (first column). In both cases, the table schema is duplicated and correlated rows are concatenated to form a single table line.

Type Inference. By applying the aforementioned techniques, a table $T = \langle S, V \rangle$ is extracted from the spreadsheet, where $S = [col_1, \ldots, col_j, \ldots, col_m]$ is the list of column names, and $V = \{r_1, \ldots, r_n\}$ is the set of table rows. Each row r_i, $1 \leq i \leq n$, is a list of values, and $r_i = [v_{i,1}, \ldots, v_{i,j}, \ldots, v_{i,m}]$, one for each column identified in the column schema.

At this stage, to discriminate the role of each cell within the table and to correctly assign a meaning to its content, a preliminary step is the identification of the column and cell types among those in our type system $\mathcal{T}$ composed of $simple$, $mixed$ and $union$ types (details in [7]). $Simple$ types are basic domains and specific domains for our application (e.g. `SSN`, `VAT`, `emails`, `zip` codes). A $mixed$ type (e.g. `address` column) is a record-type associated with different patterns used for extracting the record (in the case of `address` column, it is `rec(city, streetName, streetNumber, zip)`) from a string. $Union$ types represent the occurrence of instances of different types in the same column and are denoted $union(t_1, \ldots, t_h)$ where $t_1, \ldots, t_h$ are simple or mixed types.

For type-inference we exploit decision trees, which are trained on synthetised table corpora, which have been automatically tagged with the types of our type system. The decision of using synthetised tables is firstly due to the fact that our tables use Italian language and are from a financial context; to the best of our knowledge, no such corpora exist yet. Secondly, our approach must be robust against mistakes, so that we have specifically developed the synthetizer in order to produce common mistakes we experienced in the first spreadsheets we received



#	SSN/ VAT	company name/ surname	name	date of birth	address	fine	penalty	correlation	SSN/ VAT_correlated	company name/ surname_correlated	address_correlated
	SSN, VAT	surname, comp...	name	date	mixed, text	decimal	decimal		SSN	company_name	text
1	012-34-234	Doe	Jane	27/03/1947	3425 Stone Street BN12HB London	801.2	16.02				
2	IE 6388047V	Google Inc.			1600 Amphitheatre 94043 Mountain View	3076	61.52				
3	984-64-221	Smith	Marc	0802/1962	24 Tottenham Court Road - W1T1JY (Jacksonville)	416.45	8.32				
4	GB 9311312LH	Crystalglass LLC			12 Abbey Road - NW90AE (London)	4060	81.2	LR	321-66-421	Legal Representative Brown Emily	12 Abbey Road - NW90AE (London)
5	654-22-123	Oliver	Jake	31/08/1978	91 Western Road B32EW Birmingham	440.00	8.80				
6	012-34-234	Doe	Jane	27/03/1947	3425 Stone Street BN12HB London	322.14	6.44				

Figure 3: Automatic Typing of the extracted table

(and tagged as error). The corpora consists of 10.000.000 synthetic tables that are randomly split 10 times in 8.000.000 documents for training and 2.000.000 for the validation set.

The decision tree works on a value $v_{i,j}$ represented by a vector u consisting of two parts: 1) $(u_{i,j,1}, \ldots, u_{i,j,R})$ are Boolean values obtained by the application of basic-type recognizers on the value $v_{i,j}$ (several kinds of recognizers have been realized for each of the considered basic and mixed types); 2) $(avg_{j,1}, \ldots, avg_{j,R})$ are average numbers of recognition for each type recognizer applied on the values occurring in the column j. The generated vectors are used for training a decision tree that predicts the type of each value contained in the table T.

In order to determine the type of each column, we cannot exploit a majority vote approach, because in our context we admit the presence of values belonging to a union type (as the case of the column SSN/VAT). For this reason, we have adopted an approach based on the frequency of the types in a column, where the types included in the union types are those that exceed a basic cutoff threshold. All the values whose type is not included in the union type are considered error for the current column.

