Semantic Web 13 (2022) 571–597
DOI 10.3233/SW-222993
IOS Press

Tab2KG: Semantic table interpretation with lightweight semantic profiles

Simon Gottschalk [a,*] and Elena Demidova [b]

[a] *L3S Research Center, Leibniz Universität Hannover, Hannover, Germany*
E-mail: gottschalk@L3S.de

[b] *Data Science & Intelligent Systems (DSIS), University of Bonn, Bonn, Germany*
E-mail: elena.demidova@cs.uni-bonn.de

Editors: Mehwish Alam, FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany; Davide Buscaldi, LIPN, Université Sorbonne Paris Nord, France; Michael Cochez, Vrije University of Amsterdam, Netherlands; Francesco Osborne, Knowledge Media Institute, (KMi), The Open University, UK; Diego Reforgiato Recupero, University of Cagliari, Italy; Harald Sack, FIZ Karlsruhe – Leibniz Institute for Information Infrastructure, Germany
Solicited reviews: Benno Kruit, Vrije Universiteit Amsterdam, Netherlands; five anonymous reviewers

Abstract. Tabular data plays an essential role in many data analytics and machine learning tasks. Typically, tabular data does not possess any machine-readable semantics. In this context, semantic table interpretation is crucial for making data analytics workflows more robust and explainable. This article proposes Tab2KG – a novel method that targets at the interpretation of tables with previously unseen data and automatically infers their semantics to transform them into semantic data graphs. We introduce original lightweight semantic profiles that enrich a domain ontology's concepts and relations and represent domain and table characteristics. We propose a one-shot learning approach that relies on these profiles to map a tabular dataset containing previously unseen instances to a domain ontology. In contrast to the existing semantic table interpretation approaches, Tab2KG relies on the semantic profiles only and does not require any instance lookup. This property makes Tab2KG particularly suitable in the data analytics context, in which data tables typically contain new instances. Our experimental evaluation on several real-world datasets from different application domains demonstrates that Tab2KG outperforms state-of-the-art semantic table interpretation baselines.

Keywords: Semantic table interpretation, domain knowledge graphs, semantic profiles, one-shot learning

1. Introduction

A vast amount of data is currently published in a tabular format [6,34,42]. Typically, this data does not possess any machine-readable semantics. Semantic table interpretation is an essential step to make this data usable for a wide variety of applications, with data analytics workflows (DAWs) as a prominent example [17]. DAWs include data mining algorithms and sophisticated deep learning architectures, and require a large amount of heterogeneous data as an input. Typically, DAWs treat tabular data as character sequences and numbers and potentially miss important information that has not been made available explicitly. This practice can lead to error-prone analytics processes and results, particularly when data analytics frameworks utilize the data from various heterogeneous sources. Therefore, DAWs can substantially profit from the semantic interpretation of the involved data tables [16,40].

*Corresponding author. E-mail: gottschalk@L3S.de.

In this context, semantic table interpretation aims to transform the input data table into a semantic data graph. In this process, table columns are mapped to a domain ontology's classes and properties; table cell values are transformed into literals, forming the data graph – a network of semantic statements, typically encoded in RDF. In the context of DAWs, semantic table interpretation can bring several advantages. First, an abstraction from tabular data to semantic concepts and relations can guide domain experts in the DAW creation [17]. Second, validation options (e.g., type inference) that become available through the semantic table interpretation can increase the robustness of DAWs [19]. Third, semantic descriptions can be employed to facilitate the explainability of the data analytics results [31]. Finally, semantic table interpretation adds structure directly usable for knowledge inference [32].

The existing approaches to semantic table interpretation do not adequately support the interpretation of tabular data for DAWs. At the core of such approaches (e.g., [8,11,37]) is the instance lookup task, where table cells are linked to known instances in a target knowledge graph, with DBpedia [5] being a popular cross-domain target. Subsequent steps such as property mapping are based on the results of this lookup step. However, as shown by Ritze et al. [44], only about 3% of the tables contained in the 3.5 billion HTML pages of the Common Crawl Web Corpus[1] can be matched to DBpedia. In the context of DAW, the input data typically represents new instances (e.g., sensor observations, current road traffic events, ...), and substantial overlap between the tabular data values and entities within existing knowledge graphs cannot be expected.

In this article, we present *Tab2KG* – a novel semantic table interpretation approach. *Tab2KG* aims to transform a data table into a semantic data graph. As a backbone of the data graph, *Tab2KG* relies on an existing domain ontology that defines the concepts and relations in the target domain. To facilitate the transformation, *Tab2KG* introduces original lightweight semantic profiles for domains and data tables. Domain profiles enrich ontology relations and represent domain characteristics. A domain profile associates relations with feature vectors representing data types and statistical characteristics such as value distributions. To transform a data table, *Tab2KG* first creates a data table profile. Then, *Tab2KG* uses the domain and data table profiles to transform the table into a data graph using a novel one-shot learning approach.

Lightweight semantic profiles generated by *Tab2KG* can be utilized as compact domain representations and complement and enrich existing dataset catalogs. Such profiles can be generated automatically from the existing datasets and described using the DCAT[2] and the SEAS[3] vocabularies to enable their reusability. We believe that lightweight semantic profiles presented in this article are an essential contribution that can benefit a wide range of semantic applications beyond semantic table interpretation.

In summary, *Tab2KG* is driven by the following three goals, which distinguish *Tab2KG* from traditional semantic table interpretation approaches:

- Interpretation of previously unseen data: *Tab2KG* can interpret domain-specific tables in the common case where the table values are not yet present in existing knowledge graphs.
- Usage of lightweight semantic profiles: *Tab2KG* relies on lightweight semantic profiles with profile features which can be easily modeled as part of data catalogs (e.g., histograms and data types).
- Generalisation through one-shot learning: *Tab2KG* generalises towards new domain ontologies with a one-shot learning approach.

Consequently, our contributions presented in this article are as follows:

1. We introduce lightweight semantic domain and table profiles. Domain profiles enrich relations of domain ontologies and serve as a lightweight domain representation. Semantic table profiles summarize data tables facilitating effective semantic table interpretation.
2. We propose the *Tab2KG* approach to transform tabular data into a data graph with one-shot learning based on semantic profiles.
3. We evaluate the proposed method on several real-world datasets. Our evaluation results demonstrate that *Tab2KG* outperforms state-of-the-art semantic table interpretation baselines.

[1] http://commoncrawl.org/
[2] https://www.w3.org/TR/vocab-dcat-2/
[3] https://ci.mines-stetienne.fr/seas/index.html

4. We make the scripts for creating lightweight semantic profiles and transforming data tables into data graphs publicly available.[4]

The structure of this article is as follows: In Section 2, we introduce a running example used throughout this article. Then, in Section 3, we define the problem of semantic table interpretation, followed by the definition and creation of domain and data table profiles (Section 4). In Section 5, we describe our proposed *Tab2KG* approach and its implementation (Section 6). We present evaluation setup and results in Sections 7 and 8, followed by a discussion of our profile-based approach in Section 9. Then, we discuss related work in Section 10. Finally, we provide a conclusion in Section 11.

2. Running example from the weather observation domain

As a running example, we use the weather observation domain and a data table that provides observations of sensors measuring air conditions.

Consider the table in Fig. 1 that contains weather observation sensor data, separated by a tab character (↠). The table does not include column titles. As a human, we can observe that the first column refers to the air condition (*cloudy*, *clear*, *rain*). The second and third column may represent a time interval of the measurement (e.g., *16:30* and *17:00*). The fourth column containing the values *S2*, *S3*, and *S1* is hard to interpret without background knowledge.

A domain ontology and a domain profile are required to map the table columns to their respective semantic concepts and to transform the whole table into a data graph.

- We use the Semantic Sensor Network Ontology[5] illustrated in Fig. 2 as **domain ontology**. Among others, this ontology provides classes to model sensors, their observations, and corresponding time intervals.
- Fig. 3 provides an exemplary illustration of a **domain profile** in the weather observation domain. Here, we illustrate statistical features of two observation properties (the beginning of the observation and the sensor label) using box plots and histograms.

```
cloudy ↠ 16:30 ↠ 17:00 ↠ S2
clear  ↠ 10:00 ↠ 10:30 ↠ S3
cloudy ↠ 17:00 ↠ 17:30 ↠ S3
rain   ↠ 16:30 ↠ 17:00 ↠ S1
cloudy ↠ 17:30 ↠ 18:00 ↠ S2
clear  ↠ 08:30 ↠ 09:00 ↠ S1
clear  ↠ 09:00 ↠ 09:30 ↠ S1
```

Fig. 1. Example of a data table as a tab-separated file without column titles.

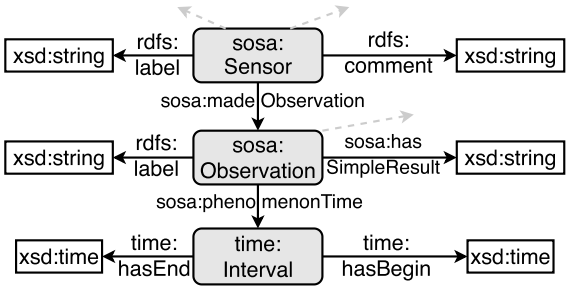

Fig. 2. Excerpt of the semantic sensor network ontology.

[4] https://github.com/sgottsch/Tab2KG
[5] `sosa:` https://www.w3.org/TR/vocab-ssn/

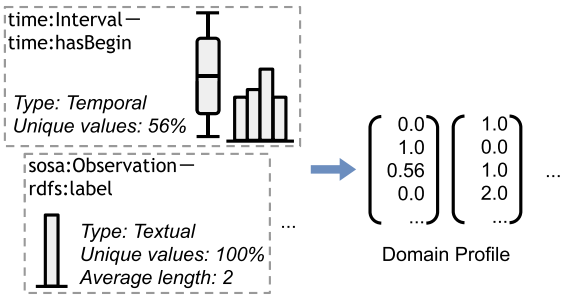

Fig. 3. An example profile of the weather observation domain. The domain profile is represented as a set of feature vectors, each containing statistical features, such as value distributions. The domain profile can also be used for visualization.

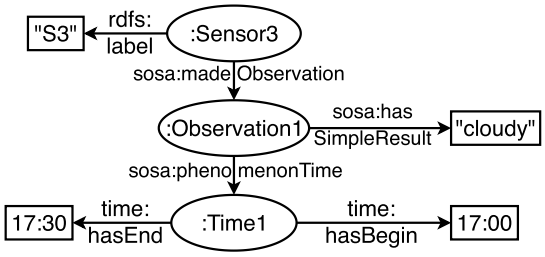

Fig. 4. Correct mapping of the third line in Fig. 1 to the ontology in Fig. 2. For brevity, we omit `rdf:type` relations.

