Multi-level Semantic Labelling of Numerical Values

Sebastian Neumaier[1], Jürgen Umbrich[1(✉)], Josiane Xavier Parreira[2], and Axel Polleres[1]

[1] Vienna University of Economics and Business, Vienna, Austria
juergen.umbrich@wu.ac.at
[2] Siemens AG Österreich, Vienna, Austria

Abstract. With the success of Open Data a huge amount of tabular data sources became available that could potentially be mapped and linked into the Web of (Linked) Data. Most existing approaches to "semantically label" such tabular data rely on mappings of textual information to classes, properties, or instances in RDF knowledge bases in order to link – and eventually transform – tabular data into RDF. However, as we will illustrate, Open Data tables typically contain a large portion of numerical columns and/or non-textual headers; therefore solutions that solely focus on textual "cues" are only partially applicable for mapping such data sources. We propose an approach to find and rank candidates of semantic labels and context descriptions for a given bag of numerical values. To this end, we apply a hierarchical clustering over information taken from DBpedia to build a background knowledge graph of possible "semantic contexts" for bags of numerical values, over which we perform a nearest neighbour search to rank the most likely candidates. Our evaluation shows that our approach can assign fine-grained semantic labels, when there is enough supporting evidence in the background knowledge graph. In other cases, our approach can nevertheless assign high level contexts to the data, which could potentially be used in combination with other approaches to narrow down the search space of possible labels.

1 Introduction

With the uptake of the Open Data movement a large number of tabular data sources become freely available comprising a wide range of domains, such as finance, mobility, tourism, sports, or cultural heritage, just to name a few. The published data is a rich corpus that could be mapped and linked into the Web of Data, but RDF and Linked Data still remain too high an entry barrier in many cases, such that "3-star Open Data" (cf. http://5stardata.info/) in the form of tabular CSV data remains the predominant data format of choice in the majority of Open Data portals [19]. Connecting CSV data to the Web of Linked Data involves typically two steps, that is, (i) transforming tabular data to RDF and (ii) mapping, i.e. linking the columns (which adhere to different arbitrary

P. Groth et al. (Eds.): ISWC 2016, Part I, LNCS 9981, pp. 428–445, 2016.
DOI: 10.1007/978-3-319-46523-4_26

schemata) and contents (cell values) of such tabular data sources to existing RDF knowledge bases. While a recent W3C standard [18],[1] provides a straightforward canonical solution for (i), the mapping step (ii) though remains difficult.

Mapping involves to semantically label columns by linking column headers or cell values to either properties or classes in ontologies or instances in knowledge bases, and to determine the relationship between columns [17]. For the semantic labelling, most approaches so far rely on mapping textual values [16,20,21]; these work well e.g. for HTML/Web tables which have rich textual descriptions, as they are published mainly for human consumption. However, in typical Open Data portals many data sources exist where such textual descriptions (such as column headers or cell labels) are missing or cannot be mapped straightforwardly to known concepts or properties using linguistic approaches, particularly when tables contain many numerical columns for which we cannot establish a semantic mapping in such manner. Indeed, a major part of the datasets published in Open Data portals comprise tabular data containing many numerical columns with missing or non human-readable headers (organisational identifiers, sensor codes, internal abbreviations for attributes like "population count", or geo-coding systems for areas instead of their names, e.g. for districts, etc.) [9]. We verified this observation by inspecting 1200 tables collected from the European Open Data portal and the Austrian Government Open Data Portal and attempted to map the header values using the BabelNet service (http://babelnet.org): on average, half of the columns in CSV files served on these portals contain numerical values, only around 20 % of which the header labels could be mapped with the BabelNet services to known terms and concepts (cf. more details in our evaluation in Sect. 6.3). Therefore, the problem of semantically labelling numerical values, i.e., identifying the most likely property or classes for instances described by a bag of numerical values remains open.

Some early attempts focus on specific "known" numerical datatypes, such as longitude and latitude values [3], or – more generally – on classifying numerical columns using (manually) pre-labelled numeric value sets [11]. To the best of our knowledge, so far no unsupervised approaches have been devised for semantic labelling of numerical value sets. Additionally, the latter approach by Ramnandan et al. only assigns a single predefined semantic label, corresponding to a "property" per column. In the context of RDF, we deem such semantic labelling insufficient in (at least) two aspects: (a) We do not only need to map columns to properties, but to what we will call "contexts", that is property-domain pairs. (b) Since, given the variety and heterogeneity of Open

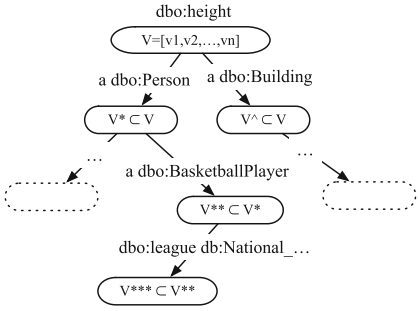

Fig. 1. Hierarchical background knowledge

[1] Or, likewise with RDB2RDF direct mapping [2], the basis of [18].

Data, it is likely we cannot rely on a manually curated, pre-defined set of semantic labels. Therefore, there is a need to build a hierarchical "background knowledge graph" of semantic labels in an unsupervised manner, cf. Fig. 1. As an example for (a), we do not only want to label a bag of numerical values as *height*, but instead we want to identify that the values represent the *heights of basketball players who played in the NBA*, or that the values represent the *heights of buildings*.