Example 2. *Fig. 3 shows the result of the automatic type inference approach on our running example. White columns denote values of the same single type, whereas yellow cells denote empty values and red cells denote type mistakes w.r.t. the type identified for the column. Cells with mistakes need to be fixed by the user (also exploiting facilities made available in the platform for reducing the typing efforts). When more than one colour is used for the same column, it means that a union type was detected for such a column. Values of a mixed type are denoted by using different text colours (as in the address column where each sub-component has a different text colour).*

In the address column, there are also values that the ML algorithm was not able to extract because a non-supported pattern was used. In this case, the user exploits the interface in Fig. 4 for extracting the pattern. Note that, the identification of a correlated row led to the introduction of further columns w.r.t. those initially contained in the spreadsheet for representing all the information related to a single invoice in the same row. Further, a new column named correlation *has been introduced for representing the correlation typed identified by the rule.*

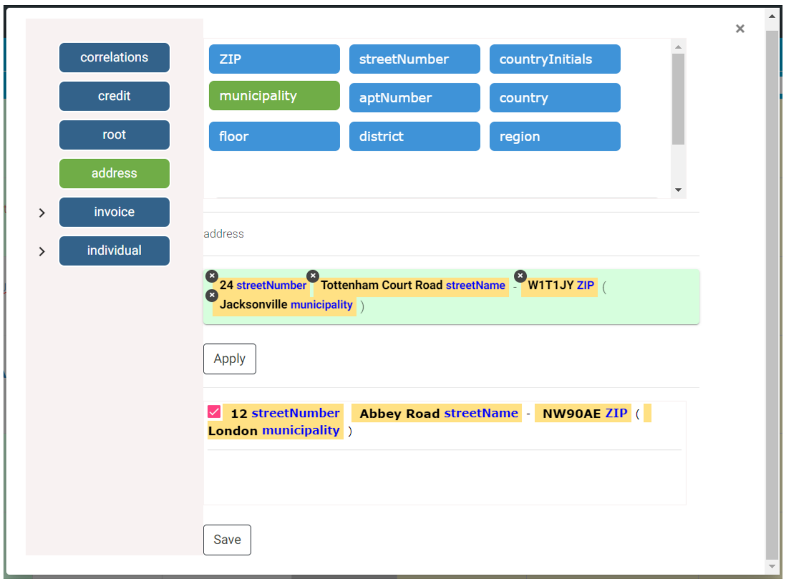

correlations
credit
root
address
invoice
individual
ZIP
streetNumber
countryInitials
municipality
aptNumber
country
floor
district
region
address
24 streetNumber Tottenham Court Road streetName - W1T1JY ZIP (Jacksonville municipality)
Apply
12 streetNumber Abbey Road streetName - NW90AE ZIP (London municipality)
Save

Figure 4: Assign or modify a type to single values

Interfaces for the manual typing and correction. The automatically identified types, however, may contain errors due to the occurrence of mistakes in the data or in the classification algorithm. Therefore, specific interfaces have been developed to make ease the correction. A key point of this manual modification is to consider the concept and properties of the Domain Ontology (details will be given in Section 3) in this way the human operator can give further information for the generation of the semantic description of the data contained in the table.

Fig. 4 shows the interface for altering the type associated with a single value (a similar interface is used for altering one of the types specified for a column). In the left panel, the Domain Ontology concepts are reported. Once a concept has been chosen, the list of its properties is shown in the top part of the interface, so that the user can use the selected property for labelling a portion of the value or the entire string. By assigning different properties to differing parts of the string, the pattern of a mixed type is identified. The identified pattern can thus be checked to other values of the same column (those sharing the same initial type). In this way, a single type specification can be applied to many values. Moreover, the identified pattern is stored in the system and made available for processing subsequent spreadsheets.

