Improving the Usability of Tabular Data Through Data Annotation, Repair and Augmentation

Rabeb Abida(✉) and Anthony Cleve

PReCISE, NaDI, Faculty of Computer Science, University of Namur, Namur, Belgium
{rabeb.abida,anthony.cleve}@unamur.be

Abstract. In recent years, a rapidly increasing amount of information has been made publicly available in tabular form on the Web. Many of these data are not usable due to their poor quality (e.g., misspelled or missing values, missing or incomplete metadata, and missing meaningful columns). Solutions have been proposed in the literature to address these data quality issues, but there is still a lack of all-in-one approaches that can fully solve them. Therefore, users need to use several methods to solve these data quality issues. In this paper, we present an all-in-one and automatic approach called SINATRA that helps to bridge this gaps by providing the following features: *data annotation* (to address misspelled and incomplete metadata issues), *data repair* (to address missing values (data) issues), and *data augmentation* (to dynamically add meaningful columns and corresponding cell values to the dataset). An evaluation of the SINATRA approach based on datasets from a state-of-the-art benchmark shows promising results in terms of F1-measure and precision.

Keywords: Usability · Tabular data · Data annotation · Data repairing · Data augmentation

1 Introduction

Nowadays a vast amount of information is provided on the Web in unstructured text, semi-structured data, and more structured data in the form of tables [2,4,10,12]. They can sometimes be difficult to use due to data quality issues, such as misspellings and missing metadata, ambiguity in table cells, missing cell values, and missing significant columns [4,6–8,10,12].

Several methods have been proposed in the literature to solve the aforementioned issues. On the one hand, the use of *Semantic Table Annotation* (STA), also known as *data annotation*, consists of assigning semantic tags from knowledge graphs (KGs) (e.g., Wikidata [15] and DBpedia [3]) to the data columns elements. The *data annotation* has proven to effectively solve the problem of spelling errors and missing or incomplete metadata [8–10,12,13]. On the other hand, *data repair* handles the problem of missing cell data (values), and *data augmentation* adds meaningful columns and corresponding cell values to the

E. Zouganeli et al. (Eds.): NAIS 2022, CCIS 1650, pp. 65–77, 2022.
https://doi.org/10.1007/978-3-031-17030-0_6

data. As part of the "Tabular Data to knowledge Graph Matching" competition [9], some approaches have implemented the STA process, such as [8,10,12], but they have not incorporated *data repair* and *augmentation* phases. Meanwhile, other works such as OpenRefine[1] and Magic [13] propose a system that is capable of both annotating and augmenting a dataset, but they do not support any *data repair* phase.

Despite the systems proposed in the literature to solve these data quality issues, there is still no all-in-one approach that can handle them, and nor are there other features that can further support the STA process. Therefore, users need to use multiple methods to tackle these problems.

In this paper, we present an all-in-one and fully automatic proposal called SINATRA (**S**emant**I**c a**N**notation **A**ugmenta**T**ion and **R**ep**A**ir) that helps fill these gaps by providing the following features:

(i) ***data annotation*** is used to resolve spelling errors and missing or incomplete metadata. It is based on the STA process, which consists of three main tasks: Column type Annotation (CTA) (Fig. 1c), Column property annotation (CPA) (Fig. 1a) and Column Entity Annotation (CEA) (Fig. 1b). They assigned the data elements to the concepts in the knowledge graph (DBpedia KG), as shown in Fig. 1. To describe each task in the STA process [12], we consider a table of real dataset[2] in Fig. 1, which presents the names of the presidents (col1) and their place of birth (col2).

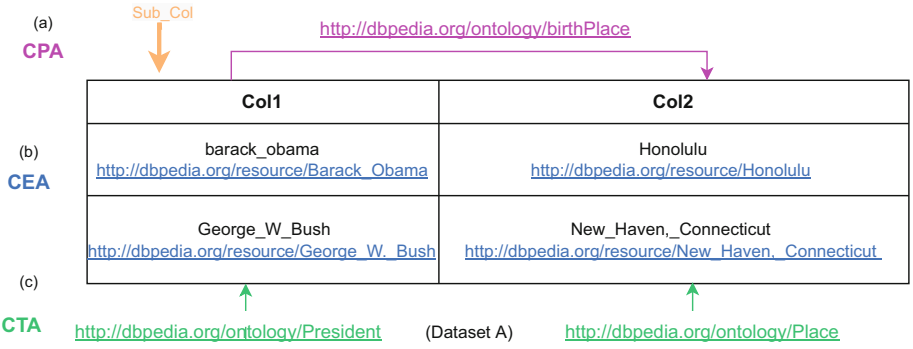

Fig. 1. Data annotation. Tabular data (black) is annotated with the properties (magenta), entities (blue), and types (green) from DBpedia as asked in the CPA (a), CEA (b), and CTA (c) tasks respectively. (Color figure online)

