Typology-based Semantic Labeling of Numeric Tabular Data

Ahmad Alobaid [a], Emilia Kacprzak [b] and Oscar Corcho [a]
[a] *Department of Artificial Intelligence, Universidad Politécnica de Madrid, 28223, Madrid, Spain*
E-mails: aalobaid@fi.upm.es, ocorcho@fi.upm.es
[b] *Department of Electronics and Computer Science, University of Southampton, UK*
E-mail: e.kacprzak@soton.ac.uk

Abstract.
A lot of tabular data are being published on the Web. Semantic labeling of such data may help in their understanding and exploitation. However, many challenges need to be addressed to do this automatically. With numbers, it can be even harder due to the possible difference in measurement accuracy, rounding errors, and even the frequency of their appearance. Multiple approaches have been proposed in the literature to tackle the problem of semantic labeling of numeric values in existing tabular datasets. However, they also suffer from several shortcomings: closely coupled with entity-linking, rely on table context, need to profile the knowledge graph and the prerequisite of manual training of the model. Above all, they all treat different kinds of numeric values evenly. In this paper, we tackle these problems and validate our hypothesis: whether taking into account the typology of numeric data in semantic labeling yields a better solution.

Keywords: Semantic Labeling, Semantic Annotation, Levels of Measurements, Typology of Numbers, Fuzzy Clustering, Semantic Web

1. Introduction

The number of structured data published on the web is constantly growing thanks to initiatives such as the Open Data movement (e.g., data.gov[1]) and more specialized platforms (e.g., Kaggle[2], Figshare[3], data.world[4]). One of the most fundamental problems is enabling the understanding of the dataset content, which could be beneficial for different groups, for example, researchers working on the expansion of Knowledge Bases[5] (KB), such as DBpedia or Wikidata [1], for a data scientist who found a dataset and wants to use it for their knowledge discovery task [2], or for expanding meta-data describing a dataset which could be utilized further for data search (e.g., Google Dataset Search Engine [3]).

There are several of approaches focusing on the problem of understanding the semantic meaning of the content of datasets [4–8]. They focus on detecting semantic labels for specific dataset cells (such as [6]) or assigning semantic labels to the whole column (such as [4, 5]). However, the majority of the existing efforts focused mainly on textual columns with not much attention being drawn to numerical columns present in the dataset. Recently, the research focusing on the problem of assigning semantic labels to numerical columns in a dataset started to get traction. In the work of Mitlöhnert et al. [9], it was pointed out that numerical columns are the most popular column type among datasets from several open data portals.

The approaches targeting the annotation of numerical columns need to tackle different issues than the ones developed for textual columns; it can be harder due to the possible difference in measurement accuracy, rounding errors, the frequency of their ap-

[1] https://www.data.gov/
[2] https://www.kaggle.com/
[3] https://figshare.com/
[4] https://data.world/
[5] In this work, we use the terms "Knowledge Base" and "Knowledge Graph" interchangeably

pearance or different values being reported by different resources (e.g., population data). Multiple approaches tried to solve this problem, but they also suffer from several shortcomings: closely coupled with entity-linking (each number in the numeric columns has to match a numeric property of an entity in the knowledge graph), rely on table context (e.g., the URL of the page, the caption of the table, the surrounding text), need to profile the knowledge graph (e.g., create a model of the whole knowledge graph, build inverted index), and the prerequisite of manual training of the model. Above all, they treat different types of numeric data evenly.

In this work, we expand our recent work [8], where the semantic labeling of numerical values is performed using fuzzy clustering. Our previous work shows that it is possible to semantically annotate numerical columns in tabular datasets without applying object matching or entity linking techniques. However, since all numeric data were treated uniformly, there were cases where the approach mislabel some of the numeric columns. Hence, in this work, we expand the approach to take into account the typology of the numeric data. We hypothesize that taking into account the typology of the numeric data in the semantic labeling process results in better accuracy. We propose a typology of numbers based on the typology proposed by others [10–12] for the process of semantic labeling. We take different approaches to annotate each of them.

The contributions of this paper can be summarized in the following points:

- We introduce an extension of the typology of numerical values presented in the literature. This extension includes the concept of *sub-type*, where a type can be divided further into sub-types.
- We propose a way to detect the type of a list of numerical values based solely on its content, without external resources or context.
- We propose a new approach to label numerical columns in tabular data taking into account the type of numerical values in a column.

The rest of the paper is organized as follows. We begin by reviewing the typology of numbers in Section 2. We review the state-of-the-art in Section 3, and we explain the problem statement in Section 4. We introduce the typology of numerical values in Section 5 and show how we can detect each of the (sub-)types in Section 6. In Section 7, we show how we extract data from a knowledge graph and construct the model which we use for the semantic labeling in Section 8. We evaluate our approach in Section 9 and conclude the paper in Section 10.

2. Background: Types of Numbers

Mosteller and Tukey said: “Just writing in digits does not make a number”[11]. On the other hand, Stevens and Birkhoff [10] typology of numbers does not quite agree with this[6]. They propose four types of numbers: nominal, ordinal, interval, and ratio.

Nominal where digits are just like names, so they do not imply extra meanings like order. For example, if we have two names, John and Adam, we can not say that the name John is greater (or less) than the name Adam. An example of *nominals* are the digits printed on the shirts of football players: they only distinguish the players, but it doesn’t have a “greater than” or “smaller than” concept to it. So, the only operation available is to compare two nominals, whether they are same or not.

Ordinal type implies an order among a set of elements. So one appears before the other. There is the concept of greater than (or less than) between two elements. An example would be ordering the dishes in a menu from the most favorable to the least favorable, or differences in responses in satisfaction surveys between “very satisfied” and “somewhat satisfied” [13]. However, there is no regard to the difference between the two items. For example, if a person likes dark chocolate more than white chocolate ($darkchocolate > whitechocolate$), there is nothing about the difference between his likeness of dark chocolate and white chocolate. Such an operation could be referred to as “illegal” [10].

Interval scale is used to denote the increase or expansion in some way on a scale. A classical example presented by [10] is the use of two different scales to measure the temperature. We (humans) use Centigrade and Fahrenheit, and we have a way to transform the temperature from one scale to the other. The intervals are the same (e.g., the difference between 10 and 20 degrees Celsius is the same as the difference between 25 and 35 degrees Celsius) [13].

[6]The paper was written by Stevens, but Birkhoff helped complete the list of types. They refer to nominals as numbers, but they (along with Mosteller and Tukey) agree that numbers (or collection of digits) can carry different meaning, and hence the different types. Stevens approach the typology from the measurements points of view while Mosteller and Tukey view is more from a data analysis point of view.