Preliminary experiments. To validate the type prediction algorithm we considered 40 spreadsheets offered by the debt collection agency. Each spreadsheet may contain errors and has around 100 rows and a variable number of columns (between 7 and 25). Starting from this original dataset, the ground truth dataset has been created by manual correction of the errors and subsequent type labelling. After processing the original documents as described in this section, the precision, recall, and F1 measures have been computed by taking into account the occurrences of mixed, union types, errors and empty values. We remark that the presence of union types requires to consider as true positives (TP) those cells whose predicted type corresponds to the manually assigned type, as false positives (FP) those cells whose predicted

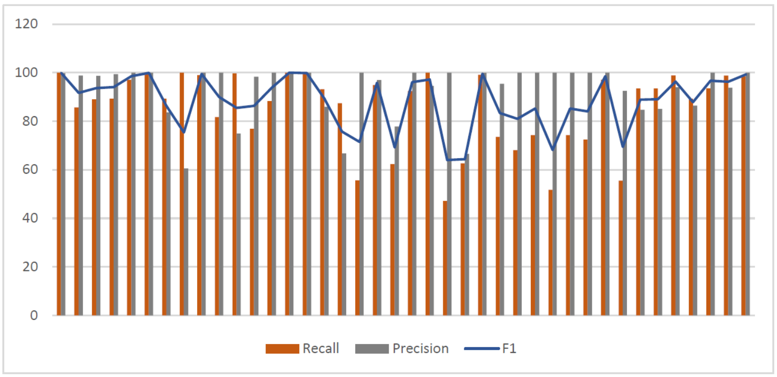


Figure 5: Recall, Precision and F1 of 40 documents

type does not match the manually assigned type (among those specified in the union type), as false negatives (FN) those cells for which an error has been predicted and the expected type is one of the union type, as true negatives (TN) those cells whose predictive type does not match those occurring in the union type.

The results are shown in Figure 5. For more than a half of all the documents we obtained a precision of 100%, while the lowest result is above the 60%; this result highlights that the system rarely identifies an error as a correct value. The recall of two-thirds of the documents is over the 85%, which means that most error cases are identified by the ML algorithm. The F1 measure is above 80% for 32 documents. In most cases, the mistakes of the ML algorithm (FN) are due to cells containing mixed types. As future work, we plan to exploit the user-defined patterns in the prediction system in order to incrementally reducing the user efforts in manual typing.

3. Phase II: Semantic Model Generation

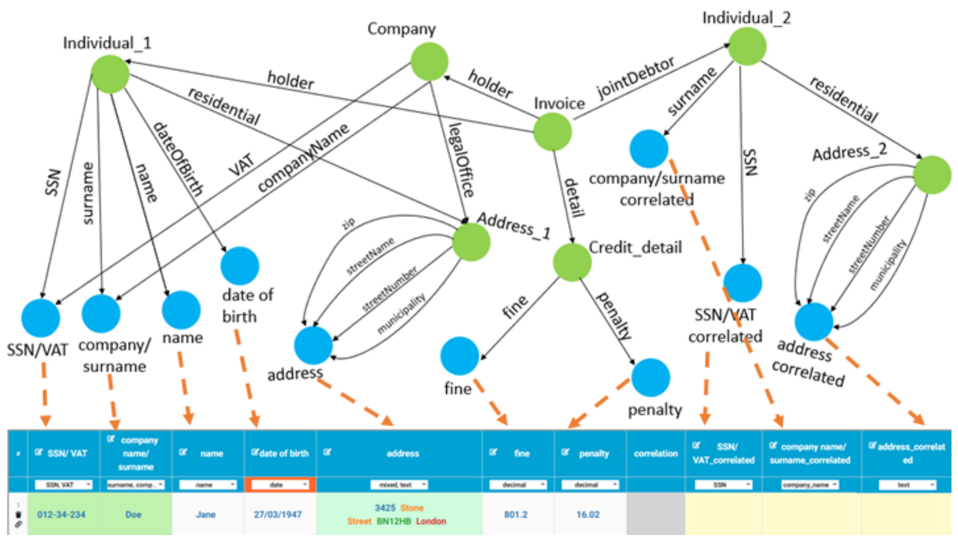
The aim of this phase is to provide a *semantic description* of the spreadsheet tables by means of annotations w.r.t. a Domain Ontology. Even if many approaches have been proposed for this problem ([3, 4, 5, 6]), in our research we wish to face the presence of union types for table columns, which is generally neglected. Our semantic description is inspired to the one used in Karma [8] but differs because it is created starting from the types identified in Phase I and allows the extraction of several data from a column (while Karma only admits a 1:1 correspondence).