(ii) ***data repair*** is used to handle missing or incomplete cell values in the dataset. It is based on a method that applies SPARQL queries to fetch missing cell values from the DBpedia KG. Figure 2 shows an example of the data repair phase by adding a cell value "http://dbpedia.org/resource/Honolulu".

[1] https://openrefine.org/.
[2] https://tinyurl.com/4hrx6s48.

(iii) ***data augmentation*** is used to dynamically add meaningful columns and their corresponding cell values to the dataset. It is based on a method that applies (i) SPARQL queries to fetch the property URIs (CPA) of the new columns proposed by users and (ii) SPARQL queries to fulfill the corresponding cell values of the newly added columns. Figure 2 shows an example the data augmentation feature by adding a new column "http://dbpedia.org/ontology/birthDate".

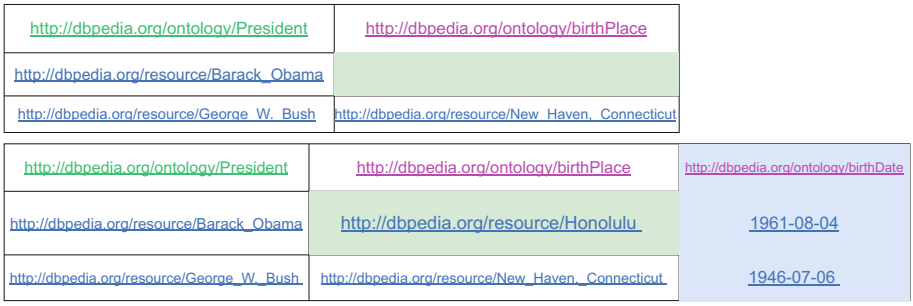

Fig. 2. Example of *data repair* by adding cell value "http://dbpedia.org/resource/Honolulu" (light green) and *data augmentation* by adding new column "http://dbpedia.org/ontology/birthDate" (light blue). (Color figure online)

For evaluating our approach, we used some of the datasets proposed by the "Tabular Data to knowledge Graph Matching" [9,10] competition to measure the effectiveness of the SINATRA approach by F1-measure and precision metrics and demonstrate the capability of its features.

The remainder of the paper is organized as follows. Section 2 positions our work with respect to related literature. Section 3 gives an overview of our approach, describes in detail the different phases it covers, and presents its implementation. Section 4 evaluates SINATRA and assesses the effectiveness of its phases. Section 5 concludes this paper and anticipates future research directions.

2 Related Work

This section reviews related work on popular approaches and tools that address gaps in data quality issues (e.g., misspelled or missing values, missing or incomplete metadata, and missing meaningful columns). We present them with their respective features, strengths and weaknesses.

Some works have been proposed, mainly with a particular and non-integrated focus on data pre-processing, subject column (Sub_Col) detection [13]. Furthermore, OpenRefine and [11,14] rely only on their own data (domain-independent) and perform only a few steps of the STI process. They can be classified as supervised (Sup: they exploit already annotated tables for training) and semi-automatic. Other works [8,10,12,13] can be classified as unsupervised (Unsp:

they do not require training data) and automatic. They do not provide a user-friendly graphical interface, and manually annotating the data is time-consuming for the user.