Ratio scales refer to things we measure that contain a real zero such as *counts*, *fundamental*, and *derived* ratios. *Counts* represent the number of elements or occurrences (e.g., number of eggs in a basket). *Fundamental* ratio represents measurements taken directly (e.g., when the width of the table is taken directly using a measuring tape or a ruler). *Derived* ratio is the mathematical function of the magnitudes of a simple (fundamental) ratio measure (e.g., computing the speed of a car using a measured time and distance). The difference between *ratio* and *interval* is the existence of a true zero[7]. For example, when we talk about the temperature, there can be an agreement on a zero value (for example 0 in Celsius and 32 in Fahrenheit for the freezing point of water at 1 atmosphere unit), but it is not a true zero point (it does not mean a complete absence of temperature). Such a point is referred to as the "starting" point [11, 13].

Some measures are not considered by Stevens, such as the cyclical measures [12]. An example of such cyclical measures is the angle; Chrisman said: "the direction 359° is as far from 0° as 1° is" [12]. The added levels presented by Chrisman are: Log-interval, extensive ratio, cyclic ratio, derived ratio, counts, and absolute[8]. Log-interval is similar to the interval scale with the scaling happening on the exponent level. Extensive ratio is the same as the fundamental ratio explained by Stevens (e.g., the width of a book). Cyclic ratio is the same as the extensive ratio where it is bounded and repeat. Absolute is when a scale is predetermined and cannot be transformed while preserving the meaning.

Mosteller and Tukey have a similar categorization: names (nominal), grades and ranks[9] (ordinal), amounts (fundamental ratio), balances[10] (derived ratio), counts, counted fractions[11] (absolute). However, it lacks the followings: interval[12], log-interval, and cyclical ratio.

Types of numbers are also referred to as kinds of numbers, scales of measurement, and levels of measurement [10–13].

[7]also referred to as "real" and "absolute".

[8]Chrisman suggestion is to consider them as separate types

[9]ranks are numbered while grades are labels (e.g., A, B, ...)

[10]It is similar to log-interval but also is a ratio, so we consider it here a derived ratio

[11]It is not exactly equal as absolute; it is more general than counted fractions, but it is enough approximation for our purpose in this paper.

[12]Even though Mosteller and Tukey warns the reader when re-expressing zeros, they do not create a separate category for it

3. State-of-the-art

The topic of semantic annotation for tabular data has been of interest for search engine companies as well as for the semantic web community for several decades. Different techniques have been used to perform this task: graphical probabilistic models [14–17], linear regression [18, 19], decision trees [20], hierarchical clustering [5], and support vector machine (SVM) [15], among others.

These techniques need to be applied to computed information extracted from the data, which are referred to as "features" in machine learning. Cafarella et al. [18] use attribute correlation statistics computed from the crawled web documents and schema coherency score. Features like the number of hits on the tables [18], column types [14, 15], and text similarity [15, 19], and the relation between columns/entities [14–16, 21] are all used and shown to be the main drivers of high-quality annotation. [22] uses a long short-term memory (LSTM) network in order to obtain a semantic representation of each cell in the table. Statistical tests to compare distributions from different samples have also been adopted, especially for dealing with numerical data [5, 7, 19, 23].

From our review of the state-of-the-art, we can see that there are two kinds of learning sources (training sets). The first kind uses knowledge graphs such as YAGO [15] and DBpedia [4, 5] to create learning sets. Others are more focused on learning from scraped web pages which do not provide such ease to focus on a specific domain [14, 18, 20, 21]. Despite the fact that these approaches may be automatic or semi-automatic[13], some of them require manual actions (e.g., provide predefined conversion rules [4, 17], a blacklist of properties [5] to improve the accuracy and abbreviations resolution [4, 17] while others rely on experts to semantically annotate columns to semantic properties such as [24]. Approaches such as, [7, 25, 26] base their analysis of numerical columns on a context subject column - a column to which other columns in the table could be linked with knowledge base properties. We also found that they treat all numerical columns and their values in the same way, not taking into account the type of numerical values except for [25]. Oulabi et al. [25] divide numeric values into Quantity and Nominals. This is not really a typology,

[13]We are not referring here to the gold standards that are built manually or the semantic models that are constructed by domain experts.

it is to distinguish whether to treat the values as measurements (Quantity) or as text (Nominals). Their detection algorithm of the type of numerical data (Nominal or Quantity) relies on the data in the knowledge graph.

In this work, we first detect the type of numerical values as a way to improve the performance of semantic labeling without matching the exact numbers to a property of a matched entity, relying on an ontology, profiling knowledge graphs, manual elimination of properties, or tweaking of parameters (that is knowledge graph dependent).

4. Problem Statement

We define a *dataset* as a collection tables, and each table is composed of multiple columns. In this work, the terms *dataset*, *tables*, and *tabular data* refer to the same thing. Columns in a table may consist of numerical values, textual values, or a combination of both. In this work, we focus on *numerical columns* and exclude any textual context, such as the header of the column or any text within column cells. This results in each numerical column being represented as a collection of numerical values. The **problem statement** of this work could be summarized as follows:

Given a collection of numerical values of a specific type, a class describing the content of a table and a target knowledge base, return the list of properties in the knowledge base that most likely correspond to those numerical values, ordered by likelihood score.

Figure 1 outlines the inputs and outputs of the semantic labeling approach.

5. Typology of Numerical Columns

In this work, we adopt the typology presented by Stevens et al. [10] as a base, and we build on top of it. This is because it is more suitable for the detection of different types as we explain in Section 6. We extend two of the high-level types (*nominal* and *ratio*) into sub-types. We based our work on types discussed by Tukey [27], Mosteller et al. [11], and Stevens et al. [10].

Example

We present a table with numerical columns, and we assign to each column a type and a sub-type (Figure 2). The table is about military individuals. The first column contains the first names, so it is not a numeric column. The second column represents the "service name". Each person has a unique service name showing on their uniform. The 3rd column represents the heights in centimeters. After that is the column about the "sex" of the people, so males are denoted as 1 and 2 for females. The "Race rank" column represents the order of the winners of a race held for those people (e.g., 1 means the person came first, 2 means the person came second place). The sixth column represents the number of goals each of the players scored in a season. The 7th column contains the internal identifiers of the personal files (there is a file about personal information for each person). The following column contains the coldest temperature (in Celsius) of the water in which the corresponding person was able to swim. The last column represents the civil id for each person. We discuss further the construction of the civil ids later in this section (Figure 3).

5.1. Nominal

Nominals are labels that are composed of digits. They are used to distinguish between things represented by different labels. Such are used instead of names because it is easy to check for uniqueness compared to names as multiple people can share the same name [14].

It could be impossible to know whether given numerals in a column are nominal or not without having extra context. However, understanding how these numerals are generated can give us some insights. To ensure the uniqueness, some use a sequence of natural numbers (e.g., military units). Some starts from large numbers and the sequence would be, for example, organized as (90000, 90001, 90002, ...). We refer to such sets of numbers as *sequential*. In Figure 2, an example of *sequential* sub-type is presented in the column "Service number", where the values range from 9001 to 9009.