The Domain Ontology. Starting from the analysis of the information system currently used by the debt collection agency for the management of the invoices, we have developed a Domain Ontology containing the concepts and relationships useful for the considered scenario. Among them, we have identified the `invoice` concept, representing the document assigned by the local authority to the debt collection agency. The invoice contains many properties (e.g. the date in which it has been issued, the total amount) and is associated with a `subject`. Subjects can be classified in `individuals` and `companys` and can be associated with one of more `addresses`. Addresses can specify the residence/domicile of an individual or the legal/administrative office of a company. The invoice is associated with the detailed credit/debit voices (concept `details`) that the total amount to be rescued is composed of. Each invoice can be also associated with other subjects that are co-obligated in its payment (e.g. a legal representative or an heir). Formally, a Domain Ontology O consists of a set of Concepts $C = \{C_1, \ldots, C_n\}$. Concepts can be organized in a hierarchy of concepts: $C_1 \sqsubseteq C_2$ denotes that C_1 is subclass of C_2. Each Concept can have associated basic properties taken from a set $P = \{p_1, \ldots, p_m\}$; each property is associated with basic types of our type system $\mathcal{T}$; the properties of a concept C are denoted $P(C)$. Among the properties belonging to a class, we identified a subset of *mandatory properties*. These properties need to be specified for properly characterizing an individual of the concept and it is relevant for identifying the presence of instances of real world concepts in the KB (and thus avoiding the introduction of duplicates). Relationships can be identified among concepts and denoted by means of a set of roles R that identifies the kind of relationships that bind the individuals of two concepts. Some relationships can be considered mandatory to be identified in the semantic description (e.g. we cannot have an invoice without its subject and the corresponding address).

The Semantic Description of a table. The semantic description of a table $T = \langle S, V \rangle$ is a graph $(U_C, U_T, E_R, E_E, \phi_{E_R}, \phi_{E_E})$, where U_C is a multiset of concepts belonging to the Ontology, $U_C \subseteq C$, (we consider a multiset because a concept can appear more than once), U_T is a set of nodes corresponding to the columns in T ($U_T \subseteq S$), $E_R \subseteq U_C \times U_C$ are edges representing the relationships existing among concepts in V_C, and E_E are edges representing the extraction of values from the columns of T that are used for feeding properties of concepts in U_C. Moreover labeling functions ϕ_{E_R}, ϕ_{E_E} are introduced. The first one is used for assign to edges in E_R the name of the relationship that they represent (i.e. $\phi_{E_R} : E_R \to R$, where R is the set of relation names), whereas the second one is used for determining the type of values that can be extracted from a column of T, i.e. $E_E \times \mathcal{T} \to P$, where $P(C)$ is the set of properties. This function assigns to an edge (u_T, u_C) and a type $t \in \mathcal{T}$ a property among those specified for the concept associated with the vertex u_C. We remark that in our context more than one type of information can be extracted from the same column because of the presence of union and mixed types. In the case of union type, the extraction assumes the meaning of identifying different kinds of values in different rows. For mixed type, the extraction assumes the meaning of identifying a record from a string value of a table row.