The STA process [10] is composed of five steps which are: (i) the data pre-processing, which aims to prepare the data inside the table; (ii) the detection of the Sub_Col is designed to detect the main column of the table; and (iii) the three sub-steps for the *data annotation*, which are CEA task (Fig. 1b), CTA task (Fig. 1c), and CPA task (Fig. 1a). Other proposals have been made to resolve the gaps in the above-mentioned approaches and perform all the steps of the STA process. In this way, [8,10,12] propose novel techniques to improve and provide high-quality annotations to address the issues of misspelling and missing or incomplete metadata. They used unsupervised learning techniques, which could be applied to general-purpose domains, and utilized Open Source KG that was freely available on the Web (DBpedia). MantisTable [8] used some features to resolve the limitation of the Subject Column (Sub_Col) task. It allowed users to apply a series of steps to prepare data and used different features to automatically assign the Sub_Col. MTab [12] tool as an automatic semantic annotation system, could jointly deal with the three tasks CTA, CEA and CPA. It was based on the joint probability distribution of multiple tables to DBpedia KG matching. MTab achieved impressive empirical performance for the three annotation tasks of the STA process and won the first prize at the SemTab challenge [9,10]. MTab did not offer subject column detection but has excellent results and MantisTable did not offer excellent results like MTab but allowed Sub_Col detection [9,10]. Those systems [8,10–12,14] can not create or add new columns to *augment* the annotation with additional knowledge graph (KG).

However, OpenRefine and Magic [13] have offered systems capable of both annotating and augmenting a dataset. OpenRefine can perform a semi-automatic reconciliation process against any database that exposes a Web service using Reconciliation Service API[3] specification or a SPARQL endpoint. This tool requires the user to manually correct a cell that has multiple entities (CEA). In addition, it is also able to create new columns through facets, where the user has to formulate the URL to fetch the URIs. Magic [13] offered a system capable of annotating a dataset using the interpretable embedding technique and utilized KGs (DBpedia, WikiData). It can be added a column to further *augment* the Tabular Data. It did not do the pre-processing data phase and used techniques, which were already proposed by the state-of-the-art approaches for that particular phase. Magic might not be outperform the existing state-of-the-art techniques to generate such annotations [1]. Despite all their achievements and results, these proposed tools are not in a position to solve the problems of missing cell values. They do not include the *data repair* phase.

In addition, in the R&D community, there is a lack of automated support [2,5], which can combine the appropriate features defined in Table 1 to assist users in overcoming data quality issues.

[3] https://github.com/OpenRefine/OpenRefine/wiki/Reconciliation-Service-API.

Table 1 summarizes the selected approaches and tools that meet certain features: *Data annotation*, *Data repair* and *Data augmentation*, and shows the difference between them and our proposed approach SINATRA.

Table 1. Approaches and tools that support the above features: *Data annotation*, *Data repair* and *Data augmentation*.

Approach & tools	STA process (data annotation feature)						Features		KGs/ontology import	Export
	Learning techniques	Data pre-pro	Sub-col	CEA	CTA	CPA	Data aug. (add-col)	Data repair (add missing cell values)		
Open refine	Sup	x	-	x	-	x	x	x/-	Wikidata FreeBase	x
Odalic [11]	Sup	-	-	x	-	x	-	-	DBpedia Dom.Ind	x
DataGraft [14]	Sup	-	-	x	x	-	-	-	Dom.Ind	-
MantisTable [8]	Unsup	x	x	x	x	x	-	-	DBpedia	-
MTab [12]	Unsup	x	x	x	x	x	-	-	DBpedia	-
Magic [13]	Unsup	-	x	x	x	x	x	-	WikiData	-
SINATRA	Unsup	x	x	x	x	x	x	x	DBpedia	x

SINATRA is a solution designed as an all-in-one and automatic approach based on MantisTable [8] and MTab [12] systems, which will be described in Sect. 3.

3 The SINATRA Approach

This section describes a fully automatic approach, which combines all methods and tools into one integrated approach.

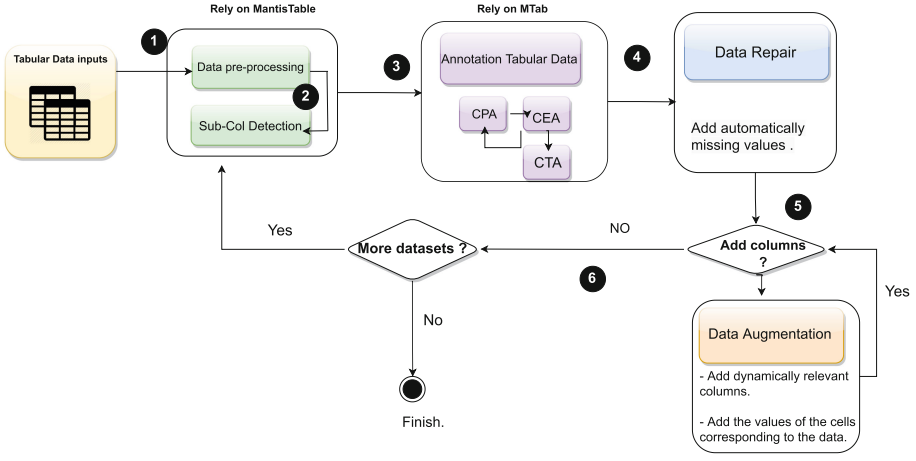

Fig. 3. An overview of **SINATRA** approach (tool).

