Semantic Web 1 (0) 1–5
IOS Press

One-shot HDT-based Semantic Labeling of Entity Columns in Tabular Data

Ahmad Alobaid [a], Wouter Beek [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] *Triply Inc. & VU University Amsterdam,*
E-mail: wouter@triply.cc

Abstract.
A lot of data are shared across organisations and on the Web in the form of tables (e.g., CSV). One way to facilitate the exploitation of such data and allow understanding their content is by applying semantic labeling techniques, which assign ontology classes to their tables (or parts of them), and properties to their columns. As a result of the semantic labeling process, such data can be then exposed as virtual or materialised RDF (e.g., by using mappings), and hence queried with SPARQL. We propose a one-shot semantic labeling approach to learn the classes to which the resources represented in a tabular data source belong, as well as properties of entity columns. In comparison to some of our previous approaches, this approach exploits the fact that the knowledge base used as an input source is only available in the RDF HDT binary format. We evaluate our approach with the T2Dv2 dataset. The results show that our approach achieves competitive results in comparison with state-of-the-art approaches without the need for using a full-fledged query language (e.g., SPARQL) or profiling of knowledge bases.

Keywords: Semantic Labeling, Semantic Annotation, HDT

1. Introduction

Private and public organisations worldwide often share part of their data on the Web or in private data spaces using tabular formats such as CSV, TSV, and XLS (Microsoft Excel format).

In the case of data sharing on the Web, the use of such formats does not follow the data publishing guidelines recommended by the W3C[1]. This makes it difficult to exploit and reuse data. It may require an expert in the domain to understand the content of the data and a knowledge engineer to develop an adapter or something alike for existing systems to handle and process such data. One of the reasons for this situation is the lack of tools to be used by data owners or publishers. This calls for semantic labeling approaches and tools that can facilitate following best practices in data sharing.

Semantic labeling assigns classes and properties from ontologies[2] to tables and columns in tabular data. It provides a way to understand the content of the data in a unified manner, so it would be much easier to exploit and reuse.

There are several approaches proposed in the literature to address this problem. However, we observed several drawbacks in them: the reliance on external sources of knowledge, such as search engines [2, 3], the need for manual intervention [4–7], and the dependence on a fixed data source [4, 8], among others.

In previous works, we have proposed applying semantic labeling techniques over some specific types of columns that appear in tabular data, such as numerical columns [9, 10] and subject columns [11]. In these approaches, we relied on SPARQL endpoints as the sources of knowledge (training set). We noticed that

[1] https://www.w3.org/TR/ld-bp/

[2] An ontology is "an explicit specification of a conceptualization" [1]

the bottle neck in these systems was in querying the SPARQL endpoints. A similar observation was also reported by Neumaier et al. [12][3].

On the other hand, HDT provides a fast way to query knowledge bases. HDT is a compressed binary format to store RDF knowledge bases [13]. It is well known for its compact representation, what allows storing large knowledge bases in small binary files. However, it is limited in the kind of supported queries, which do not cover the full range of SPARQL expressivity.

In this work, we propose a semantic labeling approach to utilize the fast query processing of HDT format, while overcoming the limited expressively of the HDT format in comparison to SPARQL. We address two kinds of columns: subject columns and property columns. Subject columns contain (the names of) the main entities of the tables. For property columns, we focus on columns which contain entities which are not the subjects. For example, a table of football players has the column with the names of the players as the subject column and the column of their country of birth as the non-subject entity column.

We summarize our contributions in this paper as follows:

1. A subject column semantic labeling algorithm to assign classes to subject columns in tabular data using HDT-compressed knowledge bases as the only source of knowledge.
2. A property column semantic labeling approach to assign properties to non-subject entity columns using also HDT-compressed knowledge bases as the sole source of knowledge.

Our approach considers the following assumptions:

1. The majority of the entities in tabular data exist in the HDT file.
2. The natural language used inside the tabular data and the HDT knowledge base is the same.
3. Each table contains a single subject column. We do not consider subjects split in multiple columns (e.g., first name and last name).

The rest of the paper is organized as follows. In Section 2, we describe briefly the RDF HDT binary format. In Section 3, we present a semantic labeling example. We describe our semantic labeling approach of subject columns in Section 4. Next, we explain our semantic labeling approach on property columns that contain non-subject entities in Section 5. In Section 6, we evaluate our semantic labeling approach on the subject and property columns. We discuss related work in Section 7, and we conclude the paper in Section 8.

[3]They reported that they were facing timeouts for running their SPARQL queries to look for numeric properties. We reported the same problem for getting numeric properties previously [9].

2. HDT

HDT is an acronym of Header, Dictionary, and Triples [13]. It is a compact binary format to store RDF data [13]. It is designed to address the need to exchange large RDF datasets. The Header stores the meta-data of the dataset. The Dictionary stores the mapping between terms and their unique integer IDs. The Triples store the graph using the integer IDs.