Example 3. *The top part of Fig. 6 reports the semantic description obtained for our running example. In our web tool, this description is editable and the user can fix it or include other properties and relationships depending on the needs. The blue nodes represent the column of the table from which data are extracted. The green nodes represent classes. Edges among green nodes represent relationships available in the Domain Ontology, whereas, edges between green and blue nodes represent extraction rules. Whenever more than one edge arrives at the same blue nodes, it means that different alternative information (i.e. values of a union type) or a mixed value can be extracted.*

Semi-automatic Construction of Semantic Descriptions. Even if our graphical interfaces allow the manual creation of the semantic description starting from the types obtained at the end of phase I, this approach is tedious and does not take into account the semantic descriptions $s_1, \ldots, s_p$ already developed for other spreadsheets. Two approaches have been adopted for constructing a semantic description by relying on previous experience. First, each s_i ($1 \leq i \leq p$)



#	SSN/ VAT	company name/ surname	name	date of birth	address	fine	penalty	correlation	SSN/ VAT_correlated	company name/ surname_correlated	address_correlated
	SSN, VAT	surname, comp...	name	date	mixed, text	decimal	decimal		SSN	company_name	text
1	012-34-234	Doe	Jane	27/03/1947	3425 Stone Street BN12HB London	801.2	16.02				

Figure 6: Mapping between the extracted table and the generated semantic description

is associated with an hash value obtained by applying an hash function on the ordered list of types obtained at the end of phase I. This value is used as an index for identifying a possible semantic description for the novel spreadsheet. This approach is adequate when the spreadsheets always adhere to a pre-defined structure. Another approach is to keep track of the types of the single columns and fragments of the semantic descriptions already associated with them. Whenever a column is labeled with a type, the fragments associated with it are collected and integrated into a single description. Since many combination of fragments could be possible, we adopt a scoring function that try to maximize the adoption of fragments extracted from types used in the same table. However, there is no guarantee that the final semantic description of all the the columns of the spreadsheet is correct. For this reason, the semantic description is drawn as a graph, and graphical facilities support the user for updating its representation.

4. Phase III: Construction of the Knowledge Base

At the end of the previous phase, we have constructed a semantic description of the information contained in the table extracted from the initial spreadsheet. The description characterizes the information contained in the table, but can also be used for transforming the table rows in instances of a KB that adhere to the constraints imposed by the Domain Ontology. In this way, the knowledge base becomes a data-store of the invoices treated by the debt agency and can be exploited for predicting the capacity of the invoice subjects in paying the due imports. Indeed, by combining the invoice information with the subjects' previous payments/not-payments, it is possible to predict when a given invoice will be positively rescued by a given subject.

In the transformation process, the following issues should be taken into account. First, the presence of union types and missed properties can lead to the generation of different semantic descriptions for each row of the table and special attention should be kept in generating

Algorithm 1 `GenerateInstances`

```
    Input: A knowledge base $KB$, A semantic description $S_D$, A table $T = \langle S, V \rangle$
    Output: $KB$ enhanced with the content extracted from $T$
 1: for each row of $T$ do
 2:     create a copy $\bar{S}_D = (U_C, U_T, E_R, E_E, \phi_{E_R}, \phi_{E_E})$ of $S_D$
 3:     apply the extraction rule associated with $\phi_{E_E}$
 4:     assign the extracted value to the corresponding vertex in $U_T$
 5:     for each class $c$ in $U_C$ do
 6:         if all the properties in $c$ have been removed and
 7:          $c$ is not involved in mandatory relationships then
 8:             remove the class $c$ and all the relationships in which it is involved
 9:         end if
10:         if $c$ still exists and mandatory properties in $c$ are missing then
11:             include the mandatory properties with fake values
12:         end if
13:     end for
14:     Consider the updated structure $\bar{S}_D$, for creating the triples to include in $KB$
15: end for
```

consistent values according to our Domain Ontology. Second, when a mandatory property is missing in the considered row, this property should be included in any case in the semantic description and proper interfaces should be developed for supporting the user in fixing these properties. Finally, the data extracted from the spreadsheet should be included in the KB. Therefore, we need to identify overlapping with the instances of the KB that need to be integrated. The integration process should guarantee that coherent and correct information are maintained in the KB by exploiting well known properties of data quality [9].

