Making Sense of Numerical Data - Semantic Labelling of Web Tables

Emilia Kacprzak[1,2], José M. Giménez-García[3], Alessandro Piscopo[1,2], Laura Koesten[1,2], Luis-Daniel Ibáñez[1], Jeni Tennison[2], and Elena Simperl[1]

[1] Electronics and Computer Science, University of Southampton, UK
{l.d.ibanez, e.simperl}@soton.ac.uk
[2] The Open Data Institute, UK
{e.kacprzak, alessandro.piscopo, laura.koesten, jeni}@theodi.org
[3] University of Lyon, UJM-Saint-Étienne, CNRS, Laboratoire Hubert Curien, France
jose.gimenez.garcia@univ-st-etienne.fr

Abstract. With the increasing amount of structured data on the web the need to understand and support search over this emerging data space is growing. Adding semantics to structured data can help address existing challenges in data discovery, as it facilitates understanding the values in their context. While there are approaches on how to lift structured data to semantic web formats to enrich it and facilitate discovery, most work to date focuses on textual fields rather than numerical data. In this paper, we propose a two level (row and column based) approach to add semantic meaning to numerical values in tables, called NUMER. We evaluate our approach using a benchmark (NumDB) generated for the purpose of this work. We show the influence of the different levels of analysis on the success of assigning semantic labels to numerical values in tables. Our approach outperforms the state of the art and is less affected by data structure and quality issues such as a small number of entities or deviations in the data.

Keywords: Semantic Labelling · Numerical Values · Linked Data

1 Introduction

Data is being generated on the web at an ever-increasing speed. Yet, most of this data is published in formats that are not machine-processable, hampering our ability to gain value from it. Whereas the Semantic Web has gained traction as a way to provide semantics and interoperability to data, its coverage is still limited: looking at Open Government Data, in 2016 only $\sim 2\%$ of datasets were published as linked data in the UK [1]. Most data is still published in non-semantic formats, especially in tables. In Open Data portals the majority of datasets are collected and published as CSV or Excel files, accompanied by metadata from vocabularies such as DCAT[4] or Schema.org[5]. Integrating tables

[4] `https://www.w3.org/TR/vocab-dcat/`, consulted on 1 May 2018.
[5] `http://schema.org/`, consulted on 1 May 2018.

in the Web of Data is useful for enriching structured knowledge bases (KB), improving search over data, or to enable question-answering systems to use larger corpora of information. Motivated by this, solutions to lift tabular data into the Semantic Web have been proposed. These solutions aim at solving the following scientific problem: given a table and a target knowledge base *KB*, return a mapping of columns to classes or properties in *KB*. However, in spite of numerical columns being the most popular column type in open governmental datasets [2], existing approaches focus mostly on mapping textual data [3–8]. This is also reflected in the benchmarks available for the problem. For instance, one of the most commonly used benchmarks, T2D [9, 10], contains only 12 of 1748 tables with numerical columns disambiguated to DBpedia properties. Previous efforts [11, 12] compare distributions of numerical values in columns with the distribution of literal values in a KB, matching (within a given certainty) columns to the numerical properties with the most similar distribution. However, these approaches use information surrounding the numerical column to assign a semantic label.

Inspired by Venetis *et al.* [6], who reported increased accuracy if a main (subject) column was identified, we introduce NUMER—an approach which uses the context of numerical columns to assign semantic labels to them. We leverage existing approaches for identifying the subject column of a table by matching textual columns to entities in a knowledge base. We propose using the subject column of the table to pick potential labels which are then matched against the numerical column. Each cell in a subject column is disambiguated to a concept (entity) in the target KB. The numerical values associated with the subject columns are subsequently examined following a composite approach: (i.) a *column level analysis*, which looks at their distribution in a column; (ii.) a *row level analysis*, which compares each of them to the values associated to the disambiguated entities in the target KB. As a result, we generate a ranked list of properties for each numerical column. By selecting a table-specific set of possible semantic labels based on the subject column we were able to narrow down the possible values in a KB to those that are likely related to the context of the table. The preselected semantic labels are than ranked according to their fit to data in a column. This reduces memory requirements (as only data related to those values needs to be processed), and may make it more suitable in cases where KB are large, diverse, or rapidly changing such as DBpedia or Wikidata.

To evaluate our approach, we created the benchmark NumDB [13]. This consists of tables with numerical values constructed from types and numerical properties from DBpedia. NumDB introduces two dimensions of benchmarking: first, it includes deviations in the values drawn from DBpedia to test the sensitivity of approaches to values that are not exactly the same as the ones in the target KB. Second, it considers versions of the same table with different number of entities, to test the accuracy of approaches when facing smaller versus larger tables. Our evaluation suggests that our approach, which includes both row and column levels of analysis, outperforms the state of the art in terms of sensitivity to value deviation and effectiveness on smaller tables. NUMER shows itself more adaptable for use in a real world scenario in terms of time and memory consumption, as it does not require to generate the background knowledge necessary for approach proposed by Neumaier *et al.* [11].