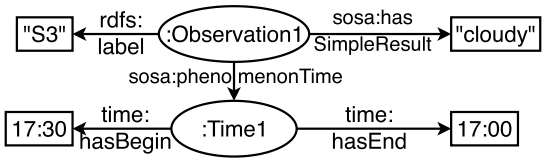

Fig. 5. Incorrect mapping of the third line in Fig. 1 to the ontology in Fig. 2. For brevity, we omit `rdf:type` relations.

Tab2KG transforms the table into the data graph shown in Fig. 4. As we can observe, the first three columns are mapped to the observations and their time intervals. The fourth column is mapped to the sensor labels. Note that not all properties of the example domain ontology are covered by the data graph, as the domain ontology contains relations not present in the table (e.g., `sosa:Observation` – `rdfs:label` – `xsd:string`).

The transformation process is challenging and potentially error-prone. For example, Fig. 5 illustrates a wrong transformation result, with an incorrect column mapping and an erroneous graph structure. In this case, the sensor label "S3" was erroneously interpreted as an observation label. In addition, the beginning and end times are swapped. *Tab2KG* utilizes semantic profiles to avoid such interpretation errors.

3. Problem statement

In this section, we first formally define relevant concepts. Then, we present the task of semantic table interpretation.

The entities and relations in the domain of interest (e.g., weather observation or soccer) can be represented in a domain knowledge graph.

Definition 1. A **domain knowledge graph** is a graph $G = (N, R)$, whose nodes N represent entities and literals, and whose edges R represent relations between these nodes in the specific domain.

A domain knowledge graph consists of two sub graphs: a domain ontology and a data graph.

Definition 2. A **domain ontology** $G_O = (N_O, R_O)$, $N_O \subset N$, $R_O \subset R$, where $N_O = C \cup D \cup P$ includes a set of classes C, a set of data types D, and a set of properties $P = P_d \cup P_o$ relevant in the specific domain, where P_d are data type properties, and P_o are object properties. Data type properties relate entities to literals. Object properties relate entities to each other.

The relations represented by $R_O = R_{\text{OC}} \cup R_{\text{OD}}$ include class relations R_{OC} and data type relations R_{OD}:

- R_{OC} is the set of class relations:
  $R_{\text{OC}} \subseteq C \times P_o \times C$.
- R_{OD} is the set of data type relations:
  $R_{\text{OD}} \subseteq C \times P_d \times D$.

For example, in the excerpt of Semantic Sensor Network Ontology illustrated in Fig. 2, (`sosa:Sensor sosa:madeObservation sosa:Observation`) is a class relation and (`sosa:Sensor rdfs:label xsd:string`) is a data type relation.

Definition 3. A **data graph** is a graph $G_D = (N_D, R_D)$, $N_D \subset N$, $R_D \subset R$. The nodes $N_D = C \cup D \cup E \cup L$ include classes C, data types D, entities E and literals L. Each literal $l \in L$ is assigned a data type $dt(l) \in D$. Within R_D, we distinguish between entity relations ($E \times P_o \times E$) and literal relations ($E \times P_d \times L$).

A data table is defined as follows:

Definition 4. A **data table** T is a $M \times N$ matrix consisting of M rows and N columns. A cell $T_{m,n}, m \in \{1, \ldots, M\}$, $n \in \{1, \ldots, N\}$, represents a data value. A row r_m is a tuple that represents a set of semantically related entities. A column c_n represent a specific characteristic of the entities in a row.

For example, the data table illustrated in Fig. 1 contains $M = 7$ rows and $N = 4$ columns, where the columns represent the weather conditions, time intervals, and sensor labels, and each row contains three semantically related entities: an observation, a time interval, and a sensor. The column values can belong to different data types, including text, numeric, Boolean, temporal and spatial.

Semantic table interpretation is the task of transforming a data table into a data graph.

Definition 5 (Semantic table interpretation)**.** Given a data table T and a domain knowledge graph G, create a data graph $G_D^T = (N_D^T, R_D^T)$ with nodes N_D^T and relations R_D^T. Its literal values $L^T \subseteq N_D^T$ are connected to the entities $E^T \subset N_D^T$ and represent the values in the literal columns of T. The entities E^T are connected via entity relations in R_D^T.

4. Semantic profiles

Semantic table interpretation in *Tab2KG* does not require any instance lookup in a domain knowledge graph. Instead, *Tab2KG* uses a domain knowledge graph to create a lightweight semantic domain profile. This domain profile, together with a domain ontology, builds reusable **domain background knowledge** that is later on used to interpret the data tables semantically.

The domain profile represents the domain knowledge graph regarding the value distributions. Note that the entities and literals in the domain knowledge graph do not need to overlap with the data tables' instances to be interpreted.

Tab2KG involves the creation of two types of profiles: *domain profiles* and *data table profiles*, both represented as feature vectors and described in a semantic data catalog to facilitate their reusability. Such profiles are inspired by the dataset profiles described in [15], where statistical features are defined as an important element of a dataset

profiles taxonomy. In *Tab2KG*, the primary purpose of the domain and data table profiles is to enable effective and efficient access to the domain and table statistics for semantic table interpretation.

We present domain profiles in Section 4.1 and data table profiles in Section 4.2. We discuss profile features in Section 4.3. Then, in Section 4.5, we describe how we represent domain and data table profiles in a semantic, machine-readable way. Finally, in Section 4.6 we provide an example of a data catalog that includes semantic profiles.

4.1. Domain profiles

For creating a domain profile, we make use of a domain knowledge graph G that contains representative values for the data type relations in the target domain. A **domain profile** $\Pi(G_O) = \{(r_D, \pi(r_D)) | r_D \in R_{\text{OD}}\}$ of a domain ontology $G_O = (N_O, R_{\text{OC}} \cup R_{\text{OD}})$ is a set of data type relation profiles $\pi(r_D)$ derived from G, where a data type relation profile is a set of features of the literals covered by this data type relation in G's data graph G_D.

Definition 6. The **data type relation profile** $\pi(r_D) \in \mathbb{R}^f$ of the data type relation $r_D \in R_{\text{OD}}$ is a vector that includes f statistical characteristics (features) of the literal relations covered in the domain knowledge graph G.

In brief, the profile of a data type relation r_D is a feature vector containing a set of statistics, computed using all literals corresponding to r_D. A description of the features of the literal relations follows in Section 4.3.

To create a profile for the data type relation $r_D = (c, p_d, d)$, we utilize all literals $l \in G_D$, such that: $(e, p_d, l) \in R_D$, $(e, \texttt{rdf:type}, c) \in R_D$, and $dt(l) = d$.

In our running example, in Fig. 4, the data type relation (`sosa:Sensor rdfs:label xsd:string`) in the domain ontology corresponds to the literal relation (`:Sensor3 rdfs:label` "S3") in the data graph. Therefore, we use "S3" as one of the literals to create the data type relation profile.

4.2. Data table profiles

To facilitate semantic interpretation of a data table, we create a data table profile.

A **data table profile** is a set of column profiles, each representing a specific data table column. More formally, the profile of a data table T consists of a column profile $\pi(c_n), n \in \{1, \ldots, N\}$ for each table column $c_n \in T$ as defined in Definition 4.

A column profile is defined as follows:

Definition 7. A **column profile** $\pi(c_n) \in \mathbb{R}^f$ of a column c_n is a vector of f features of the values contained in that column.

We create column profiles using literal values contained in the table columns.

Column profiles and data type relation profiles are created analogously and contain the same features, presented in Section 4.3.

4.3. Profile features

Motivated by the RDF profile characteristics defined by Ellefi et al. [15], we include data types, as well as completeness and statistical features described in the following into the profiles in *Tab2KG*. The selection is motivated by the expected feature effectiveness for semantic table interpretation, i.e., matching the domain and data table profiles. We demonstrate in our evaluation that these features can facilitate an effective matching in several application domains. This feature set can be extended to include relevant domains-specific characteristics.

- **Data type:** We represent data types as binary profile features. We include fine-granular data types to facilitate the precise matching of domain and data table profiles. The following data type taxonomy includes the most common cases observed in our evaluation domains.

- **Text:** Categorical, URL, Email, Other
- **Numeric:** Integer, Decimal / Sequential, Categorical, Other[6]
- **Boolean**
- **Temporal:** Date, Time, Date Time
- **Spatial:** Point, Linestring, Polygon

A data type relation or column can be assigned multiple (fine-granular) data types (e.g., integer and categorical). We provide technical details regarding the identification of fine-granular data types later in Section 6.

– **Completeness:** We include the number of values, the number of non-null values and the number of distinct values as a completeness indicator.
– **Basic statistics:** For numeric values, we include the standard deviation, mean, skewness, kurtosis and number of outliers computed using the interquartile range (1.5 IQR) rule, as well as the average numbers of characters, digits, tokens, capital letters and special characters for the literals.
– **Histograms:** Histograms are an effective means for RDF data summarization [20]. We create a histogram for a given number of buckets as part of the data type relation profile or column profile. As features, we add the number of literals in each bucket, in the increasing order of bucket ranges. For histogram creation, we remove the outliers detected before.
– **Quantiles:** We add quartiles and deciles to the profile (including minima and maxima).

To derive numerical features, we transform literals into numbers. The features of textual data type relations are computed based on the textual value lengths. Temporal values are transformed into timestamps. For spatial values, we consider the line string length or the polygon area, respectively.

4.4. *Profile-based semantic table interpretation*

Following the definition of domain profiles, their features and representation, we can now refine Definition 5. Instead of requiring a data graph G, *profile-based semantic table interpretation* solely requires a domain profile $\Pi(G_O)$.

Definition 8. Profile-based semantic table interpretation is the task of semantic table interpretation (Definition 5) of a data table T, given a domain ontology $G_O = (N_O, R_{\text{OC}} \cup R_{\text{OD}})$ and a domain profile $\Pi(G_O) = \{(r_D, \pi(r_D))|r_D \in R_{\text{OD}}\}$.

The domain profile is typically built from a domain knowledge graph G. As soon as the domain profile is created, the data graph of G is not required anymore for profile-based semantic table interpretation.

4.5. *Semantic profile representation*

Domain and data table profiles can be represented as **semantic profiles** in RDF, as an extension of the Data Catalog Vocabulary (DCAT)[7] and the SEAS Statistics ontology.[8]

Within the DCAT vocabulary, a data catalog (`dcat:DataCatalog`) consists of datasets (`dcat:Dataset`), where a dataset is a collection of data, published or curated by a single agent. In the context of *Tab2KG*, both the domain knowledge graph and the data tables can be represented using `dcat:Dataset`. We extend the descriptions of datasets in a *Tab2KG* data catalog to include semantic profiles. For example, we introduce an `Attribute` class representing the data table columns and data type relations. We make the definitions of this vocabulary available online.[9]

Figure 6 provides an overview of the classes involved in representing semantic profiles. A `dcat:Dataset` in a *Tab2KG* data catalog includes several attributes. In the case of a data table profile, these attributes are the columns.