The last problem is a well-known problem in the context of data quality (object identification problem [9]) and we plan to introduce facilities in this direction in the next release of our platform. For what concerns the first two issues, Algorithm 1 has been developed with the aim of generating the proper instances of the Domain Ontology that describe the single invoices contained in the data table. The algorithm works row by row on the table identified at the end of phase 1 (see Fig. 3) and allows the extraction of values according to the properties and classes identified in the semantic description. The algorithm takes into account the possibility that classes of the semantic description are not instantiated because alternative classes have been fed (due to the presence of union types). Moreover, the algorithm imposes the introduction of properties when they are mandatory in the Domain Ontology.

Example 4. *Fig. 7 reports the generated graph. Brown nodes represent invoices. An invoice is sometimes titled to individuals, other times to companies. In the case of the correlated tuples discussed in Example 1 for invoice 4 (the area highlighted on the top part of the figure), the invoice has been titled to a company and presents a legal representative (note that in this specific case, the two subjects share the same address). Finally, the subject Jane Doe (the area highlighted on the bottom part of the figure) is represented only once in the KB because the two instances of the subject share the same SSN, and so they are considered the representation of the same individual.*

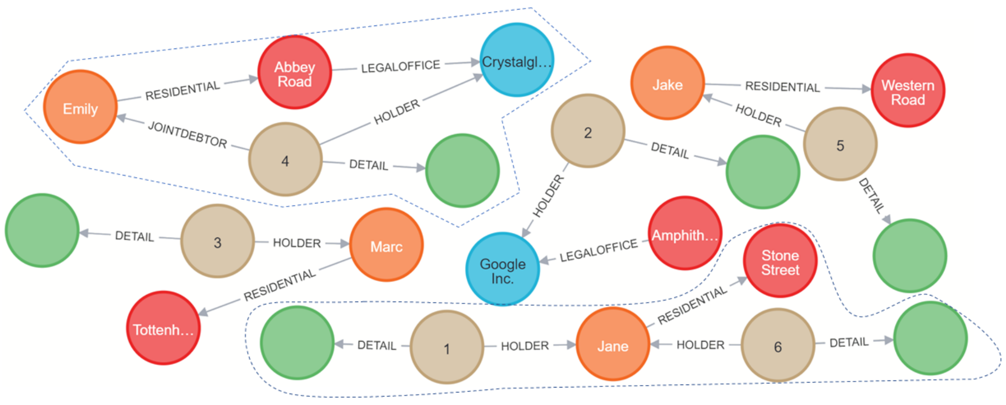


Figure 7: Knowledge base for the considered running example

5. Concluding Remarks

Both approaches for inferring simple data types and concepts of a knowledge base have been proposed in the literature. In the first category falls many studies, including wrangling tools [10, 11], software libraries (e.g. [12]) and probabilistic approaches (e.g. [13]) that exploit validation functions and regular expressions on data samples for inferring a single type. However, these approaches are able to infer a very limited number of types and usually do not work very well when missing and anomalous data occur in the sample. Many approaches are nowadays proposed for inferring concepts of a knowledge base by exploiting different pre-annotated corpora (e.g. [8, 14, 4]). However, the tolerance to mistakes is quite limited and in many cases they require that table values occur in the KB.

In this paper, we have proposed an approach that is not completely automatic but combines ML techniques, data management facilities and user-friendly interfaces for dealing with heterogeneous tables that can contain different kinds of mistakes. The approach has been tested on a collection of documents made available from the debt collection agency that needs to handle every day this kind of documents that are highly heterogeneous and with many mistakes. Our initial experiments prove the feasibility of the approach and the utility of our interfaces for easily fixing the mistakes and generating a consistent KB. The tool is not yet publicly available because it contains sensitive information, however interested readers can get in touch with us for a demo.