This proposal overcomes the associated difficulties with data quality on the Web, especially tabular data. More details on the implementation of the approach are available online[4]. It implements its features: *Data annotation*, *Data repair* and *Data augmentation* through the following four phases such as, **Data pre-processing** and **Subject Column (Sub_Col) detection**, **Data Annotation**, **Data repair**, and **Data augmentation**, which Fig. 3 presents an overview of the proposal.

1. During the **Data pre-processing and Sub_Col detection** phase, the SINATRA approach takes as input a large number of local Excel or CSV datasets on the user's computer in order to focus the users to automatically prepare the datasets and detect the Sub-Col before applying the data annotation phase. This phase is based on the Mantistable approach [8] and consists of two steps: (i) Data pre-processing step, the process begins to clean and uniform Data inside the table, remove HTML tags, stop words and some character (i.e.," '), turn text into lowercase, delete of text in brackets, and normalize measurements units. Once this step is complete, the system switches to detect (ii) the Sub_Col. It is as the Subject of relationships among columns, and the annotation of other columns as Objects (Fig. 1 represented Sub_Col by the orange color). This step starts by determining the literal columns (e.g., address, phone number, URL, color) using regular expressions. Once this step is complete, the system chooses from remaining columns (called Named Entity columns), the subject column (Sub_Col) based on different statistic features, such as the average number of words in each cell, fraction of empty cells in the column, the fraction of cells with unique content, and distance from the first-named entity column [8]. More details on those steps can be found in [8]. Once the phase has finished, it moves on to the second phase, which consists of annotating the dataset.

2. Data Annotation phase aims to automatically annotate Tabular data elements with DBpedia KG (Fig. 1). This phase relies on the MTab approach [12] to generate the three tasks: the Column Entity Annotation (CEA), whose task is to map table cells (values) to entities in DBpedia (Fig. 1b); the Column property annotation (CPA) to map column-pairs to an ontology property (Fig. 1a); and the Column type Annotation (CTA) whose task to map table columns to an ontology class (Fig. 1c). The mapping process in MTab is based on the joint probability distribution of multiple tables to KG matching. It improves the matching by using multiple services including, DBpedia Lookup, DBpedia endpoint, and WikiData lookup, as well as a cross-lingual matching strategy. Ths mapping is done in six steps. (i) The first step estimates the most candidate entities (CEA) that were found by those different search services. (ii) The second step is to infer the most classes (CTA). It estimates the entity columns and the numerical columns. If the vote returns a text or integer tag, then the column is of type entity otherwise it is numeric [16]. (iii) The third step establishes the relationships between the different columns (CPA) using the DBpedia Endpoint. (v) Step five is the selection of the highest probabilities of the candidates

[4] https://github.com/123rabida123/SINATRA-Annotation-Repair-Augmentation.

(CEA) in step four to establish their relationship (CPA) via a majority vote. (vi) Step six corresponds to the selection of the highest probabilities of the candidates (CEA) in step four to establish their type (CTA) via the majority vote. More details about each step of MTab can be found in [12]. Our contribution in the first two phases is that combined the strengths of MantisTable and MTab to perform both sub-steps.

MTab does not offer a Sub_Col detection phase but has excellent results in annotating data solves misspelling issues; and MantisTable does not offer excellent results like MTab but allows Sub_Col detection.

Once the data annotation phase completes, we get an annotated dataset, but some cells in this dataset still have null values "nan" (Fig. 4a). Hence, we can observe the MTab system's shortcoming, which cannot add the missing cell values in the datasets, as shown in the example in the screenshots (Fig. 4a).

3. Data repair phase aims to automatically add missing cell entities (values) or undefined values "nan". Our algorithm applies SPARQL queries by taking the cell entity (CEA) of the Sub_Col and the column property (CPA) (e.g., CEA + CPA) to retrieve the missing cell entities (CEA). An example of a SPARQL query to get the missing cell entity of the first row in the above dataset (Fig. 2).