Even if we cannot identify such precise labels, we still want to assign the most likely contexts the values belong to, e.g. *height of a person*. To this end, and in order to achieve (b), we automatically generate a hierarchical background knowledge base of contexts from DBpedia. Different than previous approaches that assign a single label to a bag of values, we assign different labels/contexts, with different confidence values. This way, our approach could potentially be combined with textual labelling techniques for further label refinement, which is left for future work. In this particular paper, we focus on the following concrete contributions:

1. We propose a hierarchical clustering over an RDF knowledge base to build a background knowledge graph containing information about typical numerical representatives of contexts, i.e., grouped by properties and their shared domain (subject) pairs, e.g. city temperatures, peoples ages, longitude and latitudes of cities.
2. We perform a k-nearest neighbours search and aggregate the results of semantically label numerical values at different levels in our knowledge graph.
3. We evaluate our approach by cross-validating over a sample of DBpedia data generated from the most widely used numeric properties and their associated domain concepts.
4. We test our approach "in the wild" on tabular data extracted from Open Data portals and report valuable insights and upcoming challenges which we have to tackle in order to successfully label data from the Open Data domain.

In the remainder of this paper, after an overview of related works (Sect. 2), we describe our overall approach (Sect. 3). Next, we present the construction of the background knowledge graph from DBpedia in Sect. 4, as well as the actual semantic labelling of a column (i.e., a bag of numeric values) in Sect. 5. Finally, we present the evaluation results of the efficiency of different background graph construction strategies and our experiments with attempting to find matching columns in Open Data, Sect. 6. We conclude with a summary and ideas for future work (Sect. 7).

2 Related Work

There exists an extensive body of research in the Semantic Web community to derive semantic labels for attributes in structured data sources (such as columns in tables) which are used to (i) map the schema of the data source to ontologies

or existing semantic models or (ii) categorise the content of a data source (e.g. a table talking about politicians, i.e., in our case mapping the rows of a table into classes). The majority of these approaches [1,5,11,12,17,20,21] assume well-formed relational tables, rely on textual information, such as table headers and string cell values in the data sources, and apply common, text-based entity linkage techniques for the mapping (see [24] for a good survey). Moreover, typical approaches for semantic labelling such as [1,20,21] recover the semantics of Web tables by considering as additional information the, again textual, "surrounding" (section headers, paragraphs) of the table and leverage a database of class labels and relationships automatically extracted from the Web. Note, that in contrast to our concept of "context" of a column, the labels here are one-dimensional. In summary, the main focus of all these works is on textual relations inside the tables and in their surroundings. Techniques for recovering numerical relationships are often left for future work. As for used techniques, while these are out of the scope of our paper, many advanced textual entity recognition and linkage techniques are implemented in the Babelnet system [10], as we highlighted in the previous section these techniques are not necessarily applicable to a large portion of (numerical) Open Data. In contrast, our approach assumes that we only have a bag of numerical values available, in the worst case lacking any other rich textual information.

Most closely related to our efforts is the work by Ramnandan et al. [11], where the authors proposed to semantically label tuples of attribute-value pairs (textual and numerical). The semantic labelling of numerical values is achieved by analysing the distribution of the values and compare it to known and labelled distributions given as input by using statistical hypothesis testing. In contrast to their approach, we build a knowledge hierarchy and annotate sources not only with a single label but with a possible type and shared property-object pairs. Also complementary to our efforts is the work of Cruz et al. [3] which focus on detecting geolocation information in tables and apply heuristics specifically for numerical longitude and latitude values.

Outside the area of semantic labeling as such, but as an inspiration for our approach, the authors of [6,22] developed approaches to detect natural errors/outliers in RDF knowledge bases and automatically clustered candidate sets from the RDF knowledge base they want to analyse by grouping numerical values of a selected property by their types. We use a similar approach to build our background knowledge: we also group the subjects (and their corresponding values) by their types. However, we use a more fine-grained notion of "type", not only considering named classes but also "subtypes" defined in terms of shared property-object pairs.

While our present work explicitly focuses on instance sets labeling in the absence of a schema, previous work that addressed the automatic labeling problem using different combinations of instance and schema matching are relevant and will be considered in future extensions of our work [8,13,23].

3 Approach

Next, we outline the steps of our approach of finding the most likely semantic label and to determine the context in which a bag of numerical values are derived. In the following we formally define our notation and state the problem.

We denote a bag of numerical values annotated by a given label and context description $<l, c>$ as $\mathcal{V}^{<l,c>} = \{v_1, v_2, \cdots, v_n\}$, with $v_i \in \mathbb{R}$. Similar to [11], we define a semantic label l as an attribute of a set of values, which can potentially appear in different contexts. In this work, the semantic label l is a property from an ontology. However, this could be generalised. The concept of context description corresponds to a set of attribute-value pairs which explain/describe the commonalities of the values in $\mathcal{V}^{<l,c>}$. As such, one can assume that the set of input values $\mathcal{V}^{<l,c>}$ are the result of applying a query over a knowledge base ($\mathcal{V}^{<l,c>} = \mathcal{Q}(KB) = \{v_1, v_2, \cdots, v_k\}$) with the semantic label and the set of attribute value pairs as filter attributes of the query. For instance, the following SPARQL query returns the set of values labelled with *height* and sharing the attribute-value pair *a basketball player*:

```
SELECT ?v WHERE {[a dbo:BasketballPlayer] dbp:height ?v.}
```

Numerical values for a semantic label can appear in different contexts. For instance, values can represent the *height* of a building, mountain or a person. Even further, we might find values representing the height of basketball players that played in the NBA. We model this observation in form of a tree for each label l. The root node in such a tree corresponds to the set of all values which fulfill the property l. The remaining nodes of the tree represent further semantic information for this values, i.e., a shared context in the form of attribute-value pair. Edges in the tree are subset-relations between these values, directed from the superset to the subset. For instance, considering the semantic label *height*, the root node could have child nodes corresponding to the context *a mountain* and *a person*.