The paper is structured as follows: Section 2 introduces the semantic labelling problem. Section 3 presents related work in assigning semantic labels to tables. Section 4 presents our approach. In Section 5 we provide details about the experiment, specifically about benchmark, set-up, and evaluation. Finally, sections 6 and 7 discuss our findings and outline future work.

2 Problem Statement

In this section, we introduce definitions of the concepts used, define the problem statement of assigning semantic labels to numerical columns in tables, and introduce a running example.

Definitions: We define a **table** T as a collection of related data on a specific topic. A table consists of m rows and n columns represented by a $m \times n$ matrix. Each row in a table has the same structure and can be seen as a single record of related data. Columns in a table are of specific type depending on their content; possible column types are **Numerical and Textual Columns.** A numerical column is a column where more than 50% of cells contain at least one digit. We chose this definition to not rule out cells that contain units of measure (*e.g.*, 2 Kilometers) or dates. A column that is not numerical is considered textual. One column per table is a **Subject Column**. That is, a textual column which represents the main subject of the table and connects the other columns semantically through binary relations [6, 7, 10]. Those connections are represented through properties from a KB. The process of determination of subject columns is detailed in Section 3.

Problem Statement: Given a table T and a target knowledge base KB, for each numerical column in T, return the list of properties in KB that most likely correspond to the numerical columns, ordered by likelihood score.

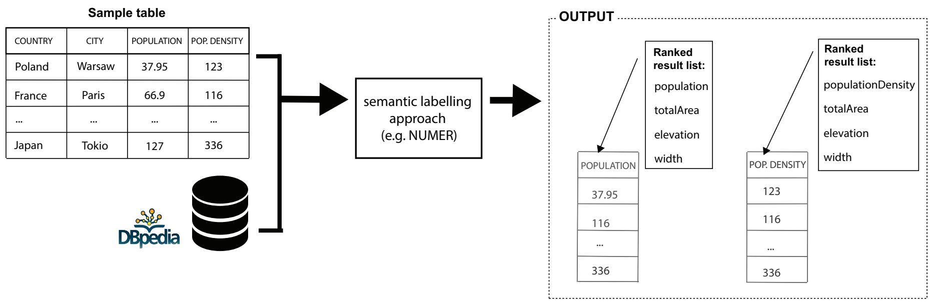

Fig. 1. Running example. A table with two textual and two numerical columns, with DBpedia as target KB.

Running example: Figure 1 shows an excerpt of a table from the T2D benchmark [14] with four columns: two textual (`Country` and `City`), and two numerical (`Population` and `Population Density`). The goal of the process is to produce a ranked list of labels for each numerical column based on the information generated from the subject column.

In this work we aim to disambiguate columns with numerical vales with use of the information from subject column in the table. We distinguish a list of scenarios that can be encountered when solving this problem. We look at the scenarios when the subject column is known and if the selection of subject column was not successful with existing approaches:

1. When the information on which textual column in the table is a subject column is known:
   (a) *Full match of numerical properties values* in the KB with numerical values in a table. This scenario is the most trivial and could be solved with basic matching techniques.
   (b) *Numerical values in the numerical column could deviate from the values in the KB.* Some values could be more distinct than others, some could be missing in the KB entirely. Our approach includes mechanisms to make the influence of the following problems negligible: the *Column Level Analysis* (presented in Section 4) compares the distribution of values in the numerical column against that of values from numerical properties, which helps when values are missing. We also take into account numerical properties connected to entities of each type that were recognised in the subject column, which helps minimise the influence of partially correct disambiguation of the cells in the subject column.
   (c) *Subject column cells can be disambiguated to a range of types* which could indicate a lack of consistency within the table or incorrect disambiguation of the cell value. Analysing tables per row allows us to compensate for the latter and detect the property the values of which are the closest to the values in the KB (Section 4 *Row Level Analysis*).
   (d) *Numerical columns can be properties of types different from the types of entities found in the subject column.* They can be properties of other textual columns (that are not the subject column) or not connected to any other column (their meaning could be identified from the context). This scenario is out of the scope of this work and for such cases alternative approaches could be used (e.g. Neumaier *et al.* [11]) which do not rely on additional information provided with the numerical column.
   (e) A property describing a numerical column could not be represented in the KB, in which case the approach will fail as the correct result of the disambiguation process is impossible to achieve.
2. Approaches for the detection of a subject column could fail, in which case the scenario is similar to the one described in 1.(d).

From these scenarios we can see that the knowledge of the subject column can be used as a basis for improving the accuracy and efficiency of labelling of numerical columns. We present the details of our approach in Section 4.

3 Related Work

Several approaches have been proposed to assign semantic labels to structured data. Some of them focus on tables embedded in web pages (in HTML `<table>` elements [7, 9]; others analyse any type of structured data with a specific focus on tabular or comma-separated data [4, 5, 8, 15]. Humans can be involved in the process to achieve better results [16]. Many approaches make use of content descriptions associated with the table (*e.g.*, information in an HTML page [6, 17, 18], headers within tables [7, 17]) or rely on data in textual columns within the table to assign semantic labels [10]. Others match table rows to KB entities, leaving out of the scope matching the table columns to KB properties [19, 20]. It is important to point out that only a few of the existing approaches propose solutions specifically targeted towards numerical values in structured data.