In some cases, the query returns ambiguous entities. In this case, our algorithm calculates the pre-score of each entity using the *confidence-score* (CFS) of the Sub_Col entity and the cell entity, and determines the relationship. If there is a relation (CPA) between them (Sub_Col entity and Cell entity), the CFS increases by 1. For example, CFS (honolulu) = 1, CFS (Honolulu) = 1 and there is a relation between "barack_Obama" and "Honolulu", hence CFS = 2.

The SPARQL query (Listing 1.1) retrieves an object for the content of the column "http://dbpedia.org/ontology/birthPlace" (Property/Predicate) and the subject of the first row "http://dbpedia.org/resource/barack_Obama" from DBpedia KG, where the cell entity (object) retrieved by the query (Listing 1.1) is "http://dbpedia.org/resource/Honolulu" (Fig. 4b).

```
{
PREFIX dbr: <http://dbpedia.org/resource/>
      SELECT ?object
       WHERE
           { <http://dbpedia.org/resource/barack_Obama>
             <http://dbpedia.org/ontology/birthPlace>
             ?object
           }
}
```

Listing 1.1. SPARQL query to retrieve a cell entity (Object).

4. During the **Data augmentation** phase, the system allows the user to add relevant columns to the annotated dataset (Fig. 2). The user simply enters a word "new-Column" (Listing 1.2) to choose a CPA (URI of the new column) in the proposal list of this approach. For the same word (e.g., new-Column = "birth"), there can be several URIs (CPA) that appear in this list,

such as: "http://dbpedia.org/ontology/birthDate" and "http://dbpedia.org/ontology/birthDeath". The user chooses the one CPA, and SINATRA will be added as a new column to the dataset, or she/he can enter the name of the column exactly as "birthDate". Therefore, the system allows the user to add the chosen CPA "http://dbpedia.org/ontology/birthDate" if it is not already in this annotated dataset (Fig. 4c). The algorithm has created a list of CPA proposals, where, each time the query (Listing 1.2) returns a CPA (Predicate has an rdf:property), which contains a word proposed by the user, it stores it in this list.

```
{
PREFIX dbr: <http://dbpedia.org/resource/>
      SELECT ?predicate
       WHERE {
                ?predicate a rdf:Property
                FILTER ( REGEX ( STR (?predicate), http://dbpedia.org/ontology/, i) )
                FILTER ( REGEX ( STR (?predicate), "␣+␣new-Column␣+␣", i ) )
              }
ORDER BY ?predicate
}
```

Listing 1.2. Generic query to detect predicates from a SPARQL endpoint to add column.

Once the user chooses a CPA, the system creates a new empty column and then applies the same SPARQL queries (Listing 1.1) of the *data repair* phase to fulfill the corresponding cell entities of the newly added column.

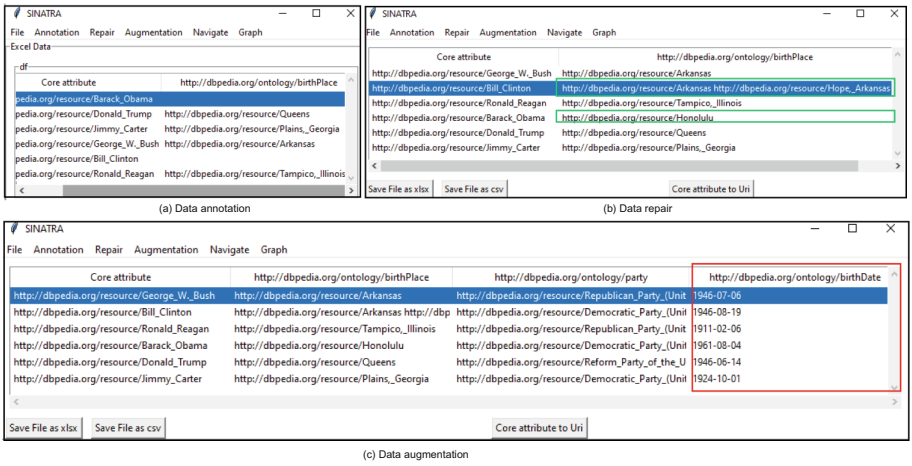

Fig. 4. Screenshots of the *data annotation*(a), *data repair*(b), and *data augmentation*(c) features of SINATRA.