In other cases, the number is a combination of segments, where each segment has a meaning [28]. A segment of a number can mean the unit of the soldier (in the case of military personnel) [15], or the date of birth [28], sex, place of birth, and usually the last part of a such a nominal number ends with a random num-

[14]As the case of baseball https://en.wikipedia.org/wiki/Uniform_number_(Major_League_Baseball

[15]https://en.wikipedia.org/wiki/Service_number

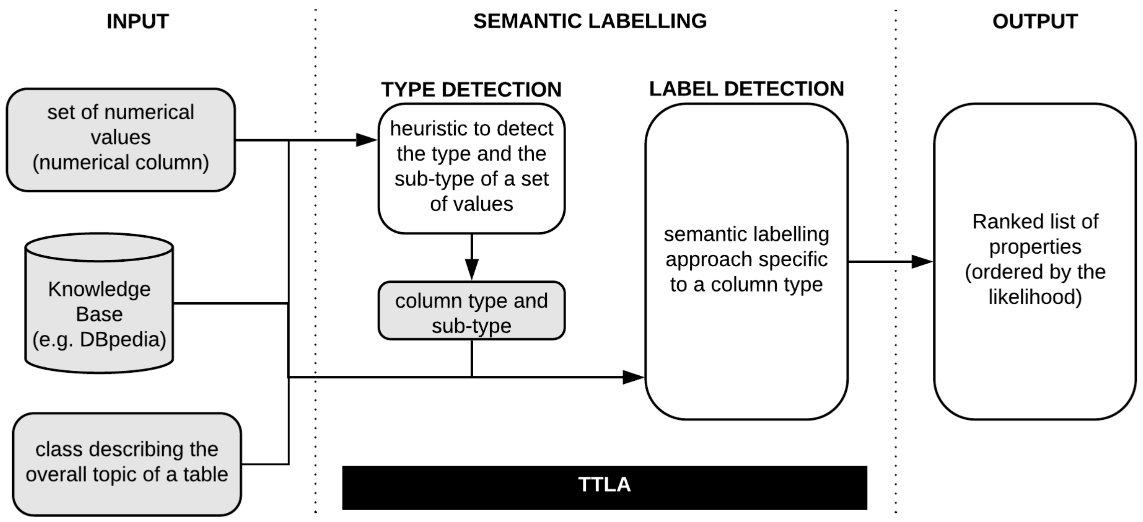
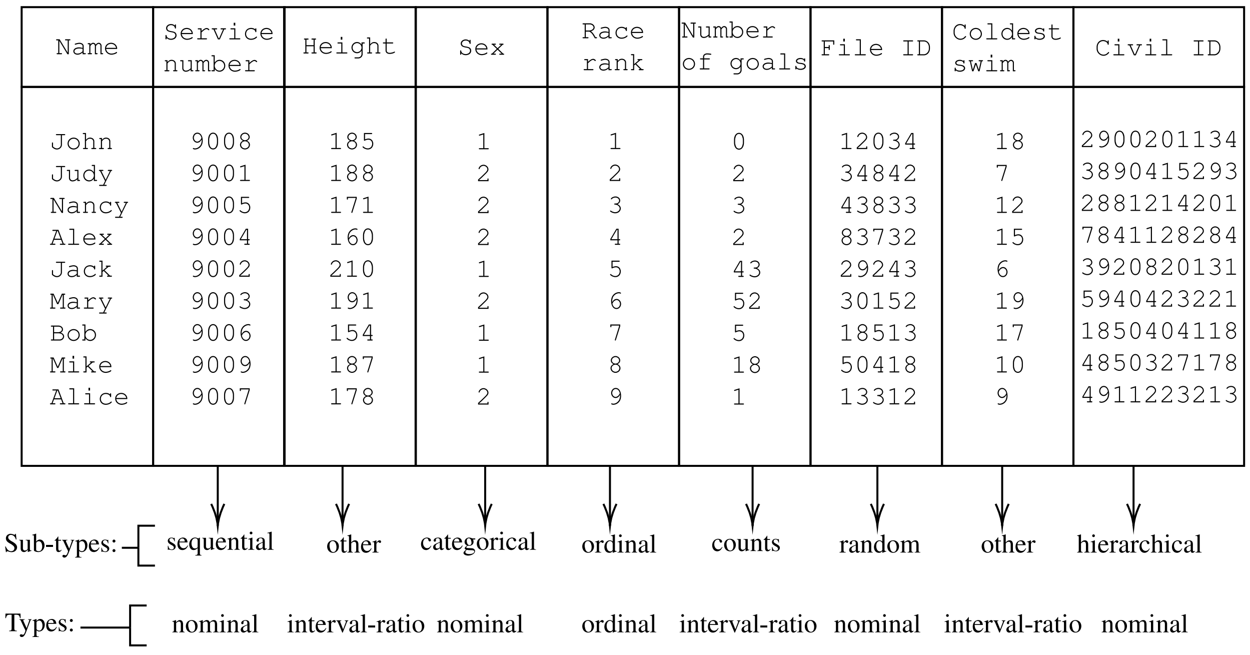

INPUT
SEMANTIC LABELLING
OUTPUT
set of numerical values (numerical column)
Knowledge Base (e.g. DBpedia)
class describing the overall topic of a table
TYPE DETECTION
heuristic to detect the type and the sub-type of a set of values
column type and sub-type
LABEL DETECTION
semantic labelling approach specific to a column type
Ranked list of properties (ordered by the likelihood)
TTLA

Fig. 1. Outline of inputs and outputs of the problem tackled in this work. Light gray highlights the inputs to the semantic labeling approach.



Name	Service number	Height	Sex	Race rank	Number of goals	File ID	Coldest swim	Civil ID
John	9008	185	1	1	0	12034	18	2900201134
Judy	9001	188	2	2	2	34842	7	3890415293
Nancy	9005	171	2	3	3	43833	12	2881214201
Alex	9004	160	2	4	2	83732	15	7841128284
Jack	9002	210	1	5	43	29243	6	3920820131
Mary	9003	191	2	6	52	30152	19	5940423221
Bob	9006	154	1	7	5	18513	17	1850404118
Mike	9009	187	1	8	18	50418	10	4850327178
Alice	9007	178	2	9	1	13312	9	4911223213

Fig. 2. An example of types and sub-types

ber or sequence [28]. Such is referred to as *hierarchical* (see Figure 3).

The third sub-type of nominal is the general case where a number represents a group, sometimes referred to as *categorical*. An example of such data can be seen in the column "sex" in Figure 2, where 1 is used to represent male and 2 to represent female.