The work discussed in this paper can be extended in several directions. By means of the graphical interfaces described in Fig. 4 it is possible to define new patterns and include them in the ML process described in Section 2. This is an interesting research direction in the spirit of incremental learning and thus being able to adapt the model without the entire re-training. The extension of the learning facilities discussed in Section 3 for learning a semantic description from previous generated descriptions is another interesting research direction that can take advantage of the work discussed in [8]. Moreover, a systematic evaluation of the performances of the entire web application should be conducted. Furthermore, we wish to combine our approach for the construction of the semantic description of a table with the approaches proposed in

[3, 4, 5, 6] by taking into account the peculiarity of our domain. Finally, we also plan to apply the concepts discussed in this paper for dealing with other kinds of semi-structured data in order to semantically characterize data produced in heterogeneous sources (e.g. IoT data [15, 16], graphs [17], and tree-structured data) and represented in different models for their integration.

Acknowledgements. The authors wish to thank Giovanni Mancinelli for working on the implementation of the tool.

References

[1] M. Hurst, The Interpretation of Tables in Texts, Ph.D. thesis, Uni. of Edinburgh, 2000.

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

[3] Z. Zhang, Effective and efficient semantic table interpretation using tableminer$^+$, Semantic Web 8 (2017) 921–957. doi:10.3233/SW-160242.

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

[5] S. Zhang, E. Meij, K. Balog, R. Reinanda, Novel entity discovery from web tables, in: Proc. of The Web Conf., ACM, 2020, p. 1298–1308. doi:10.1145/3366423.3380205.

[6] M. Cremaschi, F. D. Paoli, A. Rula, B. Spahiu, A fully automated approach to a complete semantic table interpretation, Future Gener. Comput. Syst. 112 (2020) 478–500. doi:10.1016/j.future.2020.05.019.

[7] S. Bonfitto, L. Cappelletti, F. Trovato, G. Valentini, M. Mesiti, Semi-automatic column type inference for csv table understanding, in: 47th Int. Conf. on Current Trends in Theory and Practice of Computer Science., Springer, 2021, pp. 535–549. doi:10.1007/978-3-030-67731-2.

[8] M. Taheriyan, et al. Learning the semantics of structured data sources, J. of Web Semantics (2016).

[9] C. Batini, M. Scannapieco, Data and Information Quality - Dimensions, Principles and Techniques, Data-Centric Systems and Applications, Springer, 2016. doi:10.1007/978-3-319-24106-7.

[10] Trifacta, Trifacta wrangler, 2020. URL: https://www.trifacta.com/.

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

[12] T. Petricek, G. Guerra, D. Syme, Types from data: Making structured data first-class citizens in f#, in: Proc. of 37th ACM SIGPLAN Conf. on Programming Language Design and Implementation, ACM, 2016, p. 477–490. doi:10.1145/2908080.2908115.

[13] 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.

[14] 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.

[15] S. Bonfitto, F. Hachem, E. G. Belay, S. Valtolina, M. Mesiti, On the bulk ingestion of iot devices from heterogeneous iot brokers, in: 2019 IEEE International Congress on Internet of Things (ICIOT), 2019, pp. 189–195. doi:10.1109/ICIOT.2019.00039.

[16] S. Valtolina, L. Ferrari, M. Mesiti, Ontology-based consistent specification of sensor data acquisition plans in cross-domain iot platforms, IEEE Access 7 (2019) 176141–176169. doi:10.1109/ACCESS.2019.2957855.

[17] M. Mesiti, Mergegraphs: a web-based system for merging heterogeneous big graphs, in: Proc. of the 17th Int'l Conf. on Information Integration and Web-based Applications & Services, ACM, 2015, pp. 1:1–1:10. doi:10.1145/2837185.2837211.