[6] following the taxonomy defined in [3]
[7] https://www.w3.org/TR/vocab-dcat-2/
[8] https://ci.mines-stetienne.fr/seas/StatisticsOntology
[9] https://github.com/sgottsch/Tab2KG

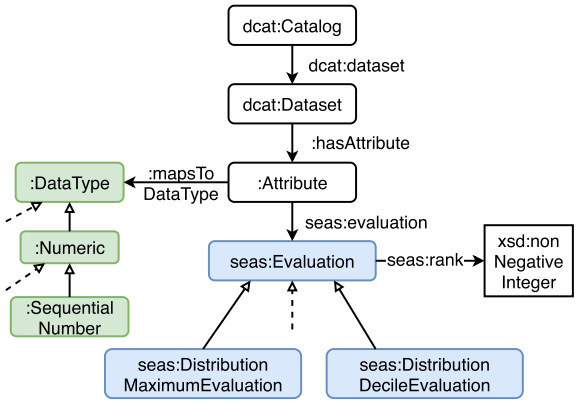

Fig. 6. Classes and properties used for describing a semantic profile. ⇾ marks `owl:subClassOf` relations. Dashed arrows indicate the existence of further classes which are not included in this excerpt. The two feature types (data types and numeric features) are grouped by different colors.

In the case of a domain profile, these attributes are the data type relations. These attributes are assigned the profile features, as presented in Section 4.3:

1. Data types: Data type assignments follow the previously mentioned taxonomy.
2. Numeric features: The numeric profile features are represented through subclasses of `seas:Evaluation`. In the case of quartiles, deciles, and histograms, the values come with a rank. For example, we can denote the second decile using `seas:rank 2`.

In the case of domain profiles, the existing mapping to the domain ontology can be modeled by connecting attributes to their corresponding classes, and data type properties [17].

Note that such semantic profiles do not only enable profile-based semantic table interpretation but can also be used to provide lightweight dataset visualizations, e.g., through box plots (quartiles) or as histograms.

4.6. Running example: Weather data catalog

An excerpt of an example data catalog for our running example from the weather observation domain introduced in Section 2 is shown in Fig. 7. The data catalog identified as `WeatherCatalog` includes two data tables (`RainData` and `AirData`). Here, the `AirData` data table has two columns, one of them with a column profile feature denoting the maximum value of the observation end time.

With *Tab2KG*, we can directly utilize this catalog for profile-based semantic table interpretation. Both example data tables can be interpreted through their data table profiles when a domain profile of the weather observation domain is provided.

5. *Tab2KG* profile-based semantic table interpretation

This section describes the process of *Tab2KG*'s profile-based semantic table interpretation.

5.1. Approach overview

Figure 8 provides an overview of our proposed *Tab2KG* approach to semantic table interpretation, where a data graph G_D^T is created from a data table T. To facilitate the interpretation, *Tab2KG* utilizes background knowledge that includes a domain ontology G_O and a domain profile. The domain profile is generated in a pre-processing step from a domain knowledge graph G.

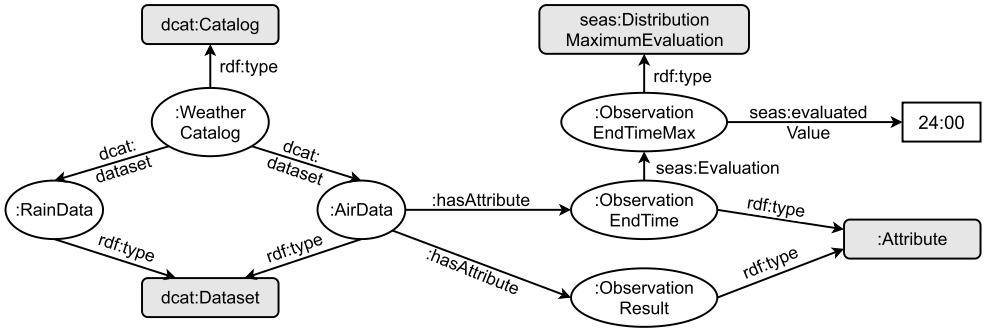

Fig. 7. Running example: excerpt of a weather data catalog containing two data tables and an exemplified column profile feature denoting the maximum end time value (`ObservationEndTimeMax`).

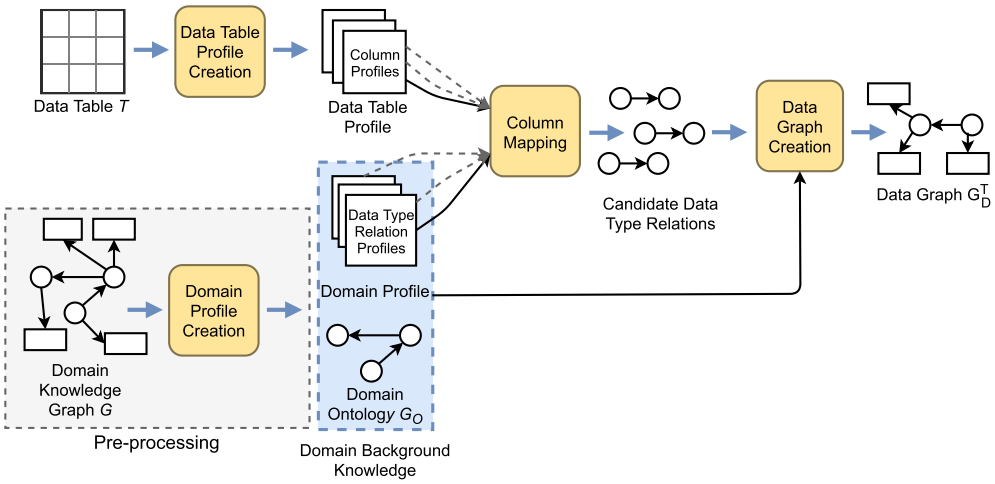

Fig. 8. An overview of semantic table interpretation with *Tab2KG*. Input is a data table T and a domain knowledge graph G. The output is a data graph G_D^T that represents the content of T as a data graph.

In brief, the *Tab2KG* pipeline consists of the following steps:

1. **Domain Profile Creation**: In a pre-processing step, we create a domain profile from a domain knowledge graph G.
2. **Data Table Profile Creation**: We create a profile of the input data table T.
3. **Column Mapping**: We generate candidate mappings between the columns of T and the data type relations in G_O using the domain profile, the data table profile, and a one-shot learning mapping function.
4. **Data Graph Creation**: We use the candidate column mappings and the domain ontology G_O to create a data graph G_D^T representing T's content.

In the following, we describe these steps in more detail.

5.2. *Domain profile creation*

The profile-based semantic table interpretation in *Tab2KG* requires the availability of a domain profile. This profile can be inferred from a domain knowledge graph G as described in Section 4.1. The domain profile is created by computing the feature values given the literal relations in the domain knowledge graph. This profile can be used as a lightweight domain representation. The domain profile can be created in a pre-processing step and become

available as part of a data catalog as described in Section 4.5. The domain profile does not contain any entities or literals from the domain knowledge graph G.

5.3. Data table profile creation

From the input data table T, we create a data table profile by computing the profile features based on the column values as described in Section 4.2.

5.4. Column mapping

With the help of the domain profile and the data table profile, we create *column mappings*.

Definition 9. A **column mapping** is a mapping from a column c_n in a data table T to a data type relation $r_D \in R_D$ in the domain ontology G_O: $c_n \mapsto r_D$.

For example, we can create a mapping from column c_2 of the data table illustrated in Fig. 1 to the data type relation (`time:Interval` – `time:hasBegin` – `xsd:time`) in the ontology illustrated in Fig. 2.

For the column mapping, we utilize a domain-independent *column mapping function*. Given a column profile $\pi(c_n)$ and a data type relation profile $\pi(r_D)$, this mapping function returns a similarity score.

The column mapping function is trained in a pre-processing step. It learns from positive pairs of columns and data type relations that should be mapped (similarity: 1.0) and negative pairs that should not (similarity: 0.0). After training, the function predicts the similarity of previously unseen pairs.

The architecture of the column mapping function is shown in Fig. 9. The column profile $\pi(c_n)$ and the data type relation profile $\pi(r_D)$ are taken as an input and normalized jointly. Then, their similarity is computed by a Siamese network that encodes both normalized profiles using the same weights and then predicts the similarity score based on the difference between the two profile encodings. As in [30], we use Rectified Linear Units for the hidden layers and a Sigmoid output layer. By using the Sigmoid output layer, the column mapping function learned by our Siamese network returns a similarity score in the range [0, 1].

Within the *Tab2KG* pipeline shown in Fig. 8, the column mapping function is utilized to create a set of candidate column mappings M_{c_n} for each column c_n in the data table T. In this step, we only consider mappings between columns and data type relations of the same data type (numeric, textual, temporal, spatial or Boolean).

The use of a Siamese network follows the idea used for one-shot learning for image classification [30]. Here, the task is not only to classify images into known classes (e.g., many images showing tigers) but also to generalize towards new classes (e.g., new images showing lions). That means, the underlying classifier needs to acquire features which enable the model to successfully generalize. This is done by inducing a metric that represents the domain-independent similarity between two input feature vectors (e.g., between an unknown image and a single image showing a lion).

As we cannot train a classifier on known classes (in contrast to domain-specific approaches such as Sherlock [23] and ColNet [8]), we are in a one-shot learning setting as well: We may learn how to map column profiles to known data type relations. But when facing a new domain, our classifier needs to generalize towards unseen data type

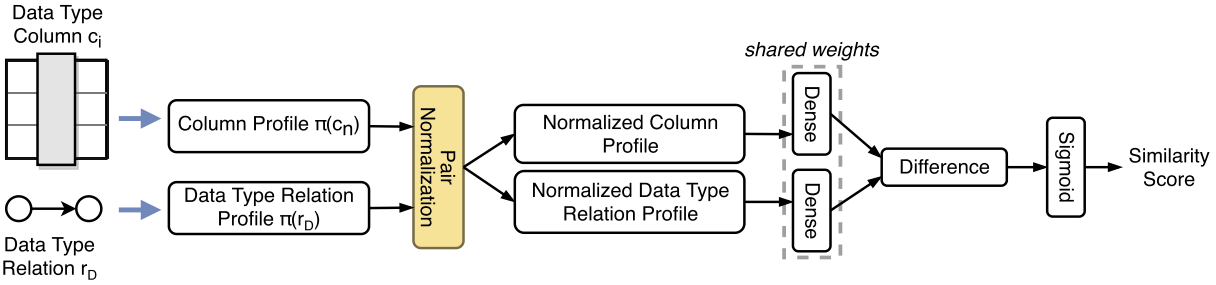

Fig. 9. Architecture of the mapping function to predict the similarity between a column c_n and a data type relation r_D.

relations. In *Tab2KG*, the similarity between a column and a data type relation is predicted based on the experience of the similarity of other profiles learned earlier.