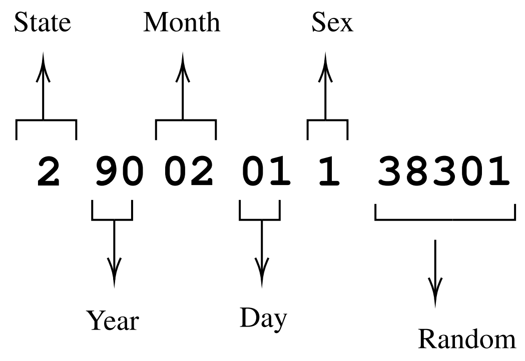


Fig. 3. An example of hierarchical sub-type

The fourth kind of nominal is something that is either randomly generated (like automatically generated ids in software), or they only make sense for the platform like addresses in memory (column “File ID” in Figure 2).

To summarize, we identify four sub-types of nominals: 1) sequential; 2) hierarchical; 3) categorical; and 4) random.

5.2. *Ordinal*

It is the same ordinal scale we explained in Section 2. It represents a rank with no regard to the difference. For example, it is concerned with the order of the winners (1st, 2nd, ...) but with no regards to how many seconds it took each of the runners to complete the race. It also does not regard on how much faster the 1st runner from the 2nd (speed difference). It is also not concerned about how much faster the 1st runner from the 2nd (speed difference).

5.3. *Interval and Ratio*

We introduce two sub-types: *counts* and *other*. Following [10], we define the sub-type *counts* (also known as cardinal numbers) as the number of instances or occurrences of something. An example of this is the number of goals scored, as shown in Figure 2.

In fundamental ratio, the numbers tend to have fewer digits in the fraction parts than derived ratios because they are measured, and the accuracy is bound by measurement tools while the derived ratio is a function which could produce more fractions (more digits after the decimal point). For example, if we have a circle with a radius of 0.13 (which is a fundamental ratio), and we want to calculate the area of this circle[16] (derived ratio). Even with π rounded to 3.14, the result is 0.053066. Despite this observation, it is just a matter of limitation that changes over time (as measurement tools get more advanced) and computers advance to handle such numbers. Furthermore, data published on the web does not tend to have many digits after the decimal point as we notice from the datasets we experiment on in this paper and it is common for people to round their numbers to a few digits after the decimal point[17]. Hence, we group fundamental ratio and derived ratio in a sub-type we refer to as “other”. An example of this sub-type is the human height (Figure 2).

For the cyclical ratio, some can fall under the fundamental ratio, while others can fall under the derived ratio (e.g., whether the angle is measured by a protractor or is the output of a function). Therefore, we add the cyclical ratio to the sub-type *other*.

Moreover, we group ratio and interval together despite the fact that they are different conceptually. But just looking at a zero; it is impossible to tell whether it is a “real” zero or just an agreement without extra information (which is the difference between ratio and interval). Since we can not tell them apart, we group them together. As the *counts* is distinguishable from the rest, we will put the interval in the *other* subtype. We show an example in Figure 2 (column “Coldest swim” represents the coldest temperature they can swim in).

6. Typology Detection

In this section, we present our approach to detect typology of numeric columns. The aim is to assign the (sub-)type of any given column. This step corresponds to the "TYPE DETECTION" box in Figure 1.

6.1. *Nominal*

In general all nominal numbers are expected to be natural numbers. Let X be the input collection of numbers,

$$\forall x_i \in X; x_i \in \mathbb{N}$$

Sequential nominal numbers are easier to detect if the data is complete. For example, if we have a list of numbers X which has 700 as the minimum value and

[16]$area = 0.5 * 2 * \pi * r^2$

[17]This observation is data source dependent and could not hold on some datasets.

923 as the maximum value, we check if this list is of the sub-type *sequential* by checking if the list X equal the natural sequence [700,923] [18] Y. It becomes tricky when we have missing values due to the selected population having something in common (e.g., sequences of solider for a sub-unit). The intuition that we follow is that if more than the square root of the numbers in the sequence Y are also in the original collection of numbers X, then we consider the collection of numbers as a sequential collection. So we consider a collection of numbers X sequential if the following equation is satisfied:

$$||X \cap Y|| > \sqrt{||Y||}$$

There is nothing magical about the square root; we can use a cube root or a simple percentage (e.g., 50%). In the end, it really depends on the kind of published data that we are dealing with (e.g., how much noise in the data). For example in Figure 2 in column "Service number", the numbers have the same number of digits, 4, and ranges from 9001 to 9009 (no missing values in the sequence), hence we consider them sequential. If there are two missing numbers, we will still consider this column sequential as $(7 > \sqrt{9})$.

Hierarchical nominal numbers are more difficult to detect. They can be easily confused with another sub-type of nominal and even non-nominal numbers. They tend to include a sequence in their composition. They also have the same number of digits, but this is also the case for the sequential nominal numbers. In order to detect hierarchical data in a column, we first check the number of digits; if it is the same in all the cells, then we consider it to be hierarchical if it is not sequential. This is if the values are unique; having duplicate values is strong evidence of non-hierarchical numbers. We have to admit that this detection method is not perfect, but it is an intuitive way to detect hierarchical nominal numbers. In Figure 2 in column "Civil ID", all the numbers have the same number of digits, and they fail the sequential test; hence, they will be considered hierarchical.

Categorical nominal numbers have some unique aspects compared to the other nominal numbers. They tend to have a large number of repetitions; the number of categories is an important signal to distinguish it from other categorical data. Another aspect of categorical numbers is that they are usually natural numbers. We consider a list of numbers X as categorical if they are nominal and the number of unique values U is much less << than the total population.

$$1 < ||U|| < \sqrt{||X||}$$

Similar to the sequential sub-type detection, other ways to interpret or execute << is cube root or log, but they might not be suitable for smaller datasets compared to the square root). We show in example of categorical data in column "Sex" in Figure 2. There are two distinct values ($U = 2$), so $(1 < 2 < \sqrt{9})$ is satisfied, and hence, the column will be considered categorical. We outline the algorithm to detect nominal categorical data in Algorithm 1.

```
Function isCategorical(objects):
    counts = ∅
    foreach object do
        if object in counts then
            counts[object] += 1
        else
            counts[object] = 1
        end
    end
    if $\sqrt{objects.length} > counts.keys.length > 1$ then
        True
    else
        False
    end
return
```

Algorithm 1: Detect Categorical Properties

In case there is only one single unique value ($U = 1$), we do not consider that categorical. We simply ignore that collection as extra knowledge would be needed to understand the meaning of this number.

Random nominal numbers are all of the remaining nominal numbers. But we have no way of detecting that, which makes sense because it is random. This kind of numbers is usually manually removed as discussed by [5, 24].