Numerical Values present different challenges than textual information when assigning semantic labels to columns in tables. An approach targeting specifically the problem of labelling numerical values in structured data was first shown by Ramnandan *et al.* [21]. They propose an algorithm that learns a semantic labelling function. The authors introduce a list of features, differentiated between those targeted at numerical and at textual values in structured data. They propose testing the distributions of numerical values corresponding to semantic labels based on the idea that the distributions of values for each semantic label are expected to be different (*e.g.*, the distribution of population of cities will be different from the distribution of population density). They used three different tests: the Welch's t-test, the Mann-Whitney U test, and the Kolmogorov-Smirnov test (KS test). Their results show that the latter achieved the best results. Neumaier *et al.* [11] and Pham *et al.* [12] used similar metrics. The solution proposed by Neumaier *et al.* [11] focuses exclusively on numerical values, and it is based on building a background knowledge graph from properties in DBpedia. They use DBpedia types and property-value pairs to structure their background knowledge with bags of numerical values which are later compared to numerical values from a data source using the KS test. However, creating and keeping the background knowledge is memory intensive. Pham *et al.* [12] introduce a number of additional features for different column types in their semantic labelling function. In addition to the KS test, they propose using metrics such as a modified Jaccard similarity for numeric data where the ranges of values are compared and measures which affect both textual and numerical values in a data source.

Our approach builds upon these by analysing numerical columns in two ways: first, in terms of the similarity (measured in terms of the KS test) of distribution of values with respect to those of properties in a target KB; second, by calculating the relative difference between numerical values in a column and numerical values of properties associated to entities of the type identified from the column that holds the main entities of the table. The main observation is that tables often include a column that identifies the entities described by the table (called the *subject column*), while the rest of the columns hold values of properties linked to those entities (or *objects*). Venetis *et al.* [6] reported 75% of the tables in their corpus of web tables had a single main subject column, and that the accuracy of semantic labelling increases when first determining a subject column.

Subject Column Identification To determine which of the columns in a table is a subject column, Venetis *et al.* [6] suggest two methods: taking the left-most column that is not a number or date column, or treating it as a binary classification problem. They propose learning a classifier for subject columns with features that are dependent on the name and type of the column and the values in different cells of the column. Wang *et al.* [7] and Ermilov *et al.* [10] proposed similar characteristics of the column: (1) the *connectivity* of a column (*i.e.*, how it is connected with other columns of the table by means of properties mapped to the KB) and (2) *support* of the column (*i.e.*, ratio of cells disambiguated to KB entities in the column). A combination of both connectivity and support is then used to determine which of textual column is the most likely the subject column. In this work we focus on labelling columns with numerical data, assuming a subject column has been previously identified.

Semantic Labelling Benchmarks The benchmark dataset created by Limaye *et al.* [3] comprises 400 tables mapped to DBpedia and YAGO at instance– and schema-level. Efthymiou *et al.* [19] introduce a benchmark including 485K tables from Wikipedia, which were mapped to DBpedia by leveraging the links in their label column. Neither of these two benchmarks was suitable for our experiment, which aims to map columns to properties. Instead, the dataset in [3] contains cell-to-entity mappings, while that in [19] row-to-entity. T2D [9] is a set of 1,748 tables[6] with schema and instance-level mappings to DBpedia. However, it does not contain a sufficient number of numeric columns to be suitable for our case (the large majority of disambiguated column were textual columns). Therefore, we decided to create a new benchmark, that we detail in Section 5.1.

4 Approach

Our approach comprises four stages, described below. We assume the availability of a tool that enables the match of textual cells to entities in the target KB, and of a tool that allows the identification of a subject column.

Preprocessing We preprocess the input table as follows: (1) Partition columns into *numerical* and *textual* columns. Following the definition in Section 2, we define a numerical column as a column that has $\geq 50\%$ numerical values. (2) From the subset of textual columns, select one subject column. This may be done following any the approaches described in Section 3. (3) Match each cell in the subject column to an entity in the target KB. (4) In numerical columns, we strip out cells containing non-numerical characters (*e.g.*, "2Km"), leaving only numerical values (*e.g.*, "2").

Figure 2 shows the output of preprocessing our running example. Columns *Country* and *City* were classified as textual, *Country* was identified as the subject

[6] `http://webdatacommons.org/webtables/goldstandard.html#toc0`

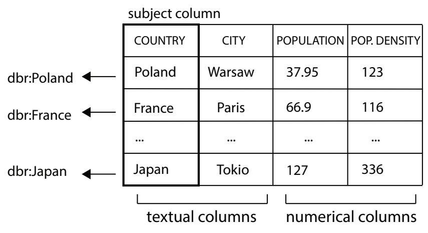

Fig. 2. Information resulting from performing preprocessing steps a table. column. All values in the subject column were disambiguated to a DBpedia entity. Columns *Population* and *Population Density* were classified as numerical.