The column profile and the data type relation profile are normalized jointly, i.e., the features are normalized in a range between 0.0 and 1.0 concerning the sum of the values in both profiles. This pairwise normalization is to ensure that the normalization happens within the same range, and this range covers all the values in both sources. If both profiles were normalized in isolation, different maximum values would be both normalized to 1.0 such that the values from different profiles were not comparable.

5.5. Data graph creation

Given a set of candidate column mappings M_{c_n} with similarity scores for each column c_n in the input data table T, we now map each table column to a data type relation in a greedy manner. First, we take the column mapping with the highest similarity score. Then, we remove all candidate column mappings with the particular column or data type relation. These two steps are repeated until all columns are mapped to a data type relation.

From the chosen column mappings set, we create the data graph G_D^T that contains all data type relations resulting from the chosen mapping. G_D^T needs to adhere to the following four conditions:

1. The data graph covers all literal columns of T, and each literal column has exactly one mapping to a data type relation.
2. The set of entities in a table row is connected via entity relations.
3. G_D^T is minimal, i.e., no relation can be removed without invalidating the previous two conditions.
4. Each class relation represented by G_D^T is connected to at least one class that is part of M_{c_n}. This condition ensures semantic closeness of the data table columns and reduces the number of potential paths in the graph.

5.6. Creation of training instances for column mapping

For the computation of the column mapping function, we utilize a Siamese network trained once in a pre-processing step. This training process requires the extraction of positive and negative training instances. Following Definition 5 (see also Fig. 8), this step requires a set of (G, T, G_D^T) triples, where G is the domain knowledge graph, T is the data table and G_D^T is the resulting data graph. For each triple (G, T, G_D^T), each pair of data type relations in G and a column in T is a positive training instance. We select the remaining (data type relation, column) pairs from the same knowledge graph G as negative instances.

In the first step, we synthetically create a set of (G, T, G_D^T) triples for the model training, intending to have a large dataset of positive and negative examples derived from existing data tables and knowledge graphs. Such data, i.e., a diverse set of tables and their mapping definitions to different domain ontologies, is difficult to obtain, except for manually created, task-specific research datasets [39], which are not large enough for training deep neural networks and do not provide enough topical and structural diversity. Therefore, we utilize existing knowledge graphs to create training data.

Given a set of knowledge graphs, we create a new dataset of triples (G_1, T, G_2^T). Each input knowledge graph G is disjunctly (i.e., without shared triples) split into two KGs: G_1 and G_2. As G_1 and G_2 do not overlap, it is ensured that the domain profile and the data table profiles used in training are not generated from the same values. G_1 represents the domain knowledge graph, while G_2 is transformed into a data table T. The transformation of G_2 into T is based on a set of domain ontology templates. A template is a directed tree with up to k nodes, where k is a parameter. The nodes and edges of these trees are placeholders for classes and properties. A set of trees is transformed into domain ontologies by replacing these placeholders with the classes and properties of G_2. From the knowledge graph G_2, a data graph G_2^T is extracted and transformed into a data table T to create the triple (G_1, T, G_2^T).

We aim to retrieve a set of heterogeneous data tables that represent the original knowledge graph characteristics. Therefore, the data table creation process incorporates several stochastic decisions in proportion to the knowledge graph statistics:

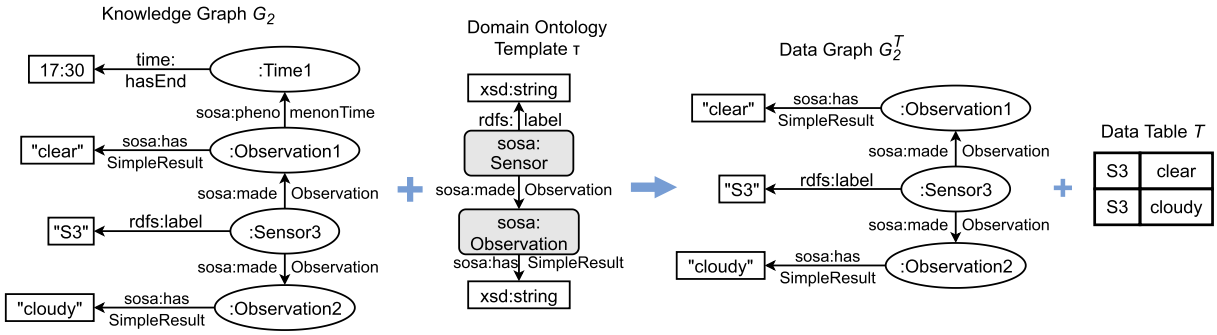

Fig. 10. Creation of a data table T and a data graph G_2^T from a knowledge graph G_2 and a domain ontology template τ. For brevity, we omit type relations.

- Entities and entity relations (and consequently, the literal relations) in G are split at a random ratio between 25% and 75% into G_1 and G_2, whereas their domain ontologies remain the same.
- During the template creation, classes, and properties are randomly mapped to a domain ontology template, proportionally to their occurrence rate in G.
- Data type relations are added in the same manner, under the condition that each leaf node has to be connected with at least one data type relation. After adding the minimal required number of data type relations, we add data type relations as long as any of them are left and if a randomly generated number between 0 and 1 exceeds a predefined threshold δ.

5.7. *Running example: Data table creation*

For our running example introduced in Section 2, Fig. 10 illustrates the transformation of a knowledge graph G_2 and a domain ontology template τ into a data table T. This transformation is performed after splitting the input knowledge graph G into G_1 and G_2, where G_1 and G_2 represent different portions of G (e.g., G_1 contains a third observation of `Sensor3`). Consequently, the triple (G_1, T, G_2^T) can be used as a training example where the feature weights to map T to G_2^T based on G_1's domain profile and T's data table profile are learnt.

6. Implementation

Tab2KG is implemented in Java 1.8. The Siamese network is trained and applied using Keras in Python 3.7. We load knowledge graphs using Apache Jena.[10] We represent the column mappings inferred by *Tab2KG* in the RDF Mapping Language (RML) [13]. RML definitions are then utilized to materialize the data graph. Data tables are provided as CSV files; knowledge graphs and data graphs as Turtle (*.ttl*) files.

6.1. *Data table pre-processing*

Each data table interpreted in *Tab2KG* runs through three pre-processing steps:

1. **Data type identification**: For each table column, we identify its data type(s) by trying to parse more than 90% of the values as numeric, Boolean, spatial (Well-Known Text or Well-Known Binary format [25]) or temporal (using the dateparser library[11]). If that is not possible, the column is assigned the text data type. For the more fine-grained data types mentioned in Section 4, we utilize regular expressions (text: URL or email), follow the algorithms proposed by Alobaid et al. [3] (numeric: sequential or categorical) or analyze the parsed objects

[10] https://jena.apache.org/
[11] https://github.com/sisyphsu/dateparser

(temporal: date, time, date-time; spatial: point, line string, polygon). We follow the algorithm in [3], using the threshold of not more than 20 different categories to detect categorical text values.

2. **Key column detection**: When we transform a data table into a data graph based on the RML mapping, we create new entities. In RDF, each entity is identified by a Unique Resource Identifier (URI). It is important to understand how to create these URIs, as we need to re-use URIs referring to the same entity: For example, each row in our running example in Fig. 1 forms a new instance of `sosa:Observation`, together with a new URI (e.g., `sosa:Observation7`), but there should only be three different `sosa:Sensor` URIs: `sosa:SensorS1`, `sosa:SensorS2`, `sosa:SensorS3`, created using the literal values of the data type property `rdfs:label`.
   To create URIs, we detect data type relations representing the unique literal values of an entity as follows: (i) the data type relation is used on all instances of the class exactly once, and the literal values are unique across their instances. Currently, we do not consider the combinations of literal relations as identifiers [21]; we leave such combinations for future work.
3. **Identifier generation**: RML transformation requires referenceable columns and instances in data tables. Therefore, we automatically generate identifiers for each column and row of the data table. If available, column names are used as part of the column identifier.

6.1.1. Mapping representation in RML

We utilize RML for storing column mappings inferred by *Tab2KG* in a machine-readable format such as Turtle. The RML defines subject maps that specify how to generate subjects for each row of the data table and predicate-object maps to create predicate-object pairs. In *Tab2KG*, the inferred column mappings are translated into the RML definitions according to the following four steps:

1. We create one instance of `rml:source` and `csvw:Table` each, denoting relevant characteristics for parsing the data table T (file location, delimiter, ...).
2. For each class part of the data graph G_D^T, we create a new instance of `rr:TriplesMap`, together with a `rr:subjectMap` that denotes the class as well as the target node URIs.
3. For each column mapping $c_n \mapsto r_D$, we create a `rr:predicateObjectMap` denoting the source column c_n, the data type property and a reference to the data type relation r_D.
4. For each class relation $r \in R_{\text{OC}}$ in the domain ontology G_O, we create a `rr:predicateObjectMap` connecting the respective entities and the object property.

6.1.2. Running example: RML mapping definitions

Listing 1 in the Appendix provides an example of the RML definitions that were automatically generated for our running example introduced in Section 2 – without the time intervals, for brevity. Instances of `sosa:Sensor` and `sosa:Observation` are created alongside their relations. The sensor labels in the third column were detected as identifiers, i.e., we create node URIs as *https://www.w3.org/TR/vocab-ssn/Sensor{col3}*.[12]

Listing 2 in the Appendix provides the resulting Turtle file representing the knowledge graph inferred from the input data table. The correct mapping of the third line shown in Fig. 4 is contained here.

7. Evaluation setup

The goal of the evaluation is to assess the performance of *Tab2KG* concerning the semantic table interpretation effectiveness using lightweight semantic profiles.

In this section, we will first describe how the data tables and domain ontologies are selected, how data table profiles and domain profiles are built and which baselines we compare to.

We make the scripts to extract the evaluation datasets and the code for training the Siamese network publicly available.[13]

[12] The RML template definitions do not allow to use prefixes.

[13] https://github.com/sgottsch/Tab2KG

Table 1

Domain ontologies used during evaluation, together with their used vocabulary, the number of classes and literal relations. In total, *Tab2KG* is evaluated against 34 domain ontologies. ST and SE use the same domain ontologies which are grouped together as ST_S and SE_S, respectively

Domain	Vocabulary	#Classes	#Relations	Classes
So_S: Soccer	schema.org	4	21	Player, League, SportsClub, SportsTeam
WA_S: Weapon ads.	schema.org	7	37	PersonOrOrganization, Place, PostalAddress, Firearm, Offer, ...
ST_S, SE_S: City	DBpedia	15	55	EducationalInstitution, BasketballPlayer, BasketballTeam, Settlement, ...
ST_S, SE_S: University	DBpedia	14	52	City, OfficeHolder, Person, Country, Artist, Organisation, ChristianBishop, ...
ST_S, SE_S: Governor	DBpedia	9	23	City, Person, Country, University, Senator, OfficeHolder, Town, Region
ST_S, SE_S: Building	DBpedia	9	25	Architect, GovernmentAgency, City, Country, Region, Organisation, ...
ST_S, SE_S: Song	DBpedia	8	21	MusicalArtist, Band, Single, Company, City, Settlement, PopulatedPlace
ST_S, SE_S: Play	DBpedia	6	15	Person, Language, Writer, MusicalArtist, Album
				⋮

7.1. Domain ontologies and vocabularies

In the real-world applications of *Tab2KG*, we require a domain ontology and a dataset that is modelled using this ontology to build the domain profile.