The background knowledge can be constructed in an either top-down or bottom-up approach The former starts with the root node of the graph and then detects subsets while the latter starts with leaf nodes which are then combined into parent/super nodes. The top-down approach is suitable for building the context graph from RDF knowledge bases and requires to start with a set of entities which are described by several attribute-value pairs. Next, we can group such entities by attributes which have numerical values, and then detect subgroups of entities with shared attribute-value pairs. We will show in the next section how we can build the background knowledge graph from an RDF knowledge base.

The bottom-up approach is more suitable for building the background knowledge from a set of CSV files. We first find a set of annotated numerical value triples $\{(v_1, l_1, c_1), (v_2, l_1, c_2), \cdots, (v_n, l_m, c_n)\}$, each consisting of a set of numerical values v_i, a label l_j and a context c_i. An input triple (v, l, c) can be extracted from a numerical column which was either manually or automatically annotated with semantic labels (e.g. based on the column header). The possible context

Table 1. Example table

name	capacity	city	country
Ernst Happel...	50865	Vienna	Austria
Franz Horr Stadium	13400	Vienna	Austria
Red Bull Arena	32000	Salzburg	Austria
...	...	...	...

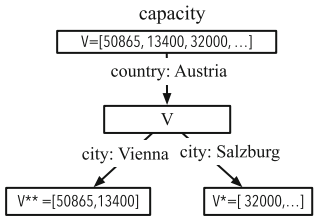

Fig. 2. Resulting tree

information can be modelled from column headers, the author or title of the table, or shared attributes within the table. For instance, take the example table in Table 1 and the numerical column `capacity`, as context we could extract that the numerical values describe an attribute of entities which are of type *football stadium*. Further, all values share the attribute-value pair *country: Austria*. Additionally, we could build a subset of values with the common context *city: Vienna* and another subgroup with the context *city: Salzburg* (cf. Fig. 2). The resulting background knowledge can be exploited by machine learning algorithms or statistical methods to predict the most likely label and context for a given bag of input values. We will outline how we apply a nearest neighbour search approach to derive the most likely label and context pair for a set of values in Sect. 5.

4 Background Knowledge Graph Construction from DBpedia

In this section, we outline our automatic top-down approach to build a background knowledge base from RDF data. To do so, we execute the following steps:

1. We extract all RDF properties which have numerical values as their objects and group the subjects by their numerical properties. These properties are used as labels. We derive the list of RDF properties which have numerical values as their range; the following SPARQL query could be used, cf. [6], however, we note that this query does not return results on the live DBpedia SPARQL endpoint due to timeouts:

   ```
   SELECT ?p, COUNT(DISTINCT ?o) AS ?cnt
   WHERE {?s ?p ?o. FILTER (isNumeric(?o))} GROUP BY ?p
   ```

   Another approach would be to directly query the vocabularies if we know that the RDF KB contains OWL vocabulary listing all datatype properties. We resorted to just filtering triples of the DBpedia dump with numeric objects, sorting them by property and counting via a script.
2. Next (in another pass/sorting), we collect/group by subjects in the different property groups the values of the numerical properties l. For "typing" of these subjects we collect property-object pairs – what we call context – for which

the object is an RDF resource (an IRI); this includes `rdf:type`-*Class* triples, but also others, e.g. `dbo:locatedIn-dbr:Japan`.

3. Next, we also extract and materialise the OWL class hierarchy for the *Class*es. This can be done directly by extracting the `rdfs:subClassOf` hierarchy from the DBpedia ontology for these *Class*es; we will use this *type hierarchy* to further enrich our background graph collecting contexts.

After grouping the entities by the selected context labels we construct our background knowledge graph as follows: An abstraction of our graph is depicted on the left hand side of Fig. 3: the graph consists of multiple trees, each tree corresponding to a property. The root node of such a tree is labelled by the property and contains the bag (i.e., multiset) of all numerical values of this property.