Searching in HDT supports a subset of SPARQL triple patterns. It is done using the triple patterns (S, P, O)[4]. S stands for Subject, P for Property, and O for Object. It also supports wildcard for any of the patterns (e.g., (?, P, O) all subjects with the given property and object pair). There are eight possible patterns in total: (S,P,O), (S,P,?), (S,?,O), (?,P,O), (S,?,?), (?,P,?), (?,?,O), (?,?,?).

Despite this limitation, it offers two important advantages: fast query processing and low memory requirement. HDT compact format makes it suitable to share RDF knowledge bases. So, offering semantic labeling on a compact format which is used for sharing knowledge bases make it more convenient than approaches that have high memory requirement (e.g., Ermilov et al. [14] reported a 100 GB of RAM as a requirement for T2K to run on their machines). HDT also provides a quick lookup for triples retrieval. Speeding semantic labeling makes it suitable for more use cases which need a quick responsive system, such as mapping editors (to generate RML mappings [15] to CSV files).

3. Example

In this section, we present an example of a table labeled with a class and two properties from the DBpedia ontology. In Figure 1, we present an example of a table with three columns. The first column represents the names of the subjects of the table (the main entities). The subject-column semantic labeling would assign the class *dbo:Scientist* to the sub-

[4]subject, property, object

Triples in the HDT file

dbr:Richard_Feynman	rdfs:label	"Richard Feynman"
dbr:Bertrand_Russell	rdfs:label	"Bertrand Russell"
dbr:United_States	rdfs:label	"United States"
dbr:United_Kingdom	rdfs:label	"United Kingdom"
dbr:Cambridge	rdfs:label	"Cambridge"
dbr:MIT	rdfs:label	"MIT"
dbr:Richard_Feynman	dbp:bornIn	dbr:United_States
dbr:Bertrand_Russell	dbp:bornIn	dbr:United_Kingdom
dbr:Richard_Feynman	dbp:studiedIn	dbr:United_States
dbr:Bertrand_Russell	dbp:studiedIn	dbr:United_Kingdom

Table about scientists and the universities they attended

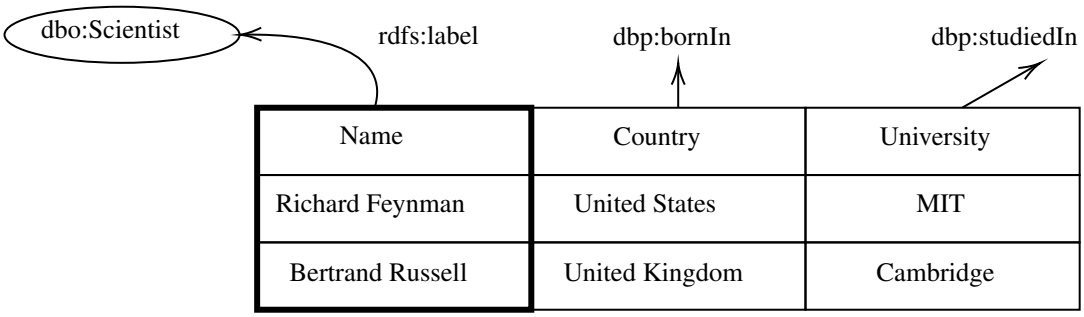

Fig. 1. An example of table labeled with a class and properties from an HDT-compressed knowledge base

ject column. In this example, we assume that both "Richard Feynman" and "Bertrand Russell" have the class *dbo:Scientist* as the *rdf:type*. The technique for assigning classes to subject columns is explained in Section 4. The second column also contain entities name. Since the second column is not the subject column, it will be matched to a property. We assume that there is a relation *dbp:bornIn* between the subjects in the column "name" and the entities in the column "Country" (e.g., the triple <*dbr:Richard_Feynman*, *dbp:bornIn*, *dbr:Unite_States*>. The same is applied to the third column and it will be matched to the property *dbp:studiedIn*. The technique for assigning the properties to the columns is explained in Section 5. We also show the triples in the HDT-compressed knowledge base.

4. Subject-column semantic labeling

In this section, we present our one-shot semantic labeling approach for labeling subject columns in tabular data, by using HDT sources. In tabular data, subject columns are the ones that contain the names of the main entities of the table [3, 10, 16]. This would be the column "Name" in our example in Figure 1.

Our approach consists in the following steps: Entity linking, Cell labeling, Hierarchy building, and Class scoring. Entity linking proposes candidate entities for each cell in the subject column using the cell content and entity properties in the same row. Then, each cell is assigned to one or more ontology classes. The class hierarchy of all the candidate classes is built. Finally, each of these classes is scored, and the highest scored class is selected as the class that represents the cells in that subject column. We explain each of these steps in more detail next.

4.1. Entity linking

This process picks the best candidate entities for each cell. The algorithm tries to find all entities which have the cell text value as the label. It uses the *rdfs:label* property to search for the name and gets the entity URIs. We use the triple pattern in Listing 1.