6.2. Ordinal

Ordinal scale is one of the easiest scales to detect. Generally, it is just a sequence of natural numbers starting from 1 until n (while n is the number of elements in the sequence). An intuitive way to detect them

[18]The natural sequence of [700, 923] is [700, 701, 702, 703, ... , 922, 923]

is to see whether the set of numbers X (what we want to examine) is equal to the list of numbers from 1 until the size of the list. Having a set of negative numbers or floats is a sign that the list of numbers we are examining is not ordinal. For example, if we have nine elements, they are ordinals if they range from 1 to 9 (see column "Race rank" in Figure 2).

6.3. *Ratio and Interval*

The numbers in the *counts* sub-type are positive by nature and do not have fractions.

One of the main aspects of counts is the way the distances between the numbers increase. Let us say we have a list of numbers of the sub-type counts X, if we order the numbers ascendingly

$$\forall x \in X : x_i < x_{i+1}$$

the distance between one number and the following one increases rapidly

$$\forall x \in X : (x_{i+1} - x_i) << (x_{j+1} - x_j)$$
$$j >> i; j, i \in [0, n-1]$$

where n is the number of elements in X [19].

To check if the numbers falling under the ratio-interval umbrella are simple counts, we use the followings:

$$1.5 * (Q_3 - Q_1) + Q_3 \leqslant P_{95} \tag{1}$$

$$\frac{(P_{95} - Q_2)}{Q_2} \geqslant \beta \tag{2}$$

P_{95} refers to the 95 percentile and Q_1, Q_2, and Q_3 refers to the first, second, and third quartiles, respectively. Eq. (1) checks whether P_{95} is considered an outlier or not. Following the intuition that counts subtype tends to increase a lot at the end (if ordered increasingly). We pick the P_{95} instead of the P_{100} to avoid possible noise or outliers (which can cause it to be falsely positive). Eq. (2), follows the same intuition that *counts* tend to have a large increase at the top (large) percentiles and here we check if it doubles ($\beta = 2$) the numbers in the middle (knows as median or Q_2). Note that in some cases, the optimal value of β is greater or less than 2, but we found it to be a good balance in our preliminaries exploratory tests. If Eqs. (1) and (2) are not satisfied, then we consider them of the sub-type *other*.

[19]This is inspired by the work of Tukey [27] in the analysis of counts data.

Table 1
Typology Detection Order

Order	Type	Sub-type	Rules
1	Ordinal	-	$\in [1, n]$
2	Nominal	Categorical	few unique numbers
3	Nominal	Sequential	same length* and $\in [min(X), max(X)]$
4	Nominal	Hierarchical	same length*
5	Ratio-Interval	Counts	natural and large increase in values†
6	Ratio-Interval	Other	anything else

* All elements have the same number of digits
† Growth similar to quadratic or more

6.4. *The Detection Order*

We start by checking whether a column is ordinal because it the most restrictive – it is the only one that checks for exact values. It tests whether the column is composed of natural numbers from 1 until n (n being the number of elements in the column).

For the second one, it should be one of the sub-types that checks for equal digits (hierarchical, sequential) or categorical. But the order here does not matter as categorical data contain a lot of duplicates, while sequential and hierarchical do not. We choose to check for categorical first. Then we check for sequential as the hierarchical detection method checks if the numbers have the same number of digits and are not sequential. For the fourth, we check if it is hierarchical.

The last check is the *counts* check; if they are not *counts*, then they will be considered of the sub-type *other* as we do not have a specific detection method for it. Note that for the *random* sub-type (under nominal), we do not have a detection method for it as mentioned earlier in this section. We show the order of detection and summarize the features in Table 1.

7. Model

In this section, we describe how we extract numeric values from a given knowledge graph. We use the extracted data to build a model that we use afterward to assign labels to numerical columns. This step (with the

Table 2
Model Features

Property URI	Numeric Type/Sub-type	Features
.../military-service-number	sequential	95000, ...
.../height	other	192.4, ...
...	...	...

Table 3
Labeling features per sub-type

Type	Sub-type	Features
nominal	sequential	trimean, tstd
nominal	categorical	# categories*, % categories†
ratio-interval	counts	trimean, tstd
ratio-interval	other	trimean, tstd

* The number of categories
† The percentages of each category

semantic labeling step in Section 8) correspond to the "LABEL DETECTION" box in Figure 1.

7.1. Data Extraction

In this step, we extract the numeric values from the knowledge graph.

To extract numeric properties from the knowledge graph, we use the following SPARQL query with a class URI. This will get all the properties for a given class.

```
SELECT distinct ?property WHERE {
?subject a <classURI>. ?subject ?property [].
} GROUP BY ?property
```

Then, for each property, we check whether more than 50% of the values (which are known as *objects*) are numeric. If so, then we consider this property as a numeric property which will be used with its values to build the model.

7.2. Model Construction

Before we build the model, we first need to detect the sub-types of the numbers for each property for the corresponding class. The model for each class will be in the format shown in Table 2.

Next, we show the different model construction methods for each numeric (sub-)type. We summarize the features for each sub-type in Table 3.

7.2.1. Nominal

The features used to build the model depend on whether the numbers are sequential, hierarchical, or categorical. We show how we compute features depending on different kinds of nominal numbers.

For the sequential kind, we use the trimean, which is more resistant to outliers than the mean [27]. We show the formula here:

$$trimean = \frac{Q1 + 2 * Q2 + Q3}{4} \tag{3}$$

We also use the standard deviation with the trimean instead of the mean. We refer to the standard deviation with the trimean as *tstd*.

For the categorical sub-type, we use the number of unique numbers (the number of categories) followed by the percentages for each category ordered ascendingly.

Matching hierarchical data type with a numeric property from the knowledge graph is complex. If we know the different sections and which digits represent them, we would be able to do more[20], but this is not the case, and hence we will not be able to semantically annotate them.

For the random sub-type, we ignore it because it is impossible to annotate such kind without extra evidence or elimination of other kinds, as it is random by definition.

7.2.2. Ordinal

This kind of data is common in tabular data, but we found that is very limited in knowledge graphs, which makes sense as such ordering usually done with some filtering (e.g., heights in zone A); hence, ordinal numbers are ignored.

7.2.3. Ratio and Interval

We can distinguish the *counts* sub-type from the *other* sub-type as they have different aspects that can be exploited to annotate numeric columns.

Counts Raw values of counts as-is are typically hard to understand and analyze [11, 27]. Annotating such data by machines is also difficult as the probability distribution or the numbers (counts) alone does not provide enough evidence to distinguish the data [11, 27]. Exact match will not be a solution as a single difference in counts or the data for the same events but from

[20]e.g., treating each section differently or treating each section as a dimension

a different year or reported by a different person could be very close but not exactly the same.