According to the user's request, the data augmentation phase can create more than one column, as illustrated in step 5 of the (Fig. 3). When the system has finished the previous phases, if there are still datasets to annotate, it restarts the first phase and executes the same phases of the SINATRA process (Fig. 3). SINATRA saves the annotated datasets in a local folder and can be exported in Excel (XLSX) and CSV format.

Figure 4 depicts the graphical interface of SINATRA and focuses on data annotation (a), data repair (b), and data augmentation (c) features. We chose to use the python library *Tkinter*[5] to develop the graphical interface. Visually, tkinter is less pretty than other extensions, but it is better to check the frequency of updates of their source code before choosing one, and its license is more flexible. The implementation of the SINATRA approach, which source code is available on GitHub[6] for future research.

4 Evaluation and Demonstration

This section presents the detail about benchmark datasets, ground truths, and evaluation metrics in Sect. 4.1, followed by the evaluation results and demonstration in Sect. 4.2. This evaluation aims to measure the performance of the *data repair* and *data augmentation* features of the SINATRA approach. In the next section, we present the results of the evaluation and the demonstration of its features.

4.1 Datasets, Ground Truths and Measures

To evaluate this proposal using randomized datasets[7] and the ground truths proposed by the SemTab competition [9,10]. These ground truths are composed of three targets (CEA-targets, CPA-targets, and CTA-targets)[8] matching with DBpedia KG for each annotation task (CEA, CTA, and CPA).

In Table 2, we present the datasets used in our evaluation: Reference of the Dataset, Dataset, #Col, #Rows, and Names of columns.

Table 2. The characteristics of the datasets were evaluated by SINATRA approach

Ref	Datasets	#Col	#Rows	Name of columns
D1	211	4	56	Col1: University of UEA; Col2: President; Col3: number of staff; Col4: Surface area
D2	212	4	15	Col1: University; Col2: Post holde; Col3: Number of students; Col4: Number of staff
D3	274	5	17	Col1: Name of the animal; Col2: Family of the animal; Col3: (unnamed); Col4: Location; Col5: (unnamed)
D4	275	5	65	Col1: Name of the animal; Col2: Family of the animal; Col3: Place; Col4: (unnamed); Col5: (unnamed)
D5	308	4	78	Col1: Group event; Col2: County; Col3: Company; Col4: (unnamed)
D6	309	6	44	Col1: Group event; Col2: Second team; Col3: Third team; Col4: Second driver country; Col5: Third driver country; Col6: (unnamed)

[5] https://docs.python.org/fr/3/library/tkinter.html.
[6] https://github.com/123rabida123/SINATRA-Annotation-Repair-Augmentation.
[7] https://zenodo.org/record/3518539#.YoOgK6hBwuU.
[8] https://www.aicrowd.com/challenges/semtab-2020.

To measure the efficiency of the *data repair* and *data augmentation* features of the SINATRA process, we used the following metrics proposed in [9,10]: Precision (P), Recall (R), and F-measure(F1).

(P), (R) and (F1) of the mapping between the datasets and the DBpedia KG are calculated using the following formula: where a *perfect annotation* refers to the annotation returned by our approach, which corresponds to the annotations of ground truths, a *submitted annotation* refers to the annotation returned by our approach and a *ground truth annotations* corresponds to the number of annotations in the Target Tables. We combined the predefined measures, which represent the harmonic mean between P and R to calculate F1.

$$P = \frac{(\#perfect\ annotations)}{(\#submitted\ annotations)} (1)\ R = \frac{(\#perfect\ annotations)}{(\#ground\ truth\ annotations)} (2)\ F1 = \frac{(2 * P * R)}{(P + R)} (3)$$

4.2 Evaluation Results and Demonstration

This section evaluates and demonstrates the performance of the SINATRA approach's features. For more details on the results of the evaluation, consulting our Github[9].

Regarding the evaluation of the ***data annotation*** feature, this phase of SINATRA is based on the MTab approach. Therefore, it automatically has the same performance as MTab. Table 3 below shows the results of the evaluation of the *data annotation* phase by the MTab approach [12].

Table 3. Evaluation results of the *data annotation* feature by MTab approach.