Listing 1: Get candidate entities of a cell

```
? rdfs:label "cell value"
```

The evaluation of this pattern returns all entities in the HDT source where the cell value is the object of the *rdfs:label* property.

The algorithm then disambiguates the entities by using the inner table context. For each cell which has candidate entities, the rest of the attributes in the same row are used. Two cases are considered for utilizing the in-table context. First, there is a textual property which matches an attribute in the same row. This is done in a single step using the following pattern (Listing 2).

Listing 2: Get candidate properties of an entity which has a given attribute value

```
entity-uri ? "attribute value"
```

Second, there is a relation between the entity in the subject column and another entity in another column. This is a two-step task. It looks for entities with the cell value of an entity column (Listing 1). Then, it looks for the relation between the two entities, subject column as *subject-uri*, and the other entity column as *entity-uri* (Listing 3).

Listing 3: Get the property between a subject and an entity

```
subject-uri ? entity-uri
```

These disambiguation techniques are performed for each candidate entity for each cell in the subject columns. Cells with candidate entities that are not disambiguated (with no equivalent property in the HDT source) are penalized, as explained in more detail in the class scoring section (Section 4.4).

4.2. Cell labeling

This step assigns candidate classes to each cell. For each of the candidate entities of a cell, the classes of these entities are obtained from the HDT using the pattern in Listing 4.

Listing 4: Get classes (types) of a given entity

```
entity-uri rdf:type ?
```

The algorithm then removes candidate classes that are ancestors of other candidate classes of the same cell. This is to remove extra weight on scores of ancestor classes, which might affect the optimal class selection of the subject column (Section 4.4). This is done by eliminating classes that other classes have *rdfs:subClassOf* relationship with. This is done using the following pattern (Listing 5).

Listing 5: Get parents of a class URI

```
class-uri rdf:subClassOf ?
```

The algorithm creates a hash of classes; each one is linked to a list of ancestors for that corresponding class. This is to speed up the lookup for ancestors instead of running the patterns each time to get the ancestors for each class.

For each candidate entity of a cell, the algorithm checks if one of them is an ancestor of the others. Ancestors will be eliminated, and only the remaining ones

(which are not necessarily leaves) are kept. For example, in DBpedia, the class *dbo:AmateurBoxer* is a *rdfs:subClassOf dbo:Boxer*. An entity might have the class *dbo:Boxer* but not *dbo:AmateurBoxer*.

4.3. Hierarchy building

The algorithm builds the class hierarchy using the *rdf:subClassOf* starting from the classes obtained in the previous step.

Each class in the hierarchy is scored. This is needed, as the overall class of a subject column of entities of the class *dbo:FootballPlayer* and *dbo:BasketballPlayer* is the class *dbo:Athlete* (which might not be a direct class obtained in Listing 4).

4.4. Class scoring

To score classes, the algorithm balances two contradicting objectives. The first one prefers classes that cover the majority of the cells in the column. All typed cells would belong to the root class in the class hierarchy (e.g., *owl:Thing* in DBpedia). The second objective favors more specific (narrow) types over more generic ones. For example, knowing that a column represents boxers (*dbo:Boxer*) is more valuable than knowing that the column represents entities of the class *dbo:Person*. We refer to these objectives as coverage and specificity, respectively[5]. The two objectives are balanced with a variable α, which ranges from 0 and 1 as follows:

$$\arg\max_t f(t) = \alpha * f_c(t) + (1-\alpha) * f_s(t) \quad (1)$$

For a class (type) t, $f_c(t)$ represents its coverage score, and $f_s(t)$ represents its specificity score. We present the notations we use in Table 1.

4.4.1. Coverage

The coverage score for a class deals with the cells in a column of that class. It balances the score for each cell taking into account the number of candidate entities in a cell. The premise is that if multiple entities are linked to a cell, it means that we were not able to disambiguate which one is the correct one, and hence each of these candidate entities is assigned less score that if there is only one candidate entity.

[5]We presented several scoring functions and compared the performance of them in our previous work [11]. However, we present them here to make the paper self-contained.

Table 1
Notation for scoring functions in Chapter **??**

Notation	Meaning
$\Psi(t)$	cells that have the type t in a given column
$L_c(t)$	the local coverage score for a type t
$I_c(t)$	the I_c coverage of a type t
$U(t)$	the children of type t
$Z(v)$	entities of a given text value v
$\|\|Z(v)\|\|$	the number of entities of a given text value v
$Q(e)$	the types of an entity e
$\|\|Q(e)\|\|$	the number of types of an entity e
$R(t)$	the entities of a type t from the given knowledge graph
$\|\|R(t)\|\|$	the number of entities of a type t
m	the number of annotated cells in a given column
$f_c(t)$	coverage score of a semantic type t
$f_s(t)$	specificity score of a semantic type t
α	balances specificity score $f_s(t)$ and coverage score $f_c(t)$
$pr(t)$	the parent of a type t

Another aspect that we did not use in the previous work is the use of in-table context. The algorithm tries to find any match of the subject name and an attribute of that subject in the input file to a triple in the HDT. If such a match is found, we consider these candidate entities of high quality. Otherwise, the algorithm tries to get candidate entities using only the names of the subjects. These candidate entities are considered of low quality. The algorithm applies a penalty to the low-quality candidate entities, which we refer to as the disambiguation penalty (dp).