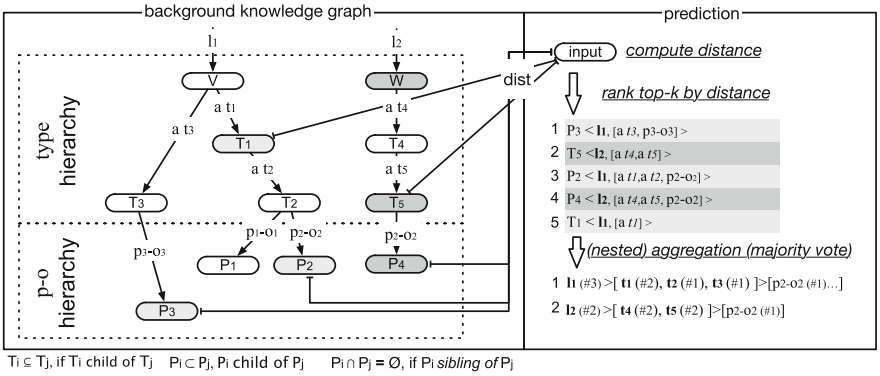

Fig. 3. background knowledge and prediction

4. The first "layer" of our knowledge graph is the so-called type hierarchy which represents the `rdfs:subClassOf` relation for all available types of the triples for property l from the *type hierarchy*. Since subjects can be of more than one type, the sibling nodes in this layer can share values from the same triples. In order to not keep too fine grained, rare classes, we filter by discarding types with less than δ instances (e.g., property-class combinations with less than 50 instances).
5. Next, we construct the second layer, termed *p-o* hierarchy for the identified non-`rdf:type` property-object pairs to further refine out context structure, beyond classes, using a divisive hierarchical clustering approach. We start with one node/group and split/compute sub-contexts recursively as we move down the hierarchy, to further refine the type hierarchy. In order to decide how to split a node, we impose the following requirements for possible candidates:
   (a) **constrain property-object:** we use the same constraint as [6] that subjects in a candidate node share the same property-object pair.

(b) **constrain size:** again, in order to avoid too fine-grained subdivision, the size of a candidate node has to be larger than 1 % of the parent node size (or, resp. larger than δ) and smaller than 99 % of the parent node size.

Once the set of possible sub-contexts is computed, we sort the candidates by their distance to the parent node in descending order. Details on the distance measures used to compare bags of numerical values are given in Sect. 5.1. To guarantee a high diversity as well as disjointedness of the sub-contexts within the hierarchy, we select the candidate with the biggest distance first, and then subsequently the non-overlapping sub-contexts from the list with decreasing parent distance. Additionally, the disjointedness requirement also helps to limit the number of sub groups. We recursively perform the above steps for the new selected groups. Consequently, shared property-object pairs of a node on the *p-o* hierarchy are encoded in the path to the resp. *p-o* node.

Node type terminology: Regarding terminology, we refer to the *exact type* of a context graph node as the lowest type node in the path to a *p-o* node. For instance, considering node P_3 in our example in Fig. 3, the exact type would be T_2. As a *super type*, we consider all type nodes on the path between the exact type node and the *p-o* node (e.g. T_1 would be a super type of node P_3). Eventually, the *root type* of a node, is the highest type node on the path to the *p-o* node (e.g. T_1 is the root type of P_3).

5 Prediction Approach

We use nearest neighbours classification over our background knowledge graph to predict the most likely "semantic context" for a given bag of numerical values. Given an input bag, we compute the distance between the values to all context nodes in our background knowledge graph and return the resp. contexts in ascending order of distance. Ideally, the node with the closest distance is the most likely semantic context/description for the input values. However, obviously numerical values for different types and properties might share the same value range and distribution and so we cannot even expect that the correct semantic description is always the top ranked result. As such, we also provide aggregation functions for predicates, type and *p-o* nodes over the top-k results. The idea is similar to the K-nearest neighbour classification for which the classification of an object is based on a majority vote over the top-k neighbour contexts.

5.1 Distance Measures

An important part for any prediction algorithm, be it based on machine learning or statistical methods, is the distance measure to determine how closely related two items (e.g. feature vectors) are. We consider two distance measures, namely (i) the euclidean distance between two feature vectors and (ii) the distribution similarity between two bags of numerical values.

Euclidean Distance Between Descriptive Features. The first distance function is the euclidean distance between two numerical n-dimensional feature vectors. For our use case we consider the following features for the vectors:

- *min and max value:* The range of minimum and maximum values is an important feature which allows us to easily discard "out of scope" labels or contexts. For instance, the heights of humans might have a maximum range of 213 cm which distinguishes it from buildings which have much higher max height.
- *5 % and 95 % quantile*: Due to the fact that minimum and maximum values as features are prone to outliers and errors in the set of values we also consider quantiles and inter-quantile ranges, e.g., using 5 %- and 95 %-quantile instead of min and max as features in a feature vector [6].
- *Additional descriptive statistics (mean, stddev):* Additionally, descriptive features such as the mean and the standard deviation of a set of values give better results for values which are within the same range but follow different distributions.

Distribution Similarity. Another distance measure is the similarity of two distributions of numeric values. This approach was already successfully used in a similar setup by Ramnadan et al. [11]. The authors also showed in their evaluation that the Kolmogorov-Smirnov (KS) test performs best for this particular setup compared to tests such as Welch's t-test or Mann-Whitney's U-test.

Kolmogorov-Smirnov (KS) distance: The KS test is a non-parametric test which quantifies the distance between two empirical distribution functions with the advantage of making no assumptions about the distribution of the data. As a distance measure between two samples, the KS test computes the KS-statistic D for two given cumulative distribution functions F_1 and F_2 in the following way:

$$D = \underset{x}{sup}|F_1(x) - F_2(x)| \tag{1}$$

where *sup* is the supremum of the distances. If two samples are equally distributed, i.e., the two bags hold the same numeric values, then the statistic D converges to 0.