Nonetheless, exploratory data analysis techniques may help us here. Although they are generally meant to be used by humans to explore the data, we intend to automate this process. So, we transform the data to help us in understanding it; exploratory data analysis experts use square root ($\sqrt{X}$) and logs ($\log X$) to analyze the data; such transformation of data is referred to as "re-expression" of the data [11, 27]. Logs generally make the results too close while square roots tend to be a good balance between the logs and the raw values [27]; hence, we transform the raw data using the square root. After that, we use the trimean Eq. (3) and the trimean-standard-deviation (tstd).

Other For the sub-type *other*, we use the trimean [27] Eq. (3) and (*tstd*).

The use of trimean and *tstd* instead of the mean and the standard deviation reduces the effects of long-tailed distributions in the input data and the values of the numeric properties in the knowledge graph.

8. Semantic Labeling

In this section, we discuss the process of semantic labeling for the different types of numeric data where we treat each type differently to maximize accuracy. We introduce different features specific to each sub-type.

For the labeling task, we use the fuzzy c-means clustering technique [29]. We can divide it into two steps: the first one is computing the centers of the clusters; the second step is the classification (to assign semantic labels). Note that in the classical fuzzy c-means, it computes the cluster centers based on the data points minimizing the total error. In our case, we compute the clusters from the points (features) we extract from the knowledge graph (Section 7), which are the centers of the clusters. After that, we use the input data of the numeric column and compute the features, which is used to look for the closest cluster. As this is fuzzy clustering, it will belong to multiple clusters with a percentage of the belonging for each.

8.1. Centroids

In contrast to classical fuzzy c-means [29], we compute the clusters centers - which are also known as centroids - using the features presented in Section 7. Each numeric property extracted from the knowledge graph has a separate centroid; the value of each centroid is the set of features of the corresponding numeric property.

Table 4
Membership Vector Example

Notation	Meaning
m	weighting exponent to control fuzziness
d_{ik}	the distance between a datapoint k and a cluster center i
N	the number of data points
y_k	the data point value at index k
c	number of clusters
v_i	cluster at index i
u_{ik}	the membership value of a data point at index k to cluster i

The reason we compute the centroids that way is that we already know the clusters that correspond to the numeric properties, while in the classical fuzzy c-means the clusters are not known and need to be discovered by the algorithm.

8.2. Classification

Given a collection of numbers as an input (a numeric column), we detect the (sub-)type. Then, we retrieve a corresponding model (a model with the same class), and we strip it to include only the numeric properties that match the detected sub-type. Next, we compute the features of the input as in Section 7.

The computed features are then used as new data points. As the features are comma-separated, each value will represent a dimension (see Table 2 and 3). For each data point – which are the features computed for a given numeric column – the membership vector is computed as in Eq. (4) [21] with *fuzziness* ($m = 2$) as suggested by Bezdek et al. [29]. This membership vector indicates the percentage of belonging of a given column to each of the clusters (numeric properties) in the model.

For example, if we have three clusters: *dbp*:*height*, *dbp*:*weight*, and *dbp*:*waist*, a membership vector for a numeric column can be something like this $< 0.15, 0.80, 0.05 >$. This means that the given column belongs 15% to *dbp*:*height* cluster, 80% to *dbp*:*weight* cluster, and 5% to *dbp*:*waist* cluster. An output of the classification is the membership vector. For practical reasons, we order the results in descending order with

[21]This equation was introduced by Bezdek et al. [29] to compute the membership.

Table 5
Membership Vector Example

Cluster	Membership
dbp : weight	0.8
dbp : height	0.15
dbp : waist	0.5

Table 6
Typology Detection and Labeling

Type	Sub-type	Detect	Label
Nominal	Sequential	yes	yes
Nominal	Hierarchical	yes	no
Nominal	Categorical	yes	yes
Nominal	Random	no	no
Ordinal	-	yes	no
Ratio-Interval	Counts	yes	yes
Ratio-Interval	Other	yes	yes

the corresponding cluster and the membership vector as in Table 5. Note that the sum of the membership values for a given data point should be 1.

$$u_{ik} = \left(\sum_{j=1}^{c} \left(\frac{d_{ik}}{d_{jk}} \right)^{2/(m-1)} \right)^{-1}$$
$$\forall k, i : 1 \leqslant k \leqslant N; 1 \leqslant i \leqslant c \qquad (4)$$

9. Evaluation

In this section, we evaluate our hypothesis that detecting and treating numeric values differently depending on the kind of numeric values would results in higher precision than using a general technique (where different types of numbers are treated uniformly).

We define the set of experiments to evaluate the hypothesis; we describe the data we use, report the results of the experiments, and we finish this section with the discussion of the results.

9.1. Experiment

As we are dealing with different kinds of numbers, we divide the input data for each (sub-)type: *sequential*, *hierarchical*, *categorical*, *ordinal*, *counts*, and *other*. We do this manually, and then we apply the detection methods we reported earlier in this paper. This is done to measure the performance of our detection methods. Then, taking into account the knowledge that we have about the (sub-)types of numbers, we apply the labeling methods and report the precision, recall, and F1 scores. We do this for each dataset. The results of the detection is a type (e.g., nominal) and a sub-type (e.g., sequential). For the labeling, the output is the URI of the property of the numeric column that our algorithm determines to be more probable. The source code of the experiment and the data are published [30] (also available on GitHub [22]). It also includes the manual annotations and the (sub-)types for each of the numeric columns [23].

We summarize the (sub-)types that we are detecting and labeling in Table 6 with "yes" for the (sub-)types that we can detect or label and "no" for the ones we do not predict.

Next, we report the scores; we show the precision which we compute by dividing the number of correct prediction by the overall predicted ones Eq. (5). For the recall, we divide the number of correctly predicted over the total number of the same (class/type). Eq. (6). The final performance measure is the F1 measure, which is shown in Eq. (7). These performance measures are used for the detection and the labeling experiments.

$$precision = \frac{\#correct}{\#correct + \#incorrect} \qquad (5)$$

$$recall = \frac{\#correct}{\#correct + \#notfound} \qquad (6)$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \qquad (7)$$

9.2. Data

We use T2Dv2 dataset [31]: the only benchmark that we found in the literature that has different types of numeric columns. We use 124 numerical columns from that dataset that we were able to understand and for which we find a corresponding property in DBpe-

[22] https://github.com/oeg-upm/ttla
[23] The manual annotation and assignment of the (sub-)types have been done by one of the authors, and the process took around two working days.