We present three coverage related functions: I_c, L_c, and f_c. The function I_c computes the coverage of a single class/type t. It takes the number of entities (assigned to the cells) and the number of types of each entity into account. The higher the number of candidate entities or the number of types assigned to an entity, the lower the score of I_c is for the type t. We present the formula of $I_c(t)$ in equation 2.

$$I_c(t) = \sum_{v}^{\Psi(t)} \sum_{e}^{Z(v)} \frac{1}{||Z(v)|| \times ||Q(e)|| \times pd} \; ; t \in Q(e) \quad (2)$$

However, this does not take into account the class hierarchy. This means that a class (e.g., *dbo:Person*) is

not guaranteed to have higher coverage than a lower class (e.g., *dbo:Boxer*). We present Eq. 3 to address this, so each class has a higher coverage than its decedents.

$$L_c(t) = I_c(t) + \sum_{u}^{U(t)} L_c(u). \tag{3}$$

The L_c coverage score satisfies our intuition, but it tends to increase as the number of annotated cells increase. Hence, we normalize it by dividing the L_c by the total number of annotated cells m. We present the normalized coverage function f_c in Eq. 4.

$$f_c(t) = \frac{L_c(t)}{m}. \tag{4}$$

4.4.2. Specificity

We measure the specificity of a class/type t (how narrow a class is) by measuring the number of instances of a type t and the number of instances of its parent $pr(t)$.

$$I_s(t) = \frac{\|R(t)\|}{\|R(pr(t))\|}. \tag{5}$$

However, the I_s score only takes into account the direct parent without taking into account the ancestors. To take the whole path (to the root) into account, the I_s score is multiplied with all the I_s in the path (Eq. 6).

$$L_s(t) = I_s(t) \times L_s(pr(t)). \tag{6}$$

The L_s value indicates a low specificity, and low L_s indicates a higher specificity. We reverse it and bound it between 0 and 1 in Eq. 7.

$$f_s(t) = -L_s(t) + 1 \tag{7}$$

4.5. Label the subject column

After typing all the cells with coverage score f_c and specificity score f_s, the class with the highest f score is picked as the label to the subject column. However, sometimes not only the highest f is needed, but the top k types. Therefore, the application output the top candidate types ordered in descending order by the f score. This would make it also applicable to editors to label tabular data to suggest the most probable types.

5. Property-column semantic labeling

Attributes in tabular data can represent subjects, entities, numeric, and textual properties. In the previous section, we presented a semantic labeling approach that focuses on assigning classes to subject columns. In this section, we focus on assigning properties to attributes that represent entities.

Querying HDT is based on pattern matching, and complex queries (e.g., SPARQL) are not supported [13]. To compensate that, multiple simple patterns are applied sequentially. We present three different ways to predict entity properties: *restrictive*, *permissive*, and *heuristic*.

Restrictive technique relies on the existing patterns (relations) between the entities in the subject column and the entities in the property column. These relations are searched for in the HDT file. First, the candidate entities of a cell in the subject column are assigned. These are done using the same pattern used in the previous section using *rdfs:label* (Listing 1). Second, candidate entities of the equivalent cell in the property column (on the same row) is sought in the HDT file (Listing 1). Third, the relation (property) of a subject and an entity are gathered as candidate properties (Listing 3). We show the algorithm of the *restrictive* technique in Algorithm 1.

Permissive technique is more extensive. In the *restrictive* technique, it predicts the property based on the relations between the entities in the subject column and the property column. If no relation is found for the entities in the two columns, no properties will be predicted. However, entities of the same class (type) may share the same properties and relate to the same entities. For example, if a football player (*dbo:FootballPlayer*) is born in (*dbp:bornIn*) Spain (*dbo:Spain*), there might be another football player who is also born in Spain. To apply this intuition, the algorithm first gets all entities of the class (type) of the subject column in a table. Then, it gets all entities of the property column. Last, it collects all properties between any entities of the subject column class (type) and the entities in the property column. These properties are considered candidate entities between the subject column and the property column (Algorithm 2).