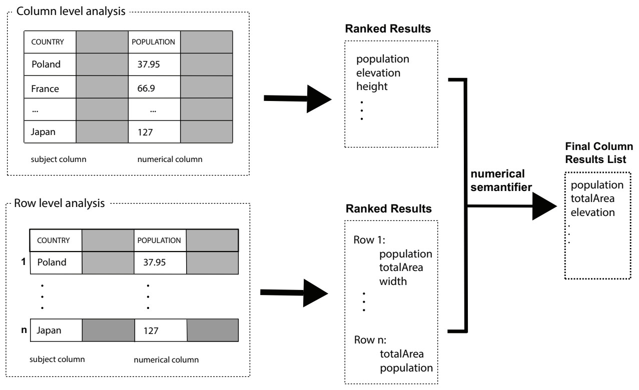

Fig. 3. An overview of the analysis stages in the semantification process.

Column Level Analysis Similarly to [11] and [12], we compare the distribution of values in numerical columns with bags of values from the target KB. However, instead of comparing to all bags of values in the target KB, we consider only the properties that have a semantic relation with the types of the entities identified by the subject column, hence reducing both number of comparisons and memory requirements. From the entities identified in the subject column in the preprocessing stage, we query the target KB for the list of all types associated to them. In our running example, a sample list of types could be: [*dbo:CapitalCity, dbo:Country, dbo:PopulatedPlace*]. Next, for each type, we generate a list of all its instances in the target KB. Then, for each entity, we select properties of *rdf:type owl:DatatypeProperty* associated to it. In our example, Poland, *dbo:PopulatedPlace* has the properties *population, area, existsFrom*, and *dbo:Country* has *population, area, populationDensity*. For each property, we select its associated values and compare them to those in the numerical columns using the two-sample KolmogorovSmirnov test [21], as shown in Equation 1.

$$D_{n,m} = \sup_x |F_{1,n}(x) - F_{2,m}(x)| \tag{1}$$

The output of the comparison is a list of properties for each numerical column, ordered by the probability given by the KS test. The output for the *population* column in our running example is `[(populationTotal, 0.98), (populationDensity, 0.44), (elevation, 0.14), (area, 0.08)]`.

Row Level Analysis Comparing value distributions does not necessarily result in meaningful matches. The size of a numerical column in a dataset and popularity of a specific property in the KB influences the accuracy of the results when comparing the distributions. To improve accuracy, we also analyse numerical values based on the context provided by the row they are in.

In this level of analysis, we perform the following steps for each row in the table: (1) From the entity disambiguated in the subject column at preprocessing stage, we query all properties of `rdf:type owl:DatatypeProperty` associated to it, together with their values. (2) Next, we compute the relative difference (Equation 2) between the value of the cell in the numerical column and the value of each of the properties collected in step 1. The intuition is that the property with the smallest relative difference is the right match for the value.

$$\text{Rel diff}(val_1, val_2) = \left| \frac{val_1 - val_2}{max(|val_1|, |val_2|)} \right| \tag{2}$$

In our running example (Figure 3), for the cell *Poland* 37.95 in the column *Population* we compute the relative difference with each of the values of the 33 numerical properties and the value in numerical column (here `37.95`). Producing a ranking of candidate labels per row ordered by decreasing relative difference.

(3) Finally, we generate a final ranking of candidate labels from the rankings generated in step 2. This is done by selecting all properties that appear in any of the lists. For each unique property we also assign its best relative difference value, intuitively giving more importance to a property that was able to exactly match one row. In case of a tie, we break it by computing the average position between all intermediate lists.

Numerical Semantifier In the last step, we create the final ranking by combining the outputs from the row level analysis (relative distance and average position) and the column level analysis (probability). Concerning the column level results list, a higher score represents a higher rank. As regards the row level analysis, a lower score means a higher rank. In order to merge the two analyses we order all outputs based on the closeness of the predictions to the highest ranking, independently of whether these represent outputs of row or column analysis (distance and probability). In case of a tie we prioritize the row level analysis labels over the column level analysis, as we identified in our evaluation row level analysis performs better than column level analysis on average.

Our final results list consists from predictions of semantic labels for a numerical column with their confidence score. We call the overall approach *NUMerical SemantifiER* (NUMER).

5 Evaluation

We evaluated our approach under two aspects: resource consumption (**Ev1.**) and accuracy in matching the correct property (**Ev2.**). For both, we compared against the approach developed by Neumaier *et al.* [11] as a baseline—which we refer to as *MultiLevelLabelling* (MLL). We generated a new benchmark for the purpose of our evaluation, which is described in the following section. All code used and results of the evaluation are available in a Github repository[7].

5.1 Benchmark

To evaluate our approach we needed a set of tables containing at least one textual subject column (*i.e.*, the column to which the values in the other columns refer) and one numerical column. We generated a benchmark by extracting tables from DBpedia. Each table has three columns: the first is the subject textual column; the second contains the DBpedia URIs corresponding to the entities listed in the first column (ignored by our algorithm); and the third numerical column.