In our experiments, we aim to assess the performance of *Tab2KG* in different domains for mapping real-world tables. This evaluation requires annotated datasets which provide the tables and their ontology mappings. In the existing datasets such as SemTab [26], the tables are typically annotated using cross-domain ontologies such as DBpedia and schema.org. To facilitate evaluation, we extract domain ontologies from tables within specific domains. This extraction results in the domain-specific parts of the existing cross-domain ontologies.

For the ontology extraction, we use two approaches: a *pairwise setting* (P) that uses one table and a *set-based setting* (S) that uses a union of tables to represent a domain. These approaches are discussed in Section 7.3.

Statistics of the domain ontologies used in the evaluation and examples of their involved classes are presented in Table 1.

7.2. Datasets

The Siamese network training requires a dataset that spans multiple vocabularies and domains to ensure generalization. However, the domain ontologies shown in Table 1 are based on schema.org and the DBpedia ontology only. To feed and evaluate the Siamese network with data from a more diverse set of vocabularies, we created a new synthetic dataset automatically extracted from GitHub repositories dealing with knowledge graphs. The column mapping function trained on the train split of this dataset is expected to generalise to other domains and vocabularies represented in our datasets. Therefore, we use this mapping function for all datasets in the evaluation and do not train any dataset-specific column mapping functions.

7.2.1. Synthetic GitHub dataset

The GitHub advanced code search[14] provides access to millions of data repositories. To collect knowledge graphs from GitHub, we selected files larger than 5 KB with the specific file extensions[15] that contain the text "`xsd`" or "`XSDSchema`". To ensure the heterogeneity of our dataset, we limited the number of files per GitHub repository to three. Each file that was successfully parsed as a knowledge graph with more than 50 statements including at least 25 literal relations was added to our dataset. This way, we obtained 3,922 files.

We set the parameter for maximum tree size $k = 3$, and the parameter for adding data type relations $\delta = 0.2$. The knowledge graphs set was split into a training set (90%) and a test set (10%). Knowledge graphs from the same repository were not included in the same set.

[14] https://github.com/search/advanced

[15] `ttl`, `rdf`, `nt`, `nq`, `trix`, `n3`, `owl`

7.2.2. Test datasets

We use the following datasets for evaluating our approach:

- **GitHub (GH):** The test split of the synthetic GitHub dataset, without a restriction on the used vocabularies.
- **Soccer (So):** 12 data tables regarding soccer players and their teams, annotated with the *schema.org* vocabulary [39].
- **Weapon Ads (WA):** 15 data tables about weapon advertisements, annotated with the *schema.org* vocabulary [48].
- **SemTab (ST):** A collection of 2, 281 data tables extracted from the T2Dv2 web table corpus, Wikipedia and others, annotated with the DBpedia ontology [26].
- **SemTab Easy (SE):** A subset of ST, including those 676 data tables whose columns are mapped to one class only. Only classes appearing in the *T2KMatch* corpus [43] are included.

For all data tables contained in these datasets, we set the following constraints:

1. The input table file is parseable as a CSV file without errors.
2. There are no classes that are instantiated multiple times in the same row. This condition is to avoid cyclic structures. We discuss limitations regarding cyclic structures in Section 9.1.
3. There is no pair of columns with identical values in the same table. This condition is to avoid randomness during the evaluation.
4. The table has at least two columns which are mapped to a data type relation. This condition is to ensure that the semantic table interpretation task does not become trivial in such cases.

Following the process to create data tables via domain ontology templates described in Section 5.5, the GH data tables did not violate these conditions. Also, these conditions were not violated by the So and WA data tables. In the case of the SemTab data tables, we had to skip 3, 017 non-parseable data tables (e.g., because of column values containing non-escaped delimiters such as in *"Bret "Hitman" Hart"*, 2, 233 cyclic data tables (e.g., mathematicians and their supervisors), 520 data tables with identical column values (e.g., columns mapped to `dbo:birthDate` and `dbo:birthYear` but with identical values) and 6, 179 data tables with one column only.

7.3. Profile & test data generation

It is important to emphasize the difference in the evaluation setting compared to typical evaluation using the previously mentioned datasets such as ST: In the experiments conducted by [9,39,52], target general-purpose knowledge graphs such as DBpedia or Wikidata are given. Each data table in the test set is then mapped to the nodes in such a knowledge graph. In our evaluation setting, we assume that no data instances are given, i.e., an instance lookup is not possible.

Instead, a domain profile and a domain ontology are provided which are derived via the pairwise and the set-based setting. Figure 11 exemplifies these two approaches which are described in more detail in the following. While we consider pairs of data tables in the pairwise setting, we create a domain ontology based on a set of tables in the set-based setting.

The set-based setting and the pairwise setting reflect different application scenarios of *Tab2KG*. The pairwise setting considers smaller domain ontologies, where the directly corresponding concepts are available in the ontology and a table. An example task for the pairwise setting is table augmentation, where a given table is populated with additional rows, columns or cell values [50]. The set-based setting reflects traditional semantic table interpretation and involves larger domain ontologies, with concepts potentially represented in several tables.

7.3.1. Pairwise setting

In the pairwise setting, we select data table pairs, such that one data table mimics the domain knowledge graph, from which we can derive a domain profile.

Following our setting defined in Definition 5 and illustrated in Fig. 8, the datasets are transformed into a set of triples (G, T, G_D^T), consisting of a data table T, a domain knowledge graph G and the mapping definition which transforms data table T into the data graph G_D^T. Technically, we extract a set of test instances, consisting of (i) a *.ttl* file representing the domain knowledge graph G, and (ii) a *.csv* file representing the data table T and a *.rml*

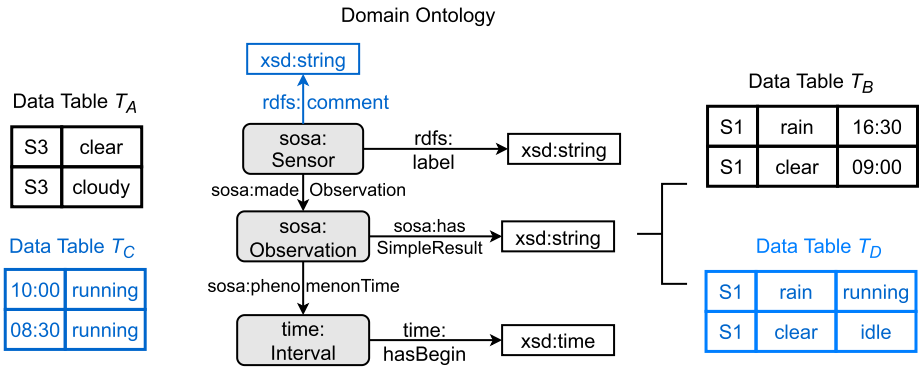

Fig. 11. Comparison of the pairwise (black) and the set-based setting (black and blue) for creating training and test instances. In the pairwise setting, one data table (T_B) represents the domain knowledge of the other data table (T_A). In the domain ontology approach, we first create a domain ontology from a set of data tables (T_B and T_D) and then interpret several data tables (T_A and T_C) against the domain ontology each. In this example, "running" and "idle" are comments (`rdfs:comment`) about specific sensors.

Table 2

Datasets used in the evaluation based on the pairwise setting. # is the number of (G, T, G_D^T) triples that we generated. The other columns contain average values. For example, the tables (T) in the ST dataset have about four columns on average, while their paired domain knowledge graphs have 5.5 data type relations on average. Following the pairwise extraction process, the number of data type relations in the domain knowledge graphs is always equal or higher than the number of columns in the paired tables

Dataset	#	Tables	Domain knowledge graphs	
		Columns	Data type relations	Class relations
GitHub$_P$ (Training)	8,078	2.18	3.03	1.35
GH$_P$: GitHub (Test)	891	2.24	3.16	1.44
So$_P$: Soccer	15	6.25	9.38	2.75
WA$_P$: Weapon-Ads	16	10.57	11.2	4.4
ST$_P$: SemTab	233	4.09	5.54	1.26
SE$_P$: SemTab Easy	125	4.2	5.53	0.0

file representing the mapping from T to G's domain ontology. To transform the datasets into such test instances, we identify pairs of data tables where the columns of the first data table are a subset of the second data table's columns (i.e., the columns of the first data table are all mapped to data type relations which also exist in the column mapping of the second data table). Then, the second data table represents the domain knowledge. Table 2 provides an overview of the datasets used during training and evaluation under these conditions. ST and SE cover all data tables from the domain ontologies illustrated in Table 1.

7.3.2. *Set-based setting*

In the set-based setting, we create a set of domain ontologies, each based on a set of data tables. For So and WA, we create one domain ontology each, based on all available data tables. In the case of ST and SE, we create several domain ontologies by grouping together data tables based on the class which most columns are mapped to. We only consider domain ontologies from at least 10 data tables and with at least five data type relations. Due to the diversity of domains in the GitHub data set, we do not evaluate GH with the set-based setting.

Table 1 provides an overview of the domain ontologies used in our evaluation.

7.4. *Baselines*

We compare *Tab2KG* against the following three semantic table interpretation baselines:

– ***DSL***: The Domain-independent Semantic Labeler [39] uses logistic regression on a set of hand-crafted features; some of them compare value pairs at the instance-level. It has been shown to outperform previous approaches such as the SemanticTyper [41]. We train *DSL* on the same GitHub training data set as *Tab2KG*.

- ***DSL****: The Domain-independent Semantic Labeler without using value similarity at the instance level. In contrast to *Tab2KG* and the other baselines, *DSL* utilizes a domain-specific data graph. As *Tab2KG* is solely based on the domain profile, *DSL* is in an advantageous setting that does not entirely reflect our setting. Therefore, we remove features at the instance-level for the *DSL** baseline.
- ***T2KMatch***: *T2KMatch* [43] performs semantic table interpretation on the instance level. In contrast to other approaches [11,37,52] that rely on a costly knowledge graph lookup at runtime, *T2KMatch* creates an index over the instances of frequently used DBpedia classes and is thus commonly used as a baseline for semantic table interpretation approaches [8,14,51]. It combines a ranking of entities found in the lookup phase for column type identification and data type-specific similarity measures (Levenshtein distance for strings, deviation for numbers, and deviation of years for dates) for property identification. *T2KMatch* assumes that a table only describes one entity class at a time and thus does not consider class relations.

8. Evaluation results

In this section, we describe the training performance and the evaluation results based on the evaluation setup described before.