Algorithm 1: Restrictive semantic labeling of non-subject entity columns

```
1  Function restrictivePrediction(subject_col, property_col):
2      hash = ∅
3      for i=0; i < subject_col.size() ; i++ do
4          subjects = getCandidateSubjects(subject_col[i])
5          entities = getCandidateEntities(property_col[i])
6          for j=0; j < subjects.size(); j++ do
7              for k=0; k < entities.size(); k++ do
8                  properties = getCandidateProperties(subjects[j],entities[k])
9                  for l=0; l < properties.size(); l++ do
10                     if properties[l] ∉ hash then
11                         hash[properties[l]] = 1
12                     else
13                         hash[properties[l]] += 1
14                     end
15                 end
16             end
17         end
18     end
19     properties = sortByValue(hash)
20 return properties
```

Algorithm 2: Permissive semantic labeling of non-subject entity columns

```
1  Function permissivePrediction(subject_col, property_col):
2      hash = ∅
3      class = getClass(subject_col)
4      subjects = getEntitiesOfClass(class)
5      for i=0; i < property_col.size() ; i++ do
6          entities = getCandidateEntities(property_col[i])
7          for j=0; j < subjects.size(); j++ do
8              for k=0; k < entities.size(); k++ do
9                  properties = getCandidateProperties(subjects[j],entities[k])
10                 for l=0; l < properties.size(); l++ do
11                     if properties[l] ∉ hash then
12                         hash[properties[l]] = 1
13                     else
14                         hash[properties[l]] += 1
15                     end
16                 end
17             end
18         end
19     end
20     properties = sortByValue(hash)
21 return properties
```

Algorithm 3: Heuristic semantic labeling of non-subject entity columns

```
Function heuristicPrediction(subject_col, property_col):
    properties = restrictivePrediction(subject_col, property_col)
    if properties.size() == 0 then
        properties = permissivePrediction(subject_col, property_col)
    end
return properties
```

Heuristic technique combines the *restrictive* and *permissive* techniques. The intuition is that the predicted properties between entities in the subject column and the property column (*restrictive* technique) are most likely to be true in comparison to predicted properties between the entities of the class (type) of the subject column (which might not be in it) and the entities property column. If the *restrictive* technique resulted in no predicted properties, the *permissive* technique is used instead. The algorithm of the *heuristic* technique is shown in Algorithm 3.

6. Evaluation

In this section, we evaluate the performance of our semantic labeling approach to label subject columns and property columns. We measure the performance using precision, recall, and F1 score.

6.1. Experiment

We performed our experiments on the T2Dv2 dataset. T2Dv2 dataset is now becoming the norm to evaluate semantic labeling approaches on tabular data. It is composed of 237 tables extracted from the Web and manually annotated [17]. These tables are annotated with classes and properties from DBpedia. We used the HDT of DBpedia[6] (version 2016-04). The experiment addresses two tasks: semantic labeling of subject columns and semantic labeling of property columns. For the semantic labeling of subject columns, we test two matching methods, one uses exact-match to perform entity linking, and the second transforms the value in the cell to title case (capitalize the first letter for each word) in case the exact match didn't link to any entity.

For the semantic labeling of property columns, we focus on property columns that contain entities. The T2Dv2 gold standard includes the equivalent DBpedia properties for the tables.

We gathered all annotated properties in the gold standard of the 234 (there are 3 files that have missing classes). The application generates all properties (except for the subject columns which are annotated with *rdfs:label*). We manually inspect all of these properties and filter out properties that do not represent entities (so it only includes properties that represent relations between the subject columns and other columns). Next, the application take these filtered properties and discard properties which are not found in the HDT for the given class. To avoid bias, this process does not check if these relations (between the subjects and the entities in other columns) are of the ones in the tables. The final list of properties is then used for the experiment. We experimented with three different methods: *restrictive*, *permissive*, and *heuristic*. The *restrictive* matching matches the subject and the entity (in the other column) by querying the HDT file. The *permissive* matching try to match all subjects[7] of the class of the subject column in the HDT file to the entities in the entity column. The *heuristic* technique applies the *restrictive* matching, and if it fails, it applies the *permissive* matching.

To perform the experiment, we developed an open-source library[8] (v1.5) to perform semantic annotation on a given column [18]. We developed another application[9] (v1.1) to perform the experiment on the T2Dv2 dataset using that library [19]. Both the library and the experiment were developed using C++ and the HDT library [20][10].

[6] http://www.rdfhdt.org/datasets/

[7] We use the term *subjects* to refer to entities in the HDT file which share the same class as the subject column of a table

[8] https://github.com/oeg-upm/tada-hdt-entity

[9] https://github.com/oeg-upm/tada-hdt-entity-experiment

[10] https://github.com/rdfhdt/hdt-cpp

We measure the performance of the approach using precision, recall, and F1 score. We use the following equations:

$$Precision = \frac{\#correct}{\#correct + \#incorrect} \tag{8}$$

$$Recall = \frac{\#correct}{\#correct + \#notfound} \tag{9}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{10}$$

#correct refers to the number of correctly labeled columns, *#incorrect* refers to the number of incorrectly labeled columns, and *#notfound* for the number of columns that the algorithm was not able to label at all (unable to find candidate entities or classes).

6.1.1. Setup

We used a MacBook Pro laptop, 2.8 GHz Dual-Core Intel Core i7, 8 GB 1600 MHz DDR3, with the macOS Catalina as the operating system. For the disambiguity penalty, we use dp=2, so disambiguated entities have double the score of non-disambiguated entities.