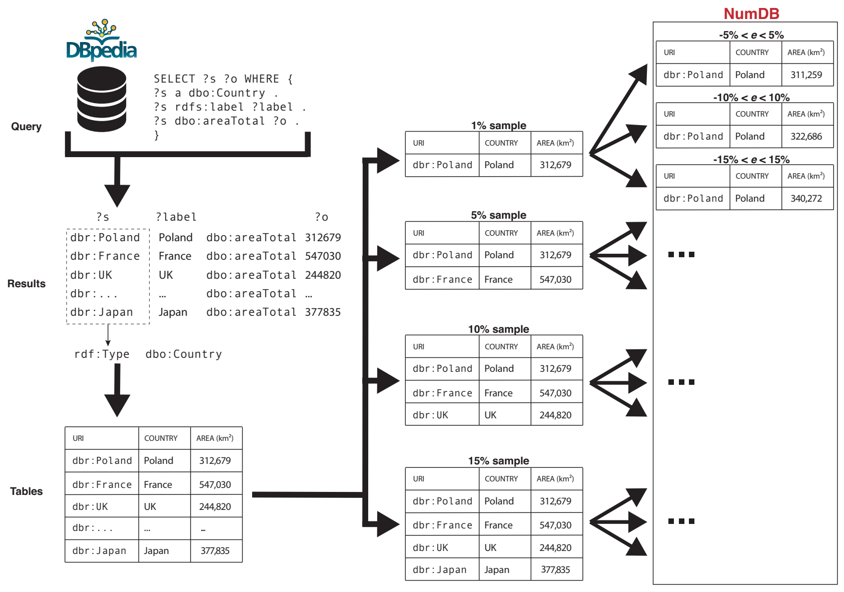

Fig. 4. Tables in the benchmark were created by extracting data from DBpedia and transposing it into tables. The figure represents the whole pipeline.

We extracted the tables according to the following process: first, we took a set of properties that could be mapped to numerical columns, namely those identified in [11]. These included the 46 most popular numerical DBpedia properties[8]. Second, we extracted the type of information (*i.e.*, the classes) of all subject entities for each property p_i in the property set and the number of entities of each type. For each property, we left out all classes whose entities comprised

[7] `https://github.com/chabrowa/semantification`

[8] 50 most popular properties excluding those linking to DBpedia internal ids.

less than 0.1% of all the property subjects, in order to exclude possible erroneous triples, and selected a number of random classes above this threshold, 10 when available, less otherwise. Subsequently, for each subset of classes we took all triples $p_i(i,o)$ for the corresponding property, where $type(i,C_j)$ for C_j in the class subset. Labels were collected for all entities and properties. All these steps were performed by querying the live DBpedia endpoint[9]. We transposed all the resulting triples into tables (see Figure 4). This produced a total of 389 tables, which we used to generate our benchmark. We created tables with different levels of sampling and introduced varying degrees of errors, with respect to the data subsequently used to disambiguate them, to also allow conclusions around the robustness of our approach in respect to inaccuracies in the data. We used four different sample sizes (*Very Small*:1% of all entities; *Small*:5%; *Medium*:10%; *Large*:15%). A statistical description of each sample can be seen in Table 1. For each sample size we generated three additional tables, to which a degree of error e of $-5\% < e < 5\%$, $-10\% < e < 10\%$, or $-15\% < e < 15\%$ to each value v was introduced. The total number of tables created was 3952, 247 for each combination of sample size and error degree. The dataset is available at [13].

5.2 Evaluation Results

For both resource consumption and accuracy we compared the performance of NUMER and MLL for each table size and degree of error in the NumDB benchmark. The resource consumption evaluation (**Ev1.**) included processing time and memory consumption. The accuracy evaluation (**Ev2.**) examined the percentage of correctly disambiguated columns, examining both the top 1 and the top 3 semantic labels on the ranked results list. Moreover, we generated scores for each level of analysis (*i.e.*, row and column) separately and compared it against the overall score, in order to gain a better overview of the influence of different levels of analysis on the results. Finally, we applied ANOVA to test for statistical significance between table sizes and between various degrees of error. We believe that this range of experiments was able to provide a better picture of the performance of our approach and to detect directions for further research.

Experiment Setup We deployed a SPARQL endpoint for DBpedia v.2016-4 using Virtuoso and AWS services[10]. We run the evaluation on a virtual machine with 6 cores and 66 GB of memory running Ubuntu Linux. MLL was evaluated using code provided by authors on an associated Github account[11].

Resource consumption We tested the overall performance of NUMER and MLL by measuring the **processing time** and **RAM** consumption to assign semantic labels to NumDB datasets without deviation. It is important to notice that both approaches differ significantly in their implementation. MLL requires to build a background knowledge, which in our experiment environment took 01:02:22 and 16.48GB of memory. Keeping a large amount of data in the memory

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

[10] `https://aws.amazon.com/marketplace/pp/B012DSCFEK`

[11] `https://github.com/sebneu/number_labelling`

allowed the MLL approach to analyse the tables with an average of 3 seconds per file. However, in the current set-up we used, following [11]'s evaluation, only 46 DBpedia properties. The resources required to build the background knowledge will grow with the number of properties used. As all the necessary information is selected based on the subject column, NUMER does not require prior set-up, resulting in a significantly lower memory consumption. However, requesting all of the information from the DBpedia endpoint at run time resulted in an average processing time of 13 seconds per table. Table 1 the processing times per set.