Table 7
Typology in T2Dv2 dataset

Numeric Type	Sub-type	Percentage
Nominal	Sequential	0.008
Nominal	Hierarchical	0.0
Nominal	Categorical	0.0
Nominal	Random	0.048
Nominal	combined	0.056
Ordinal	-	0.04
Ratio-Interval	Count	0.387
Ratio-Interval	Other	0.234
Ratio-Interval	combined	0.621
Year	-	0.282

dia. We show the typology found in the dataset in Table 7. We can see that the most prominent lies under the ratio-interval type. The count type takes close to 0.4 of the total number of numerical columns. This has been reported in our previous paper [8] when talking about long-tailed distributions, which often caused by the data being condensed in one of the sides far from the mean. This was the primary cause of incorrect labeling in that work. We also notice here that we have a type called *Year*, which is not mentioned in the background nor in the typology that we adopt. This is due to its unique properties. The type Year is confusing as it can be thought of as a simple count from the birth of Jesus until a given moment. This could also be thought of as a nominal since it is often used to tag and name things produced, created, or occurred in a year without any regard of the counting aspect. Since it can be compared, some might argue that it is not nominal (nominal does not encompass the greater or less than operator [10]). Due to that, we will be ignoring it in our experiments.

9.3. Results and discussion

We first report the scores for the detection of the typology for each of the numeric columns in Table 8. For the *sequential*, we found that there is only one column, and we got it wrong. The reason is the noise; the file contains multiple tables with headers located below each other, which does not make the column look like an actual sequence(it looks like a subset of different columns merged together into a single column). From the benchmark T2Dv2, we did not find any hierarchical or categorical columns hence the N/A in the table. For the ordinal, we got high precision and recall applying a simple function. For the *counts*, we reached

Table 8
Typology Detection Scores

Numeric Type	Sub-type	Precision	Recall	F1
Nominal	Sequential	0.0	0.0	N/A
Nominal	Hierarchical	N/A	N/A	N/A
Nominal	Categorical	N/A	N/A	N/A
Nominal	Random	N/A	N/A	N/A
Ordinal	-	0.8	1.0	0.889
Ratio-Interval	Count	0.792	0.809	0.8
Ratio-Interval	Other	0.552	0.516	0.533

a precision score of approximately 0.8 and a similar recall. We inspected the wrongly detected ones, and we see that they do not look like counts. Counts tend to have a high variance as the number increases (so differences increase as mentioned in Section 6), which was not the case in a couple of the columns that were wrongly detected. For the last type, it is when the data do not match any of the other sub-types, as mentioned in Section 6, so there is no specific detection algorithm for it.

For the labeling part, we report the precision, recall, and the F1 score. We do that for different values of k, taking the top k properties that are the most probable. For $k = 1$ (taking into account only the top suggested property) the resulted recall is high ($\geqslant 0.8$) while the precision is not (> 0.4). Looking closely to the type with the lowest precision, we found that it is the *other* sub-type; we were not expecting the precision of it to be low.

We inspected the wrongly labeled properties and found that some are due to being labeled to knowledge graph specific/internal properties (e.g., *dbo:wikiPageID*) or wrong type detection. For example, the type of *areaOfCatchment* is wrongly detected as of sub-type *counts* when building the model (while it should have been *other*). Also, a couple of them wrongly point to properties related to years (e.g., *dbp:yearLeader*, *dbo:eruptionYear*). We found that years are one of the most difficult to work with relying only on the values [8, 16]. This could also be improved by having a better type detection, but may not prevent wrongly assigning year-related properties if the used typology does not have a (sub-)type *year*.

Not only the sub-type *other* has wrongly labeled properties, but also other types have mislabeled properties. We scrutinize and found that there are properties that are simply just wrong in the knowledge graph. For example, the *dbp:iosNumber* of a *dbo:Currency* is

Table 9
Typology Labeling Scores

k	Numeric Type	Sub-type	Precision	Recall	F1
1	Nominal	Sequential	1.0	1.0	1.0
	Nominal	Hierarchical	N/A	N/A	N/A
	Nominal	Categorical	N/A	N/A	N/A
	Nominal	Random*	N/A	N/A	N/A
	Ordinal	-	N/A	N/A	N/A
	Ratio-Interval	Count	0.83	0.957	0.889
	Ratio-Interval	Other†	0.486	0.8	0.604
	All	-	0.687	0.892	0.776
3	Nominal	Sequential	1.0	1.0	1.0
	Nominal	Hierarchical	N/A	N/A	N/A
	Nominal	Categorical	N/A	N/A	N/A
	Nominal	Random*	N/A	N/A	N/A
	Ordinal	-	N/A	N/A	N/A
	Ratio-Interval	Count	0.957	0.957	0.957
	Ratio-Interval	Other†	0.914	0.8	0.853
	All	-	0.94	0.892	0.915
5	Nominal	Sequential	1.0	1.0	1.0
	Nominal	Hierarchical	N/A	N/A	N/A
	Nominal	Categorical	N/A	N/A	N/A
	Nominal	Random*	N/A	N/A	N/A
	Ordinal	-	N/A	N/A	N/A
	Ratio-Interval	Count	1.0	0.957	0.978
	Ratio-Interval	Other†	0.943	0.8	0.866
	All	-	0.976	0.892	0.932
10	Nominal	Sequential	1.0	1.0	1.0
	Nominal	Hierarchical	N/A	N/A	N/A
	Nominal	Categorical	N/A	N/A	N/A
	Nominal	Random*	N/A	N/A	N/A
	Ordinal	-	N/A	N/A	N/A
	Ratio-Interval	Count	1.0	0.957	0.978
	Ratio-Interval	Other†	1.0	0.8	0.889
	All	-	1.0	0.892	0.943

* Unable to detect and fail back
† Include failed back (sub-)types

labeled as *dbp:width*; we looked it up and found that there is nothing called the width of a currency. Looking at the values, they look similar to the values of *dbp:iosNumber*, so it makes sense why it was confused with it.

As the *k* increases and takes more properties into account, we notice the significant score increase, especially for the precision of the sub-type *other*; it rises from 0.486 to 0.914. This means that those correct properties are still in the top suggested ones for the most part and the precision increase as we increase *k*.

We have carefully checked the result for recall for the sub-type *sequential*. This may be misleading because there was only one column with that type and the algorithm got it correctly; so the two possible outcomes were 0 or 1. For the other types that have N/A, this is due to the lack of these types in either the input dataset or the knowledge graph. An exception to that is sub-type *random*; the reason is that we do not have a way to detect a such (sub-)type. As a result, the sub-type *random* is reported with the sub-type *other* (combined).