6.2. Results and Discussion

The experiment for applying semantic labeling to subject columns with exact-match took around 118 seconds. The experiment took a similar number of seconds for the title case match. We did not perform any kind of cache to speed the process. We did not take into account the time the HDT library took to generate the index file. The index file is automatically generated to speed up the querying and estimating the number of triples of the results.

The T2Dv2 dataset includes 237 tables. We found that three of them share no entities with the HDT file. So, we ignore these three tables, and we perform the experiment on 234 tables. We show the semantic labeling results of subject columns in Table 2.

The results show that our approach annotated the majority of subject columns successfully. We used exact-match to get potential entities for all cells in the subject column. We also experiment with forcing title case (capitalizing the first letter of each word) for each cell in case the entity linking of the original value does not result in any match. The use of title-case slightly improved the performance (+1 in precision and +2 in recall). This is because some of the tables have the cells all uppercased or lowercased, which causes the entity linking to fail, as the HDT pattern matching is case sensitive. We also compare the performance of our approach with T2K Match [17]. Our approach out perform T2K approach in recall and F1 score[11].

Table 2
The performance of subject-column semantic labeling approaches

Approach	k	Precision	Recall	F1
Our approach (exact match)	1	0.87	0.97	0.92
	3	0.95	0.97	0.96
	5	0.96	0.97	0.97
Our approach (title case)	1	0.88	0.99	0.93
	3	0.96	0.99	0.97
	5	0.97	0.99	0.98
T2K Match	-	0.93	0.91	0.92

We investigate some of the subject columns that our approach was not able to annotate. We found that some cells include the acronyms in the same cell after the label (e.g., "Democratic Unionist Party (DUP)"). Others are matched to the correct entities, but the entities in the HDT have no classes or are missing the classes.

For the semantic labeling of properties experiment, the run time differs for the different techniques. The *restrictive* technique took around 5 seconds, while the experiment with *permissive* technique took 9974 seconds (2 hours and 46 minutes), and 2384 seconds (40 minutes) for the experiment with the *heuristic* technique. This big gap is due to the way these techniques look for relations. For the *restrictive* technique, only possible relations between subjects and entities in the table are queried, while for the *permissive* technique, all subjects of the subject column class (even the ones which are not in the table) are queried in the HDT file for the relations (properties) to the entities in the other column.

We report the performance of the semantic labeling of the properties (relations to the non-subject entity columns) in Table 3. We see that the *permissive* technique achieves lower precision and higher recall and F1 score than the *restrictive* technique. This is because

[11] Note that we included the T2K Match scores reported in [17]. It gives an idea of how the performance of our approach is competitive to the performance of the T2K Match approach.

the subjects and entities in some tables have no equivalent relations in the HDT file, while other subjects of the same class have these relations in the HDT file. The*heuristics* technique takes the best of the other two techniques; it takes the high precision of the *restrictive* technique and the high *recall* of the *permissive* technique, which resulted in the highest F1 score.

We inspected several of the labelings which are not correctly matched to the correct properties. For the *restrictive* technique, we found two misclassified columns. One has the *dbo:wikiPageDisambiguates* more frequent than *dbo:usingCountry*, which is for the class *dbo:Currency*. The second one was of the class *dbo:City*. The majority of the subjects were matched to the property *dbo:leaderName* instead of the property *dbo:mayor*. For the *permissive* technique, we found that properties have similar entities (e.g., for the class *VideoGame*, the property *dbo:manufacturer* was misclassified as *dbo:publisher*, *dbo:developer*, and *dbo:distributor* which are often the same companies). There are other cases where the HDT file contains synonym properties (e.g., *dbo:country* and *dbo:locationCountry*). For the ones which are not found, despite the fact the properties do exist in the HDT file, they might not relate. So there are no properties between the subjects in the HDT and the entities in the table, but there are properties that relate to the entities which are not in the table. The *heuristic* prefers the *restrictive* if possible, and uses *permissive* otherwise, so the misclassifications are the same as of the *restrictive* and *permissive* techniques.

For the T2K Match, there are several techniques used to match table attributes to properties. In Table 3, we only reported the one with the highest F1 score reported in that paper [17][12]. The dictionary matcher uses the synonym dictionary built using the T2K match on the T2D dataset [17]. The duplicate matcher uses the similarity of the values (attributes and properties) for the same entities [7, 17]. We further describe the features used in T2K Match in Section 7.

The experiments show that our proposed semantic labeling approach for subject columns and non-subject entity columns were able to label the majority of columns correctly with competitive performance in comparison to the state-of-the-art approaches. It is important to mention the lack of approaches that report their used dataset hinders the comparison to the different approaches. The T2Dv2 gold standard published the data and the annotations, which allowed the comparison of different approaches on the same dataset.