5.2 Aggregation Function

As in the K-nearest neighbour classification, we also aggregate the top-k nearest neighbours by their properties, types and property object pairs. This allows us to classify the input values at several levels:

Before we apply the specific voting function, we aggregate the neighbours for the following different levels:

- `property level`: aggregation of the top-k neighbours by their properties
- `exact type level`: aggregation of the top-k neighbours by their exact type
- `root type level`: aggregation of the top-k neighbours by their root type

- `all types level`: aggregation of the top-k neighbours by each of their types (including the exact and all super types)
- *p-o* `level`: aggregation of the top-k neighbours by each of their *p-o* nodes

We consider the following two aggregation functions:

- *Majority vote:* This is the standard method for the K-nearest neighbour classification for which the input values are classified based on a majority vote over the k nearest neighbours. Therefore, given an aggregation level, we rank the aggregated results (e.g. properties) based on the appearance in the top-k neighbours. Consider the right part of Fig. 3 in which we illustrate such a ranking process. For instance, the property aggregation would rank p_1 higher than p_2 since p_1 appears three times in comparison to p_2 which only appears 2 times.
- *Aggregated distance:* Our second aggregation function, we rank the aggregated results not by the number of their appearances, but compute the average distance. For instance, we would compute the distance for p_1 in Fig. 3 by averaging the distance of node P_3, P_2 and T_1.

In addition to the aggregation of properties, types and property-object pairs, we can also perform a nested level aggregation. For instance, we could aggregate first on the property level and then inside each property on the type level. An example for the nested aggregation based on the majority vote is depict in Fig. 3; the most likely type for p_1 would be t_1 with 2 votes, followed by t_2 and t_3.

6 Evaluation and Experiments

We have implemented a prototype system in Python to evaluate our approach with different functions. As a dataset to construct our background knowledge we use the DBpedia 3.9 dump.[2] The aim of our evaluation is twofold: We first automatically evaluate the accuracy of our prediction functions with different setups of the background knowledge in a controlled environment by splitting the DBpedia data into a test and training dataset. Secondly, we manually test our approach over Open Data CSV files to gain first insights for future directions, whether there is a chance to label tabular columns outside of DBpedia.

6.1 Background Knowledge Construction

We selected 50 of the the most frequently used numerical DBpedia properties to build our background knowledge for both evaluation scenarios:[3] we excluded properties which clearly indicate internal DBpedia ids only, such as `dbo:wikiPageRevisionID` as well as properties which are not directly in the root

[2] http://downloads.dbpedia.org/3.9/en/mappingbased_properties_en.nt.bz2, last accessed 2016-04-28.

[3] The full list of properties is online at http://data.wu.ac.at/iswc2016_numlabels/properties.html.

path of the http://dbpedia.org/ontology/ prefix. Figure 4 plots in the left figure the 5 % to 95 % inter-quantile ranges of our selected properties (in logarithmic scale) and in the right figure the total number of numeric values for each property. The range plot visualises the overlap of numerical values for our different properties and the quantiles are used to smoothen the ranges and eliminate possible outliers. About 60 % of the properties have values within the range 0–1000 and about 90 % within 0–2000.[4] The shortest range has the property `dbp:displacement` (inter-quantile range of 0.0058) and the maximum range of 2.56 billion has the property `dbp:areaTotal`. Regarding the total number of values, the longest bar, with 421k values, corresponds to the `dbp:years` property and the shortest to `dbp:width` (9.6k values).

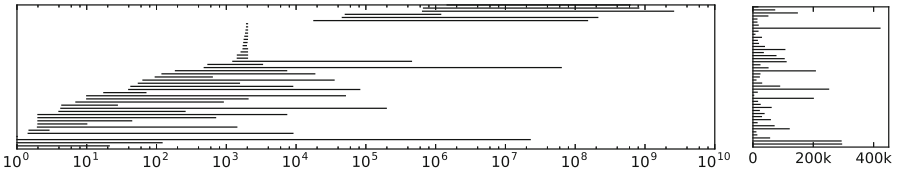

Fig. 4. 5 %–95 % inter-quantile ranges and number of values of training properties

We built three versions of our background knowledge graph to better understand the impact of the three different distance functions. One function is based on the Kolmogorov - Smirnov distribution test, and two based on the euclidean distance over feature vectors. The first type of vector uses the minimum and maximum of the values as features while the other uses the 5 % and 95 % quantile as features. We add to both vectors the mean and standard deviation as additional dimensions. Table 2 gives an overview of our three knowledge bases together with the number of nodes, the construction time of the background knowledge graph and the average prediction time for a given set of values (based on our evaluation runs).

In addition we added our average prediction times for the different setups. However, please note that we did not optimize our system wrt. runtimes.

Table 2. Setup of our three background knowledge graphs

ID	Distance measure	Nodes	Build time	Avg. pred. time
`KS`	Kolmogorov-Smirnov test	11431	30 m	2.5 s
`FV1`	(min, max, mean, std)	11432	24 m	2.3 s
`FV2`	(5-q, 95-q, mean, std)	11432	38 m	4.6 s

[4] Note, that around 30 % of the properties have values in the range of 1000–2000 and mainly describe years.

In future work we plan the improvement of these prediction times in order to provide our algorithm as a live service. All evaluation are conducted on a machine with 30 GB of RAM.

6.2 Model Evaluation

Our first experiment is designed to obtain the performance characteristics of our prediction for different distance functions.

Test and training data selection: To get an unbiased assessment we randomly assigned 20 % of the subjects for each property as test data and the remaining subjects are used to build the knowledge bases. The test data is further processed to find suitable test groups. To build those test-groups, we proceed in a similar manner as for the construction of the background knowledge base. That is, we analogously built type hierarchy and *p-o* hierarchy for per property, however, this time without imposing any constraints and creating all possible test contexts and sub-contexts. Eventually, we randomly select the leaf nodes of this "test context graph" and the respective numerical value bags as test data. This process ensures that we select context nodes which are not necessarily contained 1-to-1 in the background knowledge graph.