8.1. Accuracy of the column mapping function

We train our Siamese network on the GitHub training dataset for 1,000 epochs, a batch rate of 100 and a learning rate of 0.0001, following Hsiao et al.'s approach for one-shot image classification [22]. We apply an early stopping criterion if the validation accuracy has not improved within 100 epochs. We use 256 dimensions for the hidden layer. 10% of the training dataset are used for validation. The feature vectors contain histograms with 10 buckets.

After training for all epochs on the synthetic training dataset, the Siamese network has an accuracy of 0.959 and a loss (binary cross-entropy) of 0.149 on the validation set. On the test set, it achieves an accuracy of 0.952, when treating scores of greater than 0.5 as candidates for column mapping.

We experimented with different train/test splits (90/10 and 80/20). We achieve a similar accuracy on the splits, namely 0.948 on the 80/20 and 0.952 on the 90/10 split. Whereas slight variations are expected, we do not observe any remarkable drop in performance if less training data is provided to the algorithm.

8.1.1. Hyperparameter tuning

To identify the best combination of hyperparameters, we performed a grid search over the number of hidden layers and their dimensions. The results with respect to the classification accuracy are shown in Table 3. Following these results, using one hidden layer is sufficient, and we set its dimension to 256.

8.1.2. Ablation study

To determine the effect of the selected features on the model's performance, we have performed an ablation study where we trained the Siamese network after removal of specific feature groups as listed in Section 4.3. Table 4 shows the model's classification performance on the GH test set. As expected, the combination of all selected features into the profiles leads to the highest accuracy. Among the different feature groups, the removal of the basic statistics features leads to the highest drop in performance. In general, the ablation study demonstrates that all features contribute to increase the accuracy of *Tab2KG*.

Table 3

Grid search results (accuracy on the test set) over the number of hidden layers and their dimension. Higher accuracy values are marked darker

		Hidden layers dimension					
		16	32	64	128	256	512
# Hidden layers	1	0.947	0.943	0.949	0.950	0.952	0.937
	2	0.935	0.942	0.946	0.948	0.939	0.907
	3	0.891	0.930	0.909	0.934	0.939	0.821

Table 4
Ablation study: number of true positives (TP), false positives (FP), true negatives (TN), false negatives (FN) and accuracy (acc.) on the test set

	TP	FP	TN	FN	Acc.
All features	1,873	125	1,810	62	**0.952**
No data type features	1,895	251	1,684	40	0.924
No completeness features	1,865	141	1,794	70	0.945
No basic statistical features	1,822	194	1,741	113	0.921
No histograms and quantiles	1,862	132	1,803	73	0.947

Table 5
Semantic table interpretation performance of *Tab2KG*, compared to the baselines on five datasets. We report the accuracy, i.e. the percentage of correctly identified data type relations (R_{OD}) and class relations (R_{OC}) in the datasets. Averages are computed in relation to the number of class relations plus data type relations in the datasets. T2KMatch is only evaluated on SE_P and SE_S, as it assumes one entity class per table only

	GH_P	So_P	So_S	WA_P	WA_S	ST_P	ST_S	SE_P	SE_S	Average_P	Average_S	Average
DSL	0.89	0.56	0.52	0.40	0.39	0.75	0.51	0.81	0.56	0.79	0.52	0.62
*DSL**	0.87	0.45	0.46	0.44	0.36	0.68	0.52	0.75	0.59	0.76	0.54	0.61
T2KMatch	–	–	–	–	–	–	–	0.41	0.35	0.41	0.35	0.36
Tab2KG	0.89	0.69	0.54	0.49	0.43	0.81	0.66	0.84	0.69	**0.84**	**0.66**	**0.72**

8.2. Semantic table interpretation results

Until now, we evaluated the accuracy of the column mapping function, i.e. *Tab2KG*'s ability to provide correct data type relation mappings. Now, we evaluate the entire semantic table interpretation process. While we expect a similar accuracy for data type relation mappings for the GitHub dataset, the results depend on the accuracy of class relations mappings and can vary across datasets. The evaluation in this section follows three steps: we measure accuracy (i) of the whole semantic table interpretation, (ii) of the column mapping and graph creation in isolation, and (iii) of the semantic table interpretation on different domains.

8.2.1. Accuracy of the semantic table interpretation

We evaluate the performance of the semantic table interpretation achieved by *Tab2KG* compared to the baselines. Table 5 shows how the approaches perform on the different datasets, measured using accuracy, i.e., the percentage of the columns correctly mapped to data type relations and correctly identified class relations. We do not evaluate the performance of *T2KMatch* on other datasets than SE, as *T2KMatch* assumes one entity class per table only.

As we can observe in Table 5, the accuracy of the approaches varies considerably across the datasets, which can be explained by the different domain ontology and dataset characteristics shown in Table 1 and Table 2.

All methods perform worst on the WA dataset, which has the most columns, data type relations and class relations within the datasets and domain ontologies. This result unsurprisingly demonstrates that semantic table interpretation gets more difficult the more columns and relations are involved. In all cases except for the GH dataset, where *Tab2KG* and *DSL* show similar performance, *Tab2KG* achieves higher accuracy than the baselines concerning column mapping. Even though *Tab2KG* utilizes less information than *DSL*, *Tab2KG* performs better by 10 percentage points on average on this task.

The semantic interpretation of tables with domain ontology that were created with the set-based setting is more difficult than with the pairwise setting (*Tab2KG* accuracy: 0.84 vs. 0.66). While the domain ontologies of the set-based setting provide richer domain knowledge, the increased number of classes and relations, i.e., the increased number of candidate mappings, leads to more misinterpretations.

Surprisingly, *DSL* is also outperformed by *DSL** in three cases (WA_P, ST_S, SE_S). To explain this behavior, we have computed the percentage of table values that also appear in the mapped data table relations: GH_P (33.38%), So_P (16.92%), ST_P (14.67%), SE_P (10.63%), WA_P (2.65%). This observation shows that profile-based semantic table interpretation can outperform instance-level approaches when the overlap between the data table and the instances in the domain knowledge graph is low.

Table 6

Semantic table interpretation performance of *Tab2KG* in detail, compared to the baselines on five datasets created in the pairwise setting. The results are reported as the accuracy of class relations (R_{OC}) and data type relations (R_{OD}). Averages are computed in relation to the number of class relations and data type relations in the datasets, respectively. T2KMatch is only evaluated on SE_P, as it assumes one entity class per table only

	GH_P		So_P		WA_P		ST_P		SE_P	$Average_P$	
	R_{OC}	R_{OD}	R_{OC}	R_{OD}	R_{OC}	R_{OD}	R_{OC}	R_{OD}	R_{OD}	R_{OC}	R_{OD}
DSL	0.62	0.93	0.69	0.52	0.44	0.38	0.88	0.73	0.81	0.64	0.83
*DSL**	0.50	0.94	0.50	0.43	0.42	0.45	0.72	0.68	0.75	0.52	0.81
T2KMatch	–	–	–	–	–	–	–	–	0.41	–	0.41
Tab2KG	0.73	0.94	0.73	0.68	0.23	0.61	0.78	0.81	0.84	**0.69**	**0.87**

Table 7

Semantic table interpretation performance of *Tab2KG* in detail, compared to the baselines on four datasets created using the set-based setting. The results are as the accuracy of class relations (R_{OC}) and data type relations (R_{OD}). Averages are computed in relation to the number of class relations and data type relations in the datasets, respectively. T2KMatch is only evaluated on SE_S, as it assumes one entity class per table only

	So_S		WA_S		ST_S		SE_S	$Average_S$	
	R_{OC}	R_{OD}	R_{OC}	R_{OD}	R_{OC}	R_{OD}	R_{OD}	R_{OC}	R_{OD}
DSL	0.96	0.39	0.51	0.34	0.32	0.54	0.56	0.36	0.54
*DSL**	0.78	0.37	0.49	0.30	0.31	0.55	0.59	0.35	0.54
T2KMatch	–	–	–	–	–	–	0.35	–	0.35
Tab2KG	0.70	0.49	0.40	0.43	0.54	0.68	0.69	**0.53**	**0.67**

8.2.2. *Accuracy of the column mapping and graph creation*

Now, we assess the results of the column mapping (percentage of correctly mapped columns to the data type relations R_{OD}) and the graph creation (correctly identified class relations R_{OC}) in isolation. We report the results achieved by *Tab2KG* in comparison to the baselines in Table 6 (pairwise) and in Table 7 (set-based). Regarding column mapping in the pairwise setting, *Tab2KG* performs best on average, outperforming *DSL* by 4 percentage points. A similar distribution of results, but with less accuracy, can be observed in the set-based setting.

The high accuracy of 0.95 for the column mapping on the GitHub test dataset reported in Section 8.1 is confirmed in Table 6. Accuracy drops slightly to 0.94 which can be explained by the greedy process described in Section 5.5: during the data graph creation, the column mappings are not decided in isolation: an erroneous selection of a column mapping influences the subsequent selections.

In general, the class relation mapping results in less accuracy than the column mapping. One reason is that errors propagate along the pipeline, i.e., a wrongly mapped data type relation invokes an erroneous class relation mapping.

8.2.3. *Accuracy on different domains*

ST_S and SE_S cover various domains as shown in Table 1. We evaluated the accuracy of *Tab2KG* on ST_S regarding the different domains. For the domains with a large number of classes and relations shown in Table 1, the following accuracy is achieved: Play (0.70), Building (0.68), Song (0.55), City (0.49), University (0.37) and Governor (0.35). Three domains with a lower number of classes and relations, namely Airport, Scientist and Ligament, achieve an accuracy of 1.0.

8.3. *Error analysis*

By inspection of the results, we have identified two typical sources of erroneous results in *Tab2KG*:

- Value formatting: For example, the soccer dataset has data tables with column values such as “Germany”, whereas the domain knowledge graphs had “GER” as a country label. Thus, the respective profile features were highly different. Regarding high-quality domain knowledge graphs that, for example, distinguish between labels and abbreviations, the error rate should be lower.

– Data type differences: For example, the SemTab dataset has elevations of mountains both denoted via integer values ("5291") and as text ("2,858 ft (871 m)"). Again, the proper use of an ontology and its `rdfs:range` constraints on properties should alleviate this problem.

Overall, our evaluation results demonstrate that domain profiles in combination with one-shot learning adopted by *Tab2KG* are an effective method for semantic table interpretation. This method does not require any instance lookup and achieves the highest accuracy on several datasets compared to the baselines.

9. Discussion

In this article, we presented *Tab2KG* – an approach for tabular data semantification. *Tab2KG* relies on domain profiles that enrich the relations in a domain ontology and serve as a lightweight domain representation. *Tab2KG* matches these profiles with tabular data using one-shot learning. Our evaluation shows that *Tab2KG* outperforms the baselines for semantic table interpretation of five real-world datasets. In future work, we plan to consider integrating user feedback into the *Tab2KG* pipeline to support an extension of the domain ontology in cases where tabular data contains previously unseen relations.