[12]Note that in the experiment reported by Ritze et al. [17], the property matching experiment matches other kinds of properties as well, while in our experiment, we report the property matching to properties that represent attributes

Table 3
Semantic labeling of non-subject entity columns

Approach	k	Precision	Recall	F1
Our approach (Restrictive)	1	0.82	0.58	0.68
	3	1.0	0.62	0.77
	5	1.0	0.62	0.77
Our approach (Permissive)	1	0.76	0.7	0.72
	3	0.98	0.75	0.85
	5	0.99	0.75	0.85
Our approach (Heuristic)	1	0.78	0.72	0.75
	3	0.98	0.77	0.86
	5	0.99	0.77	0.86
T2K Match (Dictionary+Duplicate)	-	0.76	0.86	0.81

7. Related work

There are several presented approaches in the literature to understand the content of tabular data. They use different sources to interpret the content embedded in tabular data. They also differ in the techniques used to assign classes and properties from ontologies to columns and row in tabular data.

7.1. Learning sources

Knowledge graphs are rich sources of knowledge. Several approaches in the literature use knowledge graphs as the source of knowledge for semantic labeling [2–4, 6, 7, 17, 21–25]. They use DBpedia [2, 4, 6, 7, 17, 21, 23, 25], YAGO [4, 21–23], and Freebase [3, 22, 24].

However, there are others which use Web pages as the training set [5, 16, 25–27]. Cafarella et al. [26] use the Google crawl while Zhang et al. [5] use a subset of Web pages in the domain of Company, Country, and City from Microsoft Bing. Others used their own crawled Web pages [16, 27].

7.2. Labeling techniques

There are different approaches reported in the literature. Some of the use machine learning techniques,

such as Linear Regression [26], Logistic Regression [25], SVM [2, 8], Probabilistic Methods [5, 14, 16, 21, 23, 27], Decision Trees [8], and Hierarchical Clustering [12]. Such techniques are applied on different set of features, such as similarity measures. There are other techniques which are not based on machine learning techniques [3, 6, 7, 17, 22, 24, 28]. They rely on similarity measures [3, 6, 22, 24, 28], majority vote [17], and iteration (where entity linking and semantic labeling of columns alternate to disambiguate the candidate classes and entities) [3, 7, 17, 24].

7.3. Experiments on T2D

The approach proposed by Ritze et al. [17] were used to evaluate the different combination of features. Their proposed approach is called *T2K Match*. It iterates between instance matching and schema matching.

For instance matching, it uses similarity measures to compare the cell value with the entity label. It also uses the similarity measure with a catalogue of alternative names of entities, using Wikipedia articles and disambiguation pages. It takes into account the popularity of entities to disambiguate entity linking (favoring popular entities). It also exploits abstracts of entities. It compares cell values with entity abstracts using similarity measure. For schema matching, it assigns classes and properties to the columns of the table. For assigning a class to the (subject column of the) table, the following features are used: page URL, page title, text surrounding the table, classes of candidate entities, frequency of the classes (favoring specific ones over more general classes), and the agreement of the candidate classes. For assigning attributes in the table to properties, it uses the similarity between the attribute label (header text) and the property label in the knowledge base. It also uses the similarity between the hyponyms of the attribute label from WordNet, and synonymous gathered from attribute labels matched to property labels. They performed their experiment on the T2Dv2 dataset. It shows that using all the features together out perform using individual and other combinations of the features for assigning a class of the subject column. However, for assigning properties to attributes in dataset, using only the attribute values and the attribute synonyms catalogue they built using their T2K Match with DBpedia out perform using all the features.

In our previous work, we also proposed a semantic labeling approach on the T2Dv2 dataset previously [11]. The approach focuses on assigning classes to subject columns in tabular data. It balances two contradicting preferences: classes which covers as much entities in the subject column as possible (which is often the most general class e.g., *owl:Thing*) and more specific classes (e.g., to favor *dbo:Scientist* over *dbo:Person*). In order to highlight the most relevant differences our work in this paper in comparison to the previous work [11], we present the key differences below:

1. The use of HDT instead of SPARQL. The use of HDT instead of SPARQL is not just a matter of changing the syntax of the query language; the logic is also changed. HDT only supports simple triple patterns in comparison to SPARQL, which supports more complex queries. However, HDT provides a compressed format to store knowledge bases. It also offers a fast way to query HDT-compressed knowledge bases.
2. The use of the inner context. In previous work, only the cells in the subject columns are used, while in this work we use other columns to improve entity disambiguation. We also penalize entities that the algorithm was not able to disambiguate.
3. The use of the label property. In this work, we use *rdfs:label*, while in the previous one, we use all properties, which could match a given text (cell value) to incorrect entities. It also results in a smaller number of queries.
4. Our approach does not filter non-DBpedia classes. In previous work, non-DBpedia classes were eliminated as the T2D gold standard is annotated only with DBpedia classes. In this work, we did not eliminate any classes, and candidate classes are picked solely based on the scoring functions. This shows that our HDT-based approach is superior to the previous one.
5. Semantic labeling of non-subject entity columns with equivalent properties. The previous work focuses on assigning classes from the knowledge graph to the subject columns only. In this work, we also assign properties to non-subject entity columns.