Evaluating Distance Functions. Our first evaluation aims to (i) test the impact of the distance function for the prediction and (ii) to select the best setup for further tests. We set up an initial experiment by randomly selecting a maximum of 50 leave nodes from each property tree in our test dataset; resulting in 1787 test nodes.

Table 3. Accuracy in % for different distance functions

	`FV1`			`FV2`			`KS`		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
`exact`	2.5	8.2	8.2	2.5	8.2	8.2	**12.3**	**41.8**	**47.9**
`prop`	45.4	60.3	60.3	45.4	60.3	60.3	**57.1**	**74.1**	**79.8**
`type`	11.3	24.9	24.9	11.3	24.9	24.9	**16.1**	**43.9**	**56.0**
`stype`	24.9	41.1	41.1	24.9	41.1	41.1	**35.8**	**58.6**	**67.5**

To initially measure the accuracy of the top-k neighbours, we introduce the following evaluation measures:

- `exact`: the top-k neighbours contain the *correct node* in the graph, that is, the test node and predicted node share the same property, type and *p-o* pairs.
- `prop`: the top-k neighbours contain the *correct property/label*
- `type`: the top-k neighbours contain the *correct type*
- `stype`: the top-k neighbours contain the *correct super type* of the test node

The results in Table 3 show the accuracy for different metrics for the top-k neighbours, with the best results marked bold. We can clearly see that the Kolmogorov-Smirnov based distance function (KS) outperforms for all metrics the feature vectors based functions in terms of prediction accuracy. The initial results show that our approach already predicts the correct property of the input values in the top-10 neighbours for 79 % of all test and the right type in 56 % of the cases. Based on the clear results, we decided to use the prediction approach based on Kolmogorov-Smirnov distance function in the remaining evaluation.

Large-Scale Model Validation: The next experiment focuses on the evaluation of the different aggregation functions and levels. We randomly sampled 33657 test nodes by selecting a maximum of 20 % of the leave nodes for each property in our test data set. The test data is ∼18 times larger than in the previous experiment and 3 times the size of our training nodes. In addition, only 9 % of the test context nodes are contained 1-to-1. This allows us to study our approach for input data for which we have only partial evidences available. We evaluate the accuracy for the different levels by measuring if the top-k aggregated results contain the correct property, type, parent types or any *p-o* context of the test instance.

Table 4 summarises the accuracy (in %) over 33k test instances for two aggregation functions over the top-k nearest neighbours. Overall, the results show a high prediction accuracy of over 92 % across all different levels for the top-10 aggregated results using the top-50 closest neighbours. For the root-type prediction, we observe the highest accuracy within the top-5 aggregated results. Regarding test nodes for which we have only partial information available, our approach can still predict the correct property, (parent) type and even some of the shared *p-o* pairs. Our results also show that doubling the number of neighbours significantly improves the prediction accuracy by up to 15 %. Considering the two aggregation functions, we see that ranking the results based on majority votes performs slightly better than using the average distance, with the

Table 4. Accuracy in % for different aggregation levels and functions (maj. = majority vote, avg. = average distance)

Top-k		prop		type		all-types		root-type		p-o level	
Neigh.	Agg. results	Maj.	Avg.	Maj.	Avg.	Maj.	Avg.	Maj.	Avg.	Maj.	Avg.
25	1	59.3	34.5	64.7	57.8	64.7	57.8	66.4	69.2	20.4	24.9
	5	87.9	82.9	91.4	85.3	91.4	85.3	94.7	94.7	75.8	66.2
	10	98.5	98.5	94.7	94.7	94.7	94.7	94.7	94.7	83.8	74.0
50	1	57.4	23.7	66.4	37.6	66.4	37.6	66.7	70.7	20.4	24.9
	5	98.4	83.7	93.2	65.4	93.2	65.4	96.3	96.3	75.8	66.2
	10	99.3	99.3	96.3	96.1	96.3	96.1	96.3	96.3	83.8	74.0

biggest impact for the *p-o* level aggregation. Interestingly, inspecting the top-1 aggregated results using the 50 nearest neighbours, we see that the `root-type` accuracy is lower than the `all types` accuracy. This is a false negative classification which can happen if there exists more than k results with equal votes or average distances. In such case, we rank the results in alphabetical order and return only the top-k, leading to a cutoff of possible correct results.

Looking at the top-10 of the aggregated results, we correctly predicated 99.5 % of the properties, 96.3 % of the exact and parent types and 92 % of the *p-o* pairs. These results are encouraging to use our approach for labelling numerical columns in tabular data, especially since we can also partially label values for which we do not have full evidences in our background knowledge graph.

6.3 Semantic Labelling of Numerical Columns in Open Data Tables

Eventually, we study how our approach performs for numerical columns in Open Data tables. We have to emphasise upfront, that this experiment is of rather exploratory than quantitative nature, since - due to the heterogeneity of data typically published in Open Data portals vs. DBpedia, we could not expect a lot of exact matches.