Table 1. Set statistics and processing time for NUMER and MLL (V.S - very small, S - small; M - medium; L - large set; Avg - average; S.dev - standard deviation).

Set	Statistics					MLL			NUMER		
	#rows	Median	Avg	S.dev.	*Δ*	Total	Avg	*Δ*	Total	Avg	*Δ*
V.S	11,456	79.5	137.69	127.76	-	555	2.256	-	2168	8.815	-
S	56,604	390	682.75	633.08	3.94	630	2.561	0.45	3147	12.793	0.14
M	113,054	808.5	1366.58	1265.55	1.00	816	3.317	0.14	3564	14.486	0.30
L	169,484	1255.5	2069.16	1899.65	0.50	936	3.915	0.11	3973	16.152	0.18

NUMER* – *Levels of analysis The row level analysis achieved better scores compared to the column level (Table 2). On the other hand, the combination of both levels was often more accurate of to the best performing scores of each level of analysis alone. We found varying levels of accuracy, the **row level analysis** performed consistently well compared to the column level analysis. The difference between accuracy scores by table size was not statistically significant, in contrast to a comparison by error deviation. The performance of the **column level analysis** differed significantly by table size but not by error deviation. The column level analysis used the KS test to assign semantic labels to bags of numerical values. The lower levels of accuracy of the column level analysis suggest a higher dependency on the deviation of the numerical values in a specific column than the row level analysis. Concerning NUMER, which integrated row and column level analysis, it was able to assign semantic labels with a higher degree of precision than the two approaches it is based on, selecting the correct semantic label in over 80% of cases regardless of sampling size or error rate.

Comparative evaluation We compared the performance of NUMER and MLL for all tables sizes and error degrees within the NumDB benchmark. NUMER consistently outperforms the latter, across all the dimensions in which the datasets change (Table 3). NUMER was not affected by variations in table sizes, whereas it was sensible to different degrees of error in the data. Accuracy for top 1 results drops 10.5% on average when introducing any error in the original data from DBpedia. On the other hand, MLL's performance significantly decreased according to both table size and error degree. Overall, the behaviour of MLL appears to be similar to that of our column level analysis, to which it had similar, yet higher, scores. Nevertheless, whereas MLL's performance rises as table size increase, column level analysis' scores are roughly constant for *small*, *medium*, and *large* table sizes, dropping only for *very small* tables.

Table 2. Percentage of correctly assigned labels within top 1 and top 3 results in a results list for NUMER approach split by level of analysis (V.S - very small set, S - small set; M - medium set; L- large set).

Set	top k	Row Level				Column Level				**NUMER**			
		V.S	S	M	L	V.S	S	M	L	V.S	S	M	L
0% dev	1	75.61	77.24	73.98	77.64	28.46	34.96	36.18	34.15	90.65	93.50	91.87	93.09
	3	93.50	95.93	93.50	95.53	40.65	55.28	56.10	54.88	93.50	96.34	93.90	95.93
5% dev	1	75.20	77.24	76.83	78.46	23.58	30.89	30.89	28.05	80.49	84.15	78.86	81.30
	3	93.50	95.53	92.68	95.53	35.77	50.00	52.03	48.78	93.50	95.53	93.09	95.93
10% dev	1	75.61	73.98	71.14	77.64	22.76	28.05	28.05	28.86	78.05	78.86	75.61	80.89
	3	93.09	95.12	93.09	93.90	37.40	48.37	48.78	47.56	93.09	98.37	93.50	94.31
15% dev	1	75.20	73.58	69.11	76.83	23.58	26.02	26.83	26.02	78.46	76.02	73.98	78.05
	3	93.09	93.90	92.28	94.31	36.59	46.75	47.15	45.12	93.09	94.72	93.90	94.72

Table 3. Percentage of correctly assigned labels within top 1 and top 3 results in a results list for NUMER and MLL. For MLL, we show results generated with average distance and majority vote (in brackets) (V.S - very small set, S - small set; M - medium set; L - large set).

Set	top k	**NUMER**				MLL			
		V.S	S	M	L	V.S	Sm	M	L
0%	1	90.65	93.50	91.87	93.09	40.65(34.96)	53.66(40.24)	55.28(40.24)	58.13(40.65)
	3	93.50	96.34	93.90	95.93	60.16(62.20)	73.98(73.98)	78.86(76.42)	77.24(75.20)
5%	1	80.49	84.15	78.86	81.30	39.43(30.89)	48.37(33.74)	50.41(34.15)	49.59(32.52)
	3	93.50	95.53	93.09	95.93	53.25(52.44)	64.63(63.01)	66.67(66.26)	65.45(64.23)
10%	1	78.05	78.86	75.61	80.89	39.02(30.89)	47.56(30.08)	47.15(31.30)	48.78(30.49)
	3	93.09	98.37	93.50	94.31	52.85(52.44)	62.20(59.35)	63.01(60.57)	62.60(60.98)
15%	1	78.46	76.02	73.98	78.05	38.21(30.89)	42.28(28.86)	45.12(27.64)	43.90(28.86)
	3	93.09	94.72	93.90	94.72	52.85(52.85)	58.94(56.91)	59.76(57.32)	59.76(59.35)