8. Conclusion

In this paper, we proposed a one-shot semantic labeling approach to learn from HDT-compress knowledge bases instead of knowledge bases available as SPARQL endpoints. Our approach annotates subject

columns with semantic classes from HDT files and non-subject entity columns with semantic properties from the HDT. Despite the limitation of querying HDT files, our proposed approach produced competitive results in comparison to state-of-the-art approaches. Our approach was able to label subject columns of 234 tables in 2 minutes. Our experiment shows that using title-case has a slight improvement of the labeling score in comparison to the exact match in entity linking. For semantic labeling of properties of non-subject entity columns, our approach took around 5 seconds to complete the experiment on the whole T2Dv2 dataset (using the *restrictive* technique). For the use cases where semantic labeling accuracy is more valuable than the speed of assigning the properties, the *heuristic* technique is more appropriate. The *restrictive* technique achieves higher precision than the *permissive* technique, while the *permissive* technique achieves higher recall. The *heuristic* technique took the best of the two techniques and achieved high precision (as it prefers to use *restrictive* technique if it matches to at least a single property from the HDT file) and recall (as it uses the *permissive* technique when the *restrictive* technique do not predict any properties). We showed that HDT can be an alternative to SPARQL endpoints for semantic labeling. Using our semantic labeling approach, we show how we can take advantage of the fast lookup in HDT while achieving high-quality semantic labeling. This would also make it suitable to use cases that require fast semantic labeling like semi-automatic semantic labeling editors of tabular data.

As for future work, we are working on addressing tables with multi-column subjects, where subjects are split into multiple columns. For example, tables with first name and last name as the subject instead of a single column with the first and the last name concatenated. We are also working predicting properties of non-entity columns, such as textual and numeric columns. We are also planning to provide a way to benefit from additional meta-data (e.g., abstract, URL, caption, surrounding text) to improve the semantic labeling accuracy.

References

[1] T.R. Gruber, A translation approach to portable ontology specifications, *Knowledge acquisition* **5**(2) (1993), 199–220.

[2] G. Quercini and C. Reynaud, Entity discovery and annotation in tables, in: *Proceedings of the 16th International Conference on Extending Database Technology*, ACM, 2013, pp. 693–704.

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

[4] A. Tonon, M. Catasta, G. Demartini, P. Cudré-Mauroux and K. Aberer, Trank: Ranking entity types using the web of data, in: *International Semantic Web Conference*, Springer, 2013, pp. 640–656.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] A. Alobaid, E. Kacprzak and O. Corcho, Typology-based Semantic Labeling of Numeric Tabular Data, *Semantic Web* (2020), accepted.

[11] A. Alobaid and O. Corcho, Knowledge-Graph-Based Semantic Labeling: Balancing Coverage and Specificity, *Semantic Web Journal* (2019), Under review, submitted on June, 2019.

[12] 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.

[13] J.D. Fernández, M.A. Martínez-Prieto, C. Gutiérrez, A. Polleres and M. Arias, Binary RDF Representation for Publication and Exchange (HDT), *Web Semantics: Science, Services and Agents on the World Wide Web* **19** (2013), 22–41–. http://www.websemanticsjournal.org/index.php/ps/article/view/328.

[14] I. Ermilov and A.-C.N. Ngomo, TAIPAN: automatic property mapping for tabular data, in: *European Knowledge Acquisition Workshop*, Springer, 2016, pp. 163–179.

[15] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, RML: a generic language for integrated RDF mappings of heterogeneous data (2014).

[16] 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.

[17] D. Ritze and C. Bizer, Matching web tables to DBpedia-a feature utility study, *context* **42**(41) (2017), 19.

[18] A. Alobaid and O. Corcho, tada-hdt-entity, Zenodo, 2020. doi:10.5281/zenodo.3732626.

[19] A. Alobaid, O. Corcho and W. Beek, tada-hdt-entity-experiment, Zenodo, 2020. doi:10.5281/zenodo.3732641.

[20] M. Arias, J.D. Fernández, M.A. Martínez Prieto, M. Vander Sande and R. Verborgh, HDT C++ Library and Tools, Zenodo, 2014. doi:10.5281/zenodo.580298.

[21] 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.

[22] D. Deng, Y. Jiang, G. Li, J. Li and C. Yu, Scalable column concept determination for web tables using large knowledge bases, *Proceedings of the VLDB Endowment* **6**(13) (2013), 1606–1617.

[23] V. Mulwad, T. Finin and A. Joshi, Semantic message passing for generating linked data from tables, in: *International Semantic Web Conference*, Springer, 2013, pp. 363–378.

[24] Z. Zhang, Towards efficient and effective semantic table interpretation, in: *International Semantic Web Conference*, Springer, 2014, pp. 487–502.

[25] 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.

[26] 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.

[27] 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.

[28] 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.