The representation of data tables as data graphs opens up several possibilities for making the lives of data scientists easier – with benefits including increased efficiency and robustness of data analytics workflows. Automating semantic table interpretation, *Tab2KG* takes an essential step in assisting domain experts and adds an important layer of abstraction to DAWs.

9.1. Limitations

We identified few limitations of *Tab2KG*, which can be attributed to the idea of using semantic lightweight dataset profiles, without requiring knowledge about particular data instances.

9.1.1. Column mapping without knowing the dataset instances

In contrast to approaches that perform semantic table interpretation at the instance level, i.e., with the help of the instance lookup in the domain knowledge graph, *Tab2KG* derives column mappings from statistical features in the domain profile. We have identified two limitations to this approach:

– Cyclic class relations: Currently, we do not address cyclic relations in the domain ontology, as for example (`dbo:`[16]`Event dbo:nextEvent dbo:Event`). Consider Fig. 12, where the second column provides the follow-up event of the event in the first column. Even if *Tab2KG* identifies the correct mapping for the third column to the property `dbo:locationCity`, we cannot tell if the third column maps to the location of the entity in the first or the second column.
– Class relations connecting the same classes: We do not have a decision criterion for distinguishing between class relations that connect the same subject and object classes. For example, consider the two object properties `dbo:leader` and `dbo:viceLeader` mapped to the second column in Fig. 13, both connecting countries to politicians. Only via statistical features extracted from the data table (which may only include the politician's name) it does not appear possible to decide if Kamala Harris is the president or the vice president of the United States.

```
Olympics 2004  ↠ Olympics 2008  ↠ Athens
Olympics 2012  ↠ Olympics 2016  ↠ London
```

Fig. 12. Cyclic class relation: did the London olympic games happen in 2012 or in 2016?

```
USA     ↠ Kamala Harris
Russia  ↠ Michail Mischustin
```

Fig. 13. Class relations connecting the same classes: is Kamala Harris the president or the vice president of the US?

[16] http://dbpedia.org/ontology/

9.1.2. Correlations between columns and data type relations

Our data table profiles consist of column profiles, i.e., the features of the single columns are computed in isolation (the same applies to domain profiles and data type relations). Such column profiles can be efficiently computed and added to the dataset profile. However, the dependencies between columns may hold additional knowledge. Consider the running example in Fig. 1 where the time in the second column (begin time) does always precede the time in the next column (end time), i.e., there is a correlation between the values in these two columns.

We have decided against the inclusion of correlation features into the domain and data table profiles because of the following reasons: First, correlations are often implicitly captured by the column profiles (e.g., in Fig. 1, the second column's mean value is less than the third column's mean value). Second, the variety of data types requires different correlation and dependency measures that are hard to compare. Third, we observed that the number of column pairs heavily exceeds the number of potentially meaningful correlations in our datasets. For example, consider the football dataset where the length of the first names may be compared to the length of last names, team names, the number of goals, ..., potentially leading to correlations by chance. Fourth, the computation and semantic representation of all possible column combinations are impractical due to the quadratic number of pairwise comparisons.

We also perform data type identification for the columns in isolation. The combination of columns could be used to infer richer data types (e.g., when a date is split into a year, month and day column). We leave such potential optimisations where columns are not just considered in isolation for future work.

9.1.3. Asymmetry between domain profiles and data table profiles

The domain profile and the data table profile can vary, even though they represent the same knowledge. Consider our running example of weather observations. In the table shown in Fig. 1, three rows refer to the sensor labeled "S1" but only two rows refer to the other sensors. When modeled as a knowledge graph following the mapping shown in Fig. 4, each sensor is modeled precisely as one node in the knowledge graph. Consequently, the statistical characteristics related to the sensors vary between the data table profile and the domain profile.

9.1.4. Availability and representativeness of the domain profile

Tab2KG relies on the availability of domain-specific data, i.e., a domain knowledge graph. In a recent survey, Abu-Salih provides an overview of approaches to create domain knowledge graphs [2]. The profile-based approach of *Tab2KG* specifically targets domains of interest which are concise and narrower in scope than existing cross-domain ontologies. As demonstrated in the evaluation, the semantic table interpretation performance decreases when the number of classes and relations increases.

The domain knowledge graph represents available knowledge regarding the domain of interest. We cannot directly measure the representativeness of the knowledge graph. Due to the open-world assumption and, consequently, knowledge graph incompleteness [4], we do not expect the domain knowledge graph to cover the domain knowledge in its entirety. Semantic table interpretation based on a knowledge graph, or its profile, will work for the tables whose content has been adequately represented by the domain knowledge graph, i.e. the data in the domain knowledge graph is representative of the content of these tables. In that case, the domain knowledge and the data table to be interpreted reflect similar value distributions. If this is not the case, misinterpretation can occur. For example, a data profile solely built from weather observations collected at night is not expected to generalise to daytime observations. In this case, the statistical features of time-related data type relations follow a distribution that is not representative of the whole domain, and thus the profiles will most likely not be matched.

9.2. Lightweight semantic dataset profiles

Lightweight semantic profiles generated by *Tab2KG* can be utilized as a compact domain and dataset representation to complement and enrich existing dataset catalogs. Such profiles can be generated automatically from the existing datasets and described using the DCAT[17] and the SEAS[18] vocabularies to facilitate their reusability. We believe that lightweight semantic profiles presented in this article are an essential contribution that can benefit a wide range of semantic applications beyond semantic table interpretation.

[17] https://www.w3.org/TR/vocab-dcat-2/

[18] https://ci.mines-stetienne.fr/seas/index.html

Differences in value representations such as value formatting and data types, the representativeness of the domain profile, as well as the dataset characteristics (number of columns in the data table as well as the number of classes and relations in the domain ontology) are the decisive factors to perform semantic table interpretation without instance-lookup and with lightweight profiles.

The profiles of *Tab2KG* are extensible. For example, header values, language-specific information or value embeddings can be used as a source of additional profile features in future work.

10. Related work

This section provides an overview of related approaches in the areas of dataset profiling and semantic table interpretation.

Given the growth of data available on the Web and in industrial data lakes, there is a high demand for dataset profiling, e.g., for creating data catalogs [36]. The profile features typically belong to several categories, including statistical observations at the instance and schema level [1,15]. Such features are not restricted to the initially defined schemas. For example, Neumaier et al. demonstrate how user interaction and search functionalities profit from the inclusion of spatio-temporal features into a dataset profile [35]. For tabular data, other approaches for dataset profile enrichment include the generation of table titles [18] and schema labels [10]. The inferred relation-specific rules and observations can further verify the data quality and become part of dataset profiles [45,46].

Recently, approaches to annotate tabular data with concepts from a knowledge base to predict column types gained increased attention. In the following, we introduce approaches for semantic table interpretation and the methods they use.

Instance-level lookup Most semantic table interpretation tools require access to a target knowledge graph, as they link data table cells to its resources [42]. Such approaches on the instance-level have recently been driven by the SemTab Challenge [26,27], which explicitly postulates a cell-entity annotation (CEA) task, where labels in data table cells are linked to entities in a target knowledge graph. The subsequent steps of column-type annotation (CTA) and columns-property annotation typically build upon the CEA results.

Several approaches are based on entity lookup (ColNet [8], MantisTable [11], LinkingPark [9], DAGOBAH [7,24], MTab [37], T2KMatch [43], CSV2KG [47], TableMiner+ [52], and the work by Zhang et al. [51]), with different (combined) query strategies, including URL matching [47], (partial) string lookup [9,11,24], string similarity [24,37,43,51], spelling correction [9] or the use of named entity linking tools [11]. After linking data table cells to resources in the knowledge graphs, the CTA is typically decided through voting or counting [8,9,37], ranking [11,47,52] or clustering [24]. ColNet and TableMiner+ apply learning strategies to reduce the number of lookup tasks required for detecting the class of the entities represented by a column. MantisTable and CSV2KG utilize concept graphs in their ranking to identify the most-specific sub classes. Also, identifying properties represented by the data tables typically relies on the CEA and additional knowledge graph lookups. For example, MTab does pairwise queries between entities identified in different columns to identify potential entity relations. To identify literal relations, MTab and TableMiner+ row-by-row compute data-type specific similarities between the literals in the target knowledge graph and the cell values. CSV2KG also involves the target ontology in this step. Sherlock [23] is a system that performs CTA and does not rely on CEA. However, it extracts column features for training a neural network, which is solely trained on DBpedia and explicitly predicts one of the selected DBpedia classes.

The reliance on entity linking with the target knowledge graph makes these approaches unsuitable in settings where data tables only contain previously unseen data, which is a common issue [44]. Even if the data instances in the data table are not entirely unknown, these approaches do not perform well when the number of matching entities drops [8]. Another thing these approaches have in common is the reliance on a large underlying knowledge base such as DBpedia and stable lookup services. In contrast, *Tab2KG* does not require access to the target knowledge graph after the domain profile has been created.

Subject column detection Several approaches [11,52] for semantic table interpretation assume the existence of a subject column, i.e., the main column of the data table where every other column is directly connected to. The subject column detection is typically identified through a set of statistic features. Approaches relying on a subject column do not consider the involvement of any classes which are not directly represented in the data table (for example, consider Fig. 1, but without the first column). *Tab2KG* utilizes a graph-based approach where such class relations can be found.

Column titles Some data tables come with column titles, which may indicate respective classes or properties. Efthymiou et al. [14] propose a method based on ontology matching, where one column title defines the class label, and other column titles represent property labels. Domain-independent Semantic Labeler [39], DAGOBAH [24] and TableMiner+ [52] exploit column titles as one of their features. *Tab2KG* does not require any column titles. This way, we ensure the generalizability for data tables without headers and language-independence.

Data type restrictions Data tables contain data of various types, and thus there are approaches specific to some of them. For example, EmbNum+ [38] transforms data table columns with numeric values into embedding vectors. Alobaid et al. demonstrate that using more fine-grained numeric data types increases semantic table interpretation performance for numeric column values [3]. For the interpretation of cross-lingual textual values, Luo et al. propose using several translation tools [33]. *Tab2KG* aims at the interpretation of data tables as a whole without restricting to particular data types or languages and thus establishes profiles that do not depend on particular data types or languages.

User feedback Instead of relying on fully-automated approaches for semantic table interpretation, which may be error-prone due to the challenges involved in this task, manual or semi-automated approaches rely on user feedback. Karma [29], Odalic [28], and ASIA [12] are interactive tools that let users decide on the correctness of suggested table annotations and thus achieve high precision, but demand both time and expertise from the user. *Tab2KG* is a fully-automated approach for semantic table interpretation that does not require user interventions.