Feature	Data annotation (MTab approach)					
Tasks / Datasets	CEA		CPA		CTA	
Measures	F1	P	F1	P	F1	P
D1	1.0	1.0	0.881	0.929	0.850	0.852
D2	1.0	1.0	0.877	0.929	0.850	0.850
D3	0.983	0.983	0.844	0.845	0.833	0.833
D4	0.970	0.970	0.832	0.832	0.825	0.825
D5	1.0	1.0	0.987	0.975	0.929	0.933
D6	1.0	1.0	0.965	0.991	0.970	0.970

Our goal in this evaluation is to compare the results of the *data repair* and *data augmentation* phases (Table 4) with the results of the *data annotation* phase (Table 3) to show that they can correctly add the data (entities) and the missing columns.

Regarding the evaluation of the ***data repair*** feature, we re-based on the same datasets as above (Table 2). In this phase, the evaluation is based on two factors: The first factor (1): we removed some values from those datasets (Table 2) and calculated the performance of this phase. The second factor (2): we added the missing cell values into these datasets during the *data repair* phase. Table 4 below

[9] https://github.com/123rabida123/Datasets-and-Results-of-evaluation-SDA.

shows the performance results of the data repair phase based on the two factors mentioned. From the results of Table 4, we notice the results of the CEA task are reduced in the factor (1) because (R) is reduced (the removed URIs (entities) are in the CEA-targets). Based on the factor (2), we highlight that this phase can add missing data very nicely, where the CEA task has $F1 = \mathbf{1}$ of the datasets (D1 and D2). They have the same results as the *data annotation* feature. The CEA results are represented by the yellow color in Table 4. For the datasets (D5 and D6), the results of the CEA task have been reduced a little bit (from $F1 = 1$ in Data annotation to $F1 = 0.987$ in Data repair), because some URIs were not perfect or were not available in the CEA-targets. The CPA task is represented by magenta color and the CTA task is represented by cyan color, which have no variation in both factors. They have the same results as the *data annotation* feature in Table 3.

Table 4. Evaluation results of the *data repair* and *data augmentation* features.

Feature	Data repair											
Factors	Remove cell values (1)						Add missing cell values (2)					
Tasks	CEA		CPA		CTA		CEA		CPA		CTA	
Measures/ Datasets	F1	P	F1	P	F1	P	F1	P	F1	P	F1	P
D1	0.854	0.845	0.881	0.929	0.850	0.852	1.0	1.0	0.881	0.929	0.850	0.852
D2	0.852	0.874	0.877	0.929	0.850	0.850	1.0	1.0	0.877	0.929	0.850	0.852
D3	0.832	0.832	0.844	0.845	0.833	0.833	0.877	0.877	0.844	0.845	0.833	0.833
D4	0.812	0.813	0.832	0.832	0.825	0.825	0.834	0.836	0.832	0.832	0.825	0.825
D5	0.911	0.911	0.987	0.975	0.929	0.933	0.983	0.983	0.987	0.975	0.929	0.933
D6	0.943	0.945	0.965	0.991	0.970	0.970	0.987	0.975	0.965	0.991	0.970	0.970
Feature	Data augmentation											
Factors	Remove Column (1)						Add missing Column (2)					
Tasks	CEA		CPA		CTA		CEA		CPA		CTA	
Measures/ Datasets	F1	P	F1	P	F1	P	F1	P	F1	P	F1	P
D1	0.750	0.753	0.706	0.738	0.754	0.70	1.0	1.0	0.881	0.929	0.850	0.852
D2	0.751	0.754	0.706	0.738	0.736	0.739	1.0	1.0	0.881	0.929	0.850	0.852
D3	0.606	0.638	0.754	0.700	0.729	0.781	0.981	0.981	0.844	0.845	0.833	0.833
D4	0.632	0.634	0.707	0.717	0.729	0.781	0.943	0.945	0.844	0.845	0.833	0.833
D5	0.913	0.915	0.846	0.855	0.860	0.878	1.0	1.0	0.987	0.975	0.929	0.934
D6	0.845	0.845	0.846	0.855	0.833	0.835	0.996	0.997	0.939	0.987	0.956	0.956