Table 10

Compare Labeling Scores

k	Approach	Precision	Recall	F1
1	TTLA	**0.687**	0.892	0.776
	FCM	0.34	-	-
	Random	0.0004	-	-
3	TTLA	**0.94**	0.892	0.915
	FCM	0.55	-	-
	Random	0.0012	-	-
5	TTLA	**0.976**	0.892	0.932
	FCM	0.83	-	-
	Random	0.002	-	-
10	TTLA	**1.0**	0.892	0.943
	FCM	0.91	-	-
	Random	0.004	-	-

We compare our approach with our previous work [8] as we did not find any other approach reporting the classification of numeric columns separately for the only two publicly available tabular datasets with numeric columns. We denote our approach in Table 10 with TTLA (which stands for Tabular Typology-based LAbeling) and our previous approach with FCM as denoted in the other paper. We also include the scores of getting a correct property randomly from [8]. The scores of our approach (TTLA) for $k = 1$ achieve twice the precision as the previous work (FCM). TTLA continue to out-perform FCM as we increase k. We can see a clear significant improvement of precision. The recall is not reported in the previous work, so we put the sign $-$ in the corresponding cell, and we report the recall to allow comparison in future work.

10. Conclusion and Future work

In this paper, we introduce a typology of numeric data taking into account the task of semantic labeling. We show that taking into account the typology of numeric data and using such information to perform semantic labeling results in better performance. We also found out that there are (sub-)types that are under-represented in the existing benchmarks like (e.g., *hierarchical* and *categorical*).

As for the future work, we plan to extend the benchmark to include the under-represented (sub-)types. We also would like to explore the idea of having the sub-types as part of the features and use it to reduce the effect of the incorrect detection of (sub-)types.

References

[1] Y.A. Sekhavat, F.D. Paolo, D. Barbosa and P. Merialdo, Knowledge Base Augmentation using Tabular Data, in: *LDOW*, 2014.

[2] E. Kacprzak, L. Koesten, L.-D. Ibáñez, T. Blount, J. Tennison and E. Simperl, Characterising dataset search - An analysis of search logs and data requests, *Journal of Web Semantics* (2018).

[3] N. Noy, M. Burgess and D. Brickley, Google Dataset Search: Building a search engine for datasets in an open Web ecosystem, in: *28th Web Conference (WebConf 2019)*, 2019.

[4] D. Ritze, O. Lehmberg and C. Bizer, Matching html tables to dbpedia, in: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, ACM, 2015, p. 10.

[5] S. Neumaier, J. Umbrich, J.X. Parreira and A. Polleres, Multi-level semantic labelling of numerical values, in: *International Semantic Web Conference*, Springer, 2016, pp. 428–445.

[6] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro and 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.

[7] E. Kacprzak, J.M. Giménez-García, A. Piscopo, L. Koesten, L.-D. Ibáñez, J. Tennison and E. Simperl, Making sense of numerical data-semantic labelling of web tables, in: *European Knowledge Acquisition Workshop*, Springer, 2018, pp. 163–178.

[8] A. Alobaid and O. Corcho, Fuzzy Semantic Labeling of Semi-structured Numerical Datasets, in: *European Knowledge Acquisition Workshop*, Springer, 2018, pp. 19–33.

[9] J. Mitlöhnert, S. Neumaier, J. Umbrich and A. Polleres, Characteristics of Open Data CSV Files, in: *2016 2nd International Conference on Open and Big Data (OBD)*, 2016, pp. 72–79. doi:10.1109/OBD.2016.18.

[10] S.S. Stevens et al., On the theory of scales of measurement, *American Association for the Advancement of Science, Science, New Series, Vol. 103, No. 2684* (1946).

[11] F. Mosteller and J.W. Tukey, Data analysis and regression: a second course in statistics., *Addison-Wesley Series in Behavioral Science: Quantitative Methods* (1977).

[12] N.R. Chrisman, Rethinking levels of measurement for cartography, *Cartography and Geographic Information Systems* **25**(4) (1998), 231–242.

[13] D. Lane, J. Lu, C. Peres, E. Zitek et al., Online statistics: An interactive multimedia course of study, *Retrieved January* **29** (2008), 2009.

[14] A. Goel, C.A. Knoblock and K. Lerman, Exploiting structure within data for accurate labeling using conditional random fields, in: *Proceedings on the International Conference on Artificial Intelligence (ICAI)*, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, p. 1.

[15] G. Limaye, S. Sarawagi and S. Chakrabarti, Annotating and searching web tables using entities, types and relationships, *Proceedings of the VLDB Endowment* **3**(1–2) (2010), 1338–1347.

[16] M. Taheriyan, C.A. Knoblock, P. Szekely and J.L. Ambite, Learning the semantics of structured data sources, *Web Semantics: Science, Services and Agents on the World Wide Web* **37** (2016), 152–169.

[17] M. Zhang and K. Chakrabarti, Infogather+: Semantic matching and annotation of numeric and time-varying attributes in web tables, in: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ACM, 2013, pp. 145–156.

[18] M.J. Cafarella, A. Halevy, D.Z. Wang, E. Wu and Y. Zhang, Webtables: exploring the power of tables on the web, *Proceedings of the VLDB Endowment* **1**(1) (2008), 538–549.

[19] M. Pham, S. Alse, C.A. Knoblock and P. Szekely, Semantic labeling: a domain-independent approach, in: *International Semantic Web Conference*, Springer, 2016, pp. 446–462.

[20] Z. Syed, T. Finin, V. Mulwad and A. Joshi, Exploiting a web of semantic data for interpreting tables, in: *Proceedings of the Second Web Science Conference*, Vol. 5, 2010.

[21] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao and C. Wu, Recovering semantics of tables on the web, *Proceedings of the VLDB Endowment* **4**(9) (2011), 528–538.

[22] K. Nishida, K. Sadamitsu, R. Higashinaka and Y. Matsuo, Understanding the semantic structures of tables with a hybrid deep neural network architecture, in: *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[23] S. Ramnandan, A. Mittal, C.A. Knoblock and P. Szekely, Assigning semantic labels to data sources, in: *European Semantic Web Conference*, Springer, 2015, pp. 403–417.

[24] A. Merono Penuela, Refining Statistical Data on the Web, *Ph.D. thesis, Vrije Universiteit Amsterdam* (2016).

[25] Y. Oulabi and C. Bizer, Extending cross-domain knowledge bases with long tail entities using web table data (2019).

[26] Z. Zhang, Effective and efficient semantic table interpretation using tableminer+, *Semantic Web* **8**(6) (2017), 921–957.

[27] J.W. Tukey, Exploratory data analysis. 1977, *Massachusetts: Addison-Wesley* (1976).

[28] A.S. Lunde, The person-number systems of Sweden, Norway, Denmark, and Israel (1980).

[29] J.C. Bezdek, R. Ehrlich and W. Full, FCM: The fuzzy c-means clustering algorithm, *Computers & Geosciences* **10**(2–3) (1984), 191–203.

[30] A. Alobaid, E. Kacprzak and O. Corcho, TTLA source code, 2019. doi:10.5281/zenodo.2619307.

[31] D. Ritze, O. Lehmberg and C. Bizer, T2Dv2 Gold Standard for Matching Web Tables to DBpedia, 1999. http://webdatacommons.org/webtables/goldstandardV2.html.