6 Discussion and Limitations

The experiments used to evaluate NUMER enabled us to gain a number of insights about its performance, which indicate directions for future research. NUMER was highly accurate in predicting semantic labels for numerical columns, outperforming the state of the art. MLL, the approach used as a baseline, achieves better scores over the column level analysis aspect of NUMER; however, comparing the combination or row and column level analysis, NUMER outperforms MLL consistently. In most cases, the row level analysis is responsible for most of the accuracy of the whole approach. Only when there is no deviation the integration of the column level analysis yields a significant increase in accuracy.

The results in Table 3 show a large difference in terms of performance between top 1 and the top 3 results. Additional scoring factors could be introduced based on other columns or additional textual information available together with the table besides the subject column, in order to improve the top 1 result. The correct semantic labels could be listed after the top 3 (*e.g.*, as a 4th semantic label in a result list), to provide users with a set of potentially valid semantic properties from which they could choose the correct one. This type of interaction may be applied to several contexts, *e.g.*, when generating a summary, or to create dataset to train a more sophisticated machine learning model to assign properties.

When comparing both approaches according to their time and memory consumption, NUMER requires longer time ($13s$) to analyse a single NumDB table than MLL ($\sim 3s$). However, it does not need to generate the background knowledge which, in the case of MLL, carries a cost in memory consumption and initialization time. We believe that this makes NUMER more suitable for use in a real-world scenario, dealing better with memory limitations and KB evolution.

Neumaier *et al.* [11] deliberately excluded any additional textual information in MLL. Conversely, NUMER requires textual information in the table to detect potential correct semantic properties. This makes our approach more dependent on textual content in the data: the lack of a subject column or multiple subject columns would likely have a negative impact on the results. A possible solution to that could be to combine NUMER and MLL depending on the presence of the subject column in the table. Moreover, we used textual information in the tables only to disambiguate subject columns to DBpedia entity types. In the future, methods to extract further semantic information from text should be explored, *e.g.*, finding relations between the extracted entities, in order to better understand how different elements in a table relate to each other which could further inform the task of assigning semantic labels to numerical values.

Limitations As with most approaches there are some limitations connected to this approach. First, a subject column might not be present, or several columns may be considered as subjects. Those scenarios present an additional layer of complexity which would require approaches that are independent of a subject column or other, more tailored solutions. Second, we evaluated our approach by using a set of synthetic tables extracted from DBpedia. Although we processed our tables to test our approach under different conditions, an evaluation in a real world scenario, *i.e.*, with tables found on web pages, should be carried out in the future to provide more solid indications about the applicability of NUMER.

7 Conclusion and Future Work

We presented NUMER—an approach to derive semantic representations of numerical values in tables. Approaches to add semantic meaning to numerical values are particularly valuable, as these represent the most popular column type in open governmental datasets [2]. We applied a column level analysis—based on the types of entities found in the subject column of the table and the related values in a KB—matched to the column values in the table. We further applied a row level analysis in which we matched the individual values in a row to the corresponding entity in the KB and approximate the closest numerical values linked to this specific entity. This enabled us to create a table-specific ranked list of potential semantic labels for numerical columns. Automatically inferring

the meaning of numerical values found in tables has the potential to significantly improve the discovery of structured data as it can add context to otherwise obscure values. We evaluated our approach using a benchmark (NumDB), created by us, and investigate the influence of the number of rows (percentage of entities of specific type in the KB) and the influence of (intentionally introduced) deviation in the data. We can see that both levels of analysis have a positive influence on the final score in our approach, outperforming the state of the art under the given conditions.
Existing benchmarks have shown not to be useful when the focus of evaluation in the task of assigning semantic labels is mainly on numerical columns. For instance, in T2D [9, 10], only 11 of 1748 tables contain numerical columns disambiguated to DBpedia properties. This indicates a need for new reliable benchmarks to test approaches such as MLL and NUMER, preferably in a real world scenario, without the bias of automatically generated tables. NumDB, although automatically generated, can be seen as a step in that direction. We believe NUMER can provide important context to numerical values in tables. This can, for instance, support search over tables on the web and make numerical columns discoverable even if their meaning is not explicitly available in a textual format [22]. We further see the potential of our approach to be used in recommendation systems for datasets by finding similar or semantically connected tables [23].

In future work we plan to extend this work by integrating multiple knowledge bases. We aim to further improve this approach by using additional information from the numerical columns such as currencies or units of measurements (*e.g.*, kilometre, million, percentage) that might be attached to the values for disambiguation. Correlating numerical columns to other, non-numerical columns, or column headers could further improve the results. This strategy would take advantage of instances in which, for example, one column is the percentage value of another column, or witch longitude and latitude in two separate columns.