Regarding the evaluation of the ***data augmentation*** feature, we re-used the same datasets as above (Table 2). The evaluation of the data augmentation feature is based on two factors: In the first factor (1), we removed every second column from those datasets (Table 2) and calculated the performance of this phase (without the second columns). In the second factor (2), we added the missing columns into these datasets. Table 4 above shows the performance results of this phase based on the two factors mentioned: whether this proposal is able to add exactly the deleted column in each dataset. From the results of the factor (1) in Table 4, we notice that the results of the CEA, CPA, and CTA tasks are more reduced because (R) is reduced (the removed URIs (entities) are in the targets). In addition, we notice from the results of the factor (2) in Table 4, that this feature is able to add the missing column very well, where the CEA, CPA, and CTA tasks of the datasets (D1, D2, and D5) have the same results as the *data annotation* feature in Table 3 are represented by the yellow color. The magenta

color represents the results of the CPA task, and the CTA task is represented by the cyan color of the datasets (D1, D2, D3, D4, and D5). They also have the same results as the *data annotation* feature. Thus, the *data augmentation* feature is perfectly able to add missing columns to the datasets. For the datasets (D3, D4, and D6), the results of the CEA task were slightly reduced, because some URIs were not perfect or were not available in the CEA targets.

5 Conclusion and Future Work

In this paper, we present an all-in-one and automatic approach, to be called SINATRA, that seeks to improve the usability of Tabular data through *Data annotation* (relying on an existing tool Mtab [12]) maps Tabular data elements to concepts in DBpedia KG to solve the issues of misspelling and missing or incomplete metadata. *Data repair* handles missing cell values in the Tabular data by fetching the corresponding concepts from DBpedia. *Data augmentation* allows the user to dynamically add the relevant columns and the corresponding cell values to the data. The evaluation results show that the SINATRA approach was able to annotate, repair, and augment the structured data.

In the near future, we plan to compare our proposal with other existing methods and tools, and extend it with additional features, such as (1) integrating additional knowledge graphs such as WikiData, LOV, Geonames and YAGO to improve the annotation, (2) evaluating the performance of our approach on other open datasets, (3) generating a RDF file of the annotated dataset to publish in Linked Open Data, and (4) providing a visualization graph to enhance the understanding on the relatedness between the concepts of the RDF file.

Acknowledgements. Rabeb Abida is funded by a CERUNA grant from the University of Namur, Belgium. Anthony Cleve is a professor in information system evolution at University of Namur, Belgium, where he heads the data-intensive system evolution lab. He is currently a visiting professor at Universitá della Svizzera italiana, Switzerland. Anthony is a member and former president of the PReCISE research center, and a member of the Namur Digital Institute (NADI). He co-edited the book "Evolving Software Systems", published by Springer in 2014.

References

1. Abdelmageed, N., Schindler, S.: JenTab meets SemTab 2021's new challenges. In: SemTab@ ISWC, pp. 42–53 (2021)
2. Abida, R., Belghith, E.H., Cleve, A.: An end-to-end framework for integrating and publishing linked open government data. In: 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 257–262. IEEE (2020)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52

4. Azzi, R., et al.: AMALGAM: making tabular dataset explicit with knowledge graph. In: SemTab@ ISWC, pp. 9–16 (2020)
5. Benedetti, F., Bergamaschi, S., Po, L.: Online index extraction from linked open data sources. In: Second International Workshop on Linked Data for Information Extraction (LD4IE), vol. 1267, pp. 9–20. DEU (2014)
6. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.: ColNet: embedding the semantics of web tables for column type prediction. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 29–36 (2019)
7. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.: Learning semantic annotations for tabular data. arXiv preprint arXiv:1906.00781 (2019)
8. Cremaschi, M., Avogadro, R., Chieregato, D.: MantisTable: an automatic approach for the semantic table interpretation. In: SemTab@ ISWC 2019, pp. 15–24 (2019)
9. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: SemTab 2019: resources to benchmark tabular data to knowledge graph matching systems. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 514–530. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_30
10. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K., Cutrona, V.: Results of SemTab 2020. In: CEUR Workshop Proceedings, vol. 2775, pp. 1–8 (2020)
11. Knap, T.: Towards odalic, a semantic table interpretation tool in the ADEQUATe project. In: LD4IE@ ISWC, pp. 26–37 (2017)
12. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: MTab: matching tabular data to knowledge graph using probability models. arXiv preprint arXiv:1910.00246 (2019)
13. Ongenae, F.: MAGIC: mining an augmented graph using INK, starting from a CSV (2021)
14. Roman, D., et al.: DataGraft: one-stop-shop for open data management. Semant. Web **9**(4), 393–411 (2018)
15. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
16. Zhang, S., Balog, K.: Ad hoc table retrieval using semantic similarity. In: Proceedings of the 2018 World Wide Web Conference, pp. 1553–1562 (2018)