To conduct our experiment, we downloaded and parsed in total 1343 CSV files from two Open Data portals, namely the Austrian Open Government Data portal (AT[5]) and the European Open Data portal (EU[6]). We used the standard Python CSV parser to analyse the tables for missing header rows and performed a simple datatype detection to identify numerical columns. In order to get insights into the descriptiveness of these headers we tried to map header labels to BabelNet [10] in a non-restrictive manner: we performed a simple preprocessing on the headers (splitting on underscores and camel case) and retrieved all possible mappings from the BabelNet API.

Table 5 shows some basic statistics of the CSV tables in the two portals. An interesting observation is that the AT portal has an average number of 20 columns per table with an average of 8 numerical columns, while the EU portal has larger tables with an average of 4 out of 20 columns being numerical. Regarding the descriptiveness of possible column headers, we observed that 28 % of the tables have missing header rows. Eventually, we extracted headers from 7714 out of around 10K numerical columns and used the BabelNet service to retrieve possible mappings. We received only 1472 columns mappings to BabelNet concepts or instances, confirming our assumption that many headers in Open Data CSV files cannot easily be semantically mapped.

Exploratory Experiments: We used the numerical columns from our CSV corpus as input for our system and manually study select columns to gain first insights. Initially, we ranked the columns by their average distance over the 50

[5] http://data.gv.at/.
[6] http://open-data.europa.eu/.

Table 5. Header mapping of CSVs in Open Data portals

Portal	Tables	$\overline{cols}$	$\overline{num.cols}$	w/o Header	Num. H	Mapped
AT	968	13	8	154	6,482	1,323
EU	357	20	4	223	1,233	349

nearest neighbours and inspected the top-100 columns for each portal. We share the interesting results for tables and columns online.[7]

Our first observation observation is related to the time coverage of numerical values and the difference between Open Data and DBpedia. For instance, the Austrian Open Data portal hosts tables as specific as numbers of cars per brands per district in Vienna, or current (every 15 min) weather data from different weather stations in Austria. We do not expect matches for such specific numbers or even for temperature values if they are given in the form of timelines. In contrast, DBpedia typically has numeric values only for "current" or "latest" for many properties, taking population numbers of settlements as an example. Still, we are curious to see what the method would return and partially could explore interesting findings.

Another observation is that our knowledge base does not cover some of the domains and attributes of the numerical Open Data columns. For instance, many columns describe "counts" or "statistics". Examples for such count columns are the number of registered car model per district, the count of tourists grouped by their nationality, month of year and country/region they visit or the count of valid or invalid votes for an election. Examples for statistics are election results or the percentage of registered people for different age groups and districts in a city. For instance, take the 14th ranked Column#14[8] which describes election results divided by different regions, with a non-descriptive header `UNG.` (we assume this means "invalid votes"). The second-ranked property is *populationTotal* which is arguable a related labelling, since election results are basically sub-populations of different regions. Looking at the results of the type aggregation for this column, we find five times the type *Settlement* within the first ten neighbours, which further indicates that the values rise from (sub-)populations. Similarly, Column#1[9] holds counts of car models grouped by regions which our algorithm again labelled as population. This shows clearly that to label Open Data columns we need a very broad coverage of numerical domains in our background knowledge.

We also aggregate the results across columns to identify the "domain" of a table using the top-10 results of our `all-types level` aggregation and manually inspected some results. Again, we ranked the tables based on their average distances across all their numerical columns. For instance, consider the second

[7] http://data.wu.ac.at/iswc2016_numlabels/.
[8] http://data.wu.ac.at/iswc2016_numlabels/submission/col14.html.
[9] http://data.wu.ac.at/iswc2016_numlabels/submission/col1.html.

ranked Table#2[10] which consists of multiple columns which describe population counts for different districts. Aggregating and ranking the types across these columns results in the types *Town* and *PopulatedPlace* which proved to be right.

Discussion of Findings: While the findings did not yet provide, clear and convincing matches, we could collect valuable insights from this results on challenges to be tackled in future work:

- *Dealing with timeline data:* To correctly handle timeline data, we first need to be able to detect the time dependency and than regroup or transform the table.
- *Domain specific background knowledge:* Open Data contains many tabular data which is similar in itself, but not necessarily matching DBpedia categories and values reported there, e.g. reports for spendings/budget election results, tourism or population demographics. Our results highlight the limited coverage of DBpedia, which was also observed in the work from Ritze et al. [14]. Therefore, we have to gradually enrich the background knowledge graph from categories learned from Open Data tables themselves.
- *Aggregating column scores:* While single columns provided partially bad recognition, in some cases combined recognition of columns revealed interesting combinations.
- *Combine with existing complementary approaches:* Lastly, while we deliberately left it out of scope in this paper, linguistic cues could and probably should be used in combination with our numerical methods as an additional cue to gradually improve labelling/matching capabilities, as we explore and collect more Open Data sources.

7 Conclusions and Future Work

To the best of our knowledge, this is the first work addressing semantic labelling of numerical values by applying k-nearest neighbours search over a background knowledge graph, which is constructed in an unsupervised manner using hierarchical clustering. Our evaluation shows that we can assign fine-grained semantic labels when there is enough supporting evidence in our background knowledge graph. In other cases, our approach can nevertheless assign high level contexts to the data. Given a bag of numerical values, we correctly identified in 99.5 % of the test cases the properties, in 96.3 % the exact or parent type, and in 92 % the shared property-object pairs. Despite the simplicity of our solution, we can confirm that a knowledge base can be harnessed to perform automatic semantic labelling of datasets with promising results.