Acknowledgements This project is supported by the European Union Horizon 2020 program under the Marie Skłodowska-Curie grant agreement No. 642795.

References

1. E. Kacprzak, L. Koesten, T. Heath, J. Tennison, Position paper: Dataset profling for un-linked data, in: Proceedings of the 3rd International Workshop (PROFILES), the 13th ESWC Conference, 2016.
2. J. Mitlöhner, S. Neumaier, J. Umbrich, A. Polleres, Characteristics of open data CSV files, in: 2nd International Conference on Open and Big Data, OBD 2016, Vienna, Austria, August 22-24, 2016, pp. 72–79. doi:10.1109/OBD.2016.18.
3. G. Limaye, S. Sarawagi, S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, PVLDB 3 (1) (2010) 1338–1347.
4. V. Mulwad, T. Finin, Z. Syed, A. Joshi, Using linked data to interpret tables, in: Proceedings of the First International Workshop on Consuming Linked Data, Vol. 665 of CEUR Workshop Proceedings, 2010.
5. Z. Syed, T. Finin, V. Mulwad, A. Joshi, Exploiting a web of semantic data for interpreting tables, in: Proceedings of the 2nd Web Science Conference, 2010.

6. P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, C. Wu, Recovering semantics of tables on the web, Proc. VLDB Endow. 4 (9) (2011) 528–538. doi:10.14778/2002938.2002939.
7. J. Wang, H. Wang, Z. Wang, K. Q. Zhu, Understanding Tables on the Web, in: Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings.
8. M. Taheriyan, C. A. Knoblock, P. Szekely, J. L. Ambite, A scalable approach to learn semantic models of structured sources, in: Proceedings of the International Conference on Semantic Computing.
9. D. Ritze, O. Lehmberg, C. Bizer, Matching HTML Tables to DBpedia, in: Proceedings of the International Conference on Web Intelligence, Mining and Semantics, 2015, pp. 10:1–10:6.
10. I. Ermilov, A.-C. N. Ngomo, TAIPAN: Automatic Property Mapping for Tabular Data, in: Proceedings of the 20th International Conference on Knowledge Engineering and Knowledge Management, 2016, pp. 163–179.
11. S. Neumaier, J. Umbrich, J. X. Parreira, A. Polleres, Multi-level semantic labelling of numerical values, in: International Semantic Web Conference, Springer, 2016.
12. M. Pham, S. Alse, C. A. Knoblock, P. Szekely, Semantic labeling: a domain-independent approach, in: International Semantic Web Conference, Springer, 2016.
13. A. Piscopo, E. Kacprzak, Numdb (2018). doi:10.6084/m9.figshare.6205814.v4. URL `https://figshare.com/articles/numdb\_0105\_zip/6205814/4`
14. D. Ritze, O. Lehmberg, Y. Oulabi, C. Bizer, Profiling the potential of web tables for augmenting cross-domain knowledge bases, in: WWW, ACM, 2016, pp. 251–261.
15. C. A. Knoblock, P. Szekely, J. L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan, P. Mallick, Semi-automatically mapping structured sources into the semantic web, in: The Semantic Web: Research and Applications, ESWC 2012, pp. 375–390.
16. I. Ermilov, S. Auer, C. Stadler, User-driven semantic mapping of tabular data, in: Proceedings of the 9th International Conference on Semantic Systems, ACM, New York, NY, USA, 2013, pp. 105–112. doi:10.1145/2506182.2506196.
17. M. D. Adelfio, H. Samet, Schema extraction for tabular data on the web, Proc. VLDB Endow. 6 (6) (2013) 421–432. doi:10.14778/2536336.2536343.
18. D. Wienand, H. Paulheim, Detecting incorrect numerical data in dbpedia, in: V. Presutti, C. d'Amato, F. Gandon, M. d'Aquin, S. Staab, A. Tordai (Eds.), The Semantic Web: Trends and Challenges, Springer International Publishing, 2014.
19. V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro, V. Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: International Semantic Web Conference, Springer, 2017, pp. 260–277.
20. C. S. Bhagavatula, T. Noraset, D. Downey, Tabel: Entity linking in web tables, in: Proceedings of the 14th International Semantic Web Conference ISWC, 2015, pp. 425–441. doi:10.1007/978331925007625.
21. S. K. Ramnandan, A. Mittal, C. A. Knoblock, P. A. Szekely, Assigning semantic labels to data sources, in: The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC Proceedings, 2015, pp. 403–417.
22. L. M. Koesten, E. Kacprzak, J. F. A. Tennison, E. Simperl, The trials and tribulations of working with structured data: -a study on information seeking behaviour, in: Proceedings of the CHI Conference on Human Factors in Computing Systems, 2017, pp. 1277–1289. doi:10.1145/3025453.3025838.
23. A. Goel, C. A. Knoblock, K. Lerman, Exploiting Structure within Data for Accurate Labeling Using Conditional Random Fields, in: Proceedings of the 14th International Conference on Artificial Intelligence (ICAI), 2012.