The obtained results are encouraging for labelling numerical columns in tabular data. A first feasibility evaluation using numerical columns in Open Data CSV files showed that further research is needed to extend our knowledge graph to cater for the specifics of the Open Data domain, such as addressing timeliness.

[10] http://data.wu.ac.at/iswc2016_numlabels/submission/tab2.html.

In future work, we plan to extend our background knowledge, using more properties from DBpedia and combining it with knowledge from other RDF datasets, such as WikiData or eurostats. To achieve better results in such combined datasets the handling of units of measurement and the time dimension is a vital extension of our system [23]. Complementary, we will focus on building the background knowledge in a bottom-up approach from information extracted out of CSV files as outlined in this work. We will also investigate performance optimization techniques, since our prediction time increases linearly with the number of context nodes. For example, we will explore range indices or pre-filtering to reduce the search space in the context graph. Another direction is to exploit our system in combination with other approaches for labelling tables based on textual header; e.g., [8] nicely complements our approach: Halevy et al. group together semantically related attributes and relate them to corresponding classes.

We believe that our approach can provide important clues about the context of numerical values which can be exploited in other domains, e.g., as input for ontology alignment between two different RDF datasets [7,15] or as input for computing the relatedness between tables such as used in [4].

Acknowledgments. This work has been supported by the Austrian Research Promotion Agency (FFG) under the project ADEQUATe (grant no. 849982).

References

1. Adelfio, M.D., Samet, H.: Schema extraction for tabular data on the web. Proc. VLDB Endow. **6**(6), 421–432 (2013)
2. Arenas, M., Bertails, A., Prud'hommeaux, E., Sequeda, J.: A direct mapping of relational data to RDF, W3C Recommendation, September 2012. http://www.w3.org/TR/rdb-direct-mapping/
3. Cruz, I.F., Ganesh, V.R., Mirrezaei, S.I.: Semantic extraction of geographic data from web tables for big data integration. In: Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR 2013, pp. 19–26. ACM, New York (2013)
4. Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., Yu, C.: Finding related tables. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 817–828. ACM (2012)
5. Ermilov, I., Auer, S., Stadler, C.: User-driven semantic mapping of tabular data. In: Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS 2013, pp. 105–112. ACM, New York (2013)
6. Fleischhacker, D., Paulheim, H., Bryl, V., Völker, J., Bizer, C.: Detecting errors in numerical linked data using cross-checked outlier detection. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 357–372. Springer, Heidelberg (2014)
7. Gal, A., Roitman, H., Sagi, T.: From diversity-based prediction to better ontology & schema matching. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, pp. 1145–1155 (2016)
8. Halevy, A.Y., Noy, N.F., Sarawagi, S., Whang, S.E., Yu, X.: Discovering structure in the universe of attribute names. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, pp. 939–949 (2016)

9. Lopez, V., Kotoulas, S., Sbodio, M.L., Stephenson, M., Gkoulalas-Divanis, A., Aonghusa, P.M.: QuerioCity: a linked data platform for urban information management. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part II. LNCS, vol. 7650, pp. 148–163. Springer, Heidelberg (2012)
10. Navigli, R., Ponzetto, S.P.: BabelNet: the automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artif. Intell. **193**, 217–250 (2012)
11. Ramnandan, S.K., Mittal, A., Knoblock, C.A., Szekely, P.: Assigning semantic labels to data sources. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 403–417. Springer, Heidelberg (2015)
12. Rastan, R.: Towards generic framework for tabular data extraction and management in documents. In: Proceedings of the Sixth Workshop on Ph.D. Students in Information and Knowledge Management, PIKM 2013, pp. 3–10. ACM, New York (2013)
13. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML tables to DBpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, WIMS 2015, Larnaca, Cyprus, pp. 10:1–10:6 (2015)
14. Ritze, D., Lehmberg, O., Oulabi, Y., Bizer, C.: Profiling the potential of web tables for augmenting cross-domain knowledge bases. In: Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, pp. 251–261 (2016)
15. Rong, S., Niu, X., Xiang, E.W., Wang, H., Yang, Q., Yu, Y.: A machine learning approach for instance matching based on similarity metrics. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 460–475. Springer, Heidelberg (2012)
16. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a web of semantic data for interpreting tables. In: Proceedings of the Second Web Science Conference, April 2010
17. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: A scalable approach to learn semantic models of structured sources. In: Proceedings of the 8th IEEE International Conference on Semantic Computing (ICSC 2014) (2014)
18. Tandy, J., Herman, I., Kellogg, G.: Generating RDF from tabular data on the web, W3C Recommendation, December 2015. https://www.w3.org/TR/csv2rdf/
19. Umbrich, J., Neumaier, S., Polleres, A.: Quality assessment & evolution of open data portals. In: IEEE International Conference on Open and Big Data, Rome, Italy, August 2015
20. Venetis, P., Halevy, A.Y., Madhavan, J., Pasca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. PVLDB **4**(9), 528–538 (2011)
21. Wang, J., Wang, H., Wang, Z., Zhu, K.Q.: Understanding tables on the web. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 141–155. Springer, Heidelberg (2012)
22. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 504–518. Springer, Heidelberg (2014)
23. Zhang, M., Chakrabarti, K.: 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, pp. 145–156. ACM, New York (2013)
24. Zhang, Z.: Towards efficient and effective semantic table interpretation. In: Mika, P., et al. (eds.) ISWC 2014, Part I. LNCS, vol. 8796, pp. 487–502. Springer, Heidelberg (2014)