Domain-independent semantic table interpretation Domain-independent approaches are not restricted to specific target knowledge graphs. Instead, they learn domain-independent similarity features to generate the mapping. The SemanticTyper [41] scores similarity between columns and data type relations based on handcrafted features for numeric and textual values. Based on similar features, the Domain-independent Semantic Labeler [39] adopts machine learning and handcrafted features to predict the similarity between a column and a class in the domain knowledge graph. Taheriyan et al. [49] generate a ranked list of potential column mappings learned from a sample of the domain ontology, which is then presented to the user. While both approaches are flexible concerning the target domain, *Tab2KG* aims to use only features present in the dataset profiles.

11. Conclusion

This article presented *Tab2KG* – an approach for semantic table interpretation based on lightweight semantic profiles. *Tab2KG* relies on domain profiles that enrich the relations in a domain ontology and serve as a semantic domain representation. *Tab2KG* matches these profiles with the tabular data profiles using one-shot learning approach. Our evaluation on five real-world datasets shows that our approach outperforms the baselines for semantic table interpretation. In future work, we plan to consider integrating user feedback into the *Tab2KG* pipeline to support an extension of the domain ontology in cases where tabular data contains previously unseen relations.

Acknowledgements

This work is partially funded by the DFG, German Research Foundation ("WorldKG", 424985896), the Federal Ministry of Education and Research (BMBF), Germany ("Simple-ML", 01IS18054) and the European Commission (EU H2020, "smashHit", 871477).

Appendix. Running example: RML definitions

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix ex: <http://example.com/resource/>.
@prefix csvw: <http://www.w3.org/ns/csvw#>.

ex:File a rml:source ;
 rml:source ex:FileSource ;
 rml:referenceFormulation <http://semweb.mmlab.be/ns/ql#CSV> .

ex:FileSource a csvw:Table;
 csvw:url "sky_sensors.tsv" ;
 csvw:dialect [
  a csvw:Dialect;
  csvw:delimiter "↠";
 ] .

ex:Mapping0 a rr:TriplesMap ;
 rml:logicalSource ex:File ;
 rr:subjectMap [
  rr:class sosa:Sensor ;
  rr:template "https://www.w3.org/TR/vocab-ssn/Sensor{col3}" ;
 ];

 rr:predicateObjectMap [
  rr:predicate rdfs:label ;
  rr:objectMap [
   rml:reference "col3";
  ]
 ] ;

 rr:predicateObjectMap [
  rr:predicate sosa:madeObservation ;
  rr:objectMap [
   rr:template "https://www.w3.org/TR/vocab-ssn/Observation{rowNumber}";
  ]
 ] .

ex:Mapping1 a rr:TriplesMap ;
 rml:logicalSource ex:File ;
 rr:subjectMap [
  rr:class sosa:Observation ;
  rr:template "https://www.w3.org/TR/vocab-ssn/Observation{rowNumber}" ;
 ] .
```

Listing 1. A working example of an RML file transforming the data table given in Fig. 1 (“sky_sensors.tsv”) into the data graph indicated in Fig. 4. To perform the transformation, the table needs to be pre-processed: the column titles “col0”,...,“col3” were added, and a “rowNumber” column. For brevity, we skip the mapping of the second and third column to `time:interval`

```
<https://www.w3.org/TR/vocab-ssn/Observation0>
  a <http://www.w3.org/ns/sosa/Observation> .

<https://www.w3.org/TR/vocab-ssn/Observation1>
  a <http://www.w3.org/ns/sosa/Observation> .

<https://www.w3.org/TR/vocab-ssn/Observation2>
  a <http://www.w3.org/ns/sosa/Observation> .

<https://www.w3.org/TR/vocab-ssn/SensorS2>
  a <http://www.w3.org/ns/sosa/Sensor>;
  <http://www.w3.org/2000/01/rdf-schema#label> "S2";
  <http://www.w3.org/ns/sosa/madeObservation>
    <https://www.w3.org/TR/vocab-ssn/Observation0> .

<https://www.w3.org/TR/vocab-ssn/SensorS3>
  a <http://www.w3.org/ns/sosa/Sensor>;
  <http://www.w3.org/2000/01/rdf-schema#label> "S3";
  <http://www.w3.org/ns/sosa/madeObservation>
      <https://www.w3.org/TR/vocab-ssn/Observation1>,
      <https://www.w3.org/TR/vocab-ssn/Observation2> .
```

Listing 2. The turtle file resulting from running the RML file in Listing 1 to transform the table given in Fig. 1 into a data graph. For brevity, we only consider the first three lines of the data table

References

[1] Z. Abedjan, L. Golab and F. Naumann, Profiling relational data: A survey, *VLDB J.* **24**(4) (2015), 557–581. doi:10.1007/s00778-015-0389-y.

[2] B. Abu-Salih, Domain-specific knowledge graphs: A survey, *Journal of Network and Computer Applications* **185** (2021), 103076, https://www.sciencedirect.com/science/article/pii/S1084804521000990. doi:10.1016/j.jnca.2021.103076.

[3] A. Alobaid, E. Kacprzak and Ó. Corcho, Typology-based semantic labeling of numeric tabular data, *Semantic Web* **12**(1) (2021), 5–20. doi:10.3233/SW-200397.

[4] H. Arnaout, S. Razniewski, G. Weikum and J.Z. Pan, Negative knowledge for open-world Wikidata, in: *Companion of the Web Conference 2021, Virtual Event*, Ljubljana, Slovenia, April 19–23, 2021, 2021, pp. 544–551. doi:10.1145/3442442.3452339.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z.G. Ives, DBpedia: A nucleus for a web of open data, in: *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, Busan, Korea, November 11–15, 2007, Lecture Notes in Computer Science, Vol. 4825, Springer, 2007, pp. 722–735. doi:10.1007/978-3-540-76298-0_52.

[6] M.J. Cafarella, A.Y. Halevy, H. Lee, J. Madhavan, C. Yu, D.Z. Wang and E. Wu, Ten years of WebTables, *Proc. VLDB Endow.* **11**(12) (2018), 2140–2149. doi:10.14778/3229863.3240492.

[7] Y. Chabot, T. Labbé, J. Liu and R. Troncy, DAGOBAH: An end-to-end context-free tabular data semantic, *Annotation System* **2553** (2019), 41–48, http://ceur-ws.org/Vol-2553/paper6.pdf.

[8] J. Chen, E. Jiménez-Ruiz, I. Horrocks and C. Sutton, ColNet: Embedding the semantics of web tables for column type prediction, in: *Proceedings of the Thirty-Third Conference on Artificial Intelligence, AAAI 2019*, AAAI Press, 2019, pp. 29–36. doi:10.1609/aaai.v33i01.330129.

[9] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J. Yao, J. Williams, A. Gordon and C. Lin, LinkingPark: An integrated approach for semantic table, *Interpretation* **2775** (2020), 65–74, http://ceur-ws.org/Vol-2775/paper7.pdf.

[10] Z. Chen, H. Jia, J. Heflin and B.D. Davison, Generating schema labels through dataset content analysis, in: *Companion of the Web Conference 2018 on the Web Conference 2018, WWW 2018*, Lyon, France, April 23–27, 2018, ACM, 2018, pp. 1515–1522. doi:10.1145/3184558.3191601.

[11] M. Cremaschi, F.D. Paoli, A. Rula and B. Spahiu, A fully automated approach to a complete semantic table interpretation, *Future Gener. Comput. Syst.* **112** (2020), 478–500. doi:10.1016/j.future.2020.05.019.

[12] V. Cutrona, M. Ciavotta, F.D. Paoli and M. Palmonari, Asia: A tool for assisted semantic interpretation and annotation of tabular, *Data* **2456** (2019), 209–212, http://ceur-ws.org/Vol-2456/paper54.pdf.

[13] A. Dimou, M.V. Sande, P. Colpaert, R. Verborgh, E. Mannens and R.V. de Walle, RML: A generic language for integrated RDF mappings of heterogeneous data, in: *Proceedings of the Workshop on Linked Data on the Web Co-Located with the 23rd International World Wide Web Conference (WWW 2014)*, Seoul, Korea, April 8, 2014, CEUR Workshop Proceedings, Vol. 1184, CEUR-WS.org, 2014. http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[14] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro and V. Christophides, Matching web tables with knowledge base entities: From entity lookups to entity embeddings, in: *Proceedings, Part I, The Semantic Web – ISWC 2017–16th International Semantic Web Conference*, Vienna, Austria, October 21–25, 2017, Lecture Notes in Computer Science, Vol. 10587, Springer, 2017, pp. 260–277. doi:10.1007/978-3-319-68288-4_16.

[15] M.B. Ellefi, Z. Bellahsene, J.G. Breslin, E. Demidova, S. Dietze, J. Szymanski and K. Todorov, RDF dataset profiling – a survey of features, methods, vocabularies and applications, *Semantic Web* **9**(5) (2018), 677–705. doi:10.3233/SW-180294.

[16] M. Garda, A semantics-enabled approach for data lake exploration services, in: *2019 IEEE World Congress on Services, SERVICES 2019*, Milan, Italy, July 8–13, 2019, IEEE, 2019, pp. 327–330. doi:10.1109/SERVICES.2019.00091.

[17] S. Gottschalk, N. Tempelmeier, G. Kniesel, V. Iosifidis, B. Fetahu and E. Demidova, Simple-ML: Towards a framework for semantic data analytics workflows, in: *Semantic Systems. The Power of AI and Knowledge Graphs – 15th International Conference, SEMANTiCS 2019*, Karlsruhe, Germany, September 9–12, 2019, Proceedings, Lecture Notes in Computer Science, Vol. 11702, Springer, 2019, pp. 359–366. doi:10.1007/978-3-030-33220-4_26.

[18] B. Hancock, H. Lee and C. Yu, Generating titles for web tables, in: *Proceedings of the World Wide Web Conference, TheWebConf 2019*, San Francisco, CA, USA, May 13–17, 2019, ACM, 2019, pp. 638–647. doi:10.1145/3308558.3313399.

[19] C. Hartenfels, M. Leinberger, R. Lämmel and S. Staab, Type-safe programming with OWL in Semantics4J, in: *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks Co-Located with 16th International Semantic Web Conference (ISWC 2017)*, Vienna, Austria, October 23rd–25th, 2017, CEUR Workshop Proceedings, Vol. 1963, CEUR-WS.org, 2017, http://ceur-ws.org/Vol-1963/paper549.pdf.

[20] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K. Sattler and J. Umbrich, Data summaries for on-demand queries over linked data, in: *Proceedings of the 19th International Conference on World Wide Web, WWW 2010*, ACM, Raleigh, North Carolina, USA, April 26–30, 2010, 2010, pp. 411–420. doi:10.1145/1772690.1772733.

[21] A. Heise, J. Quiané-Ruiz, Z. Abedjan, A. Jentzsch and F. Naumann, Scalable discovery of unique column combinations, *Proc. VLDB Endow.* **7**(4) (2013), 301–312, http://www.vldb.org/pvldb/vol7/p301-heise.pdf. doi:10.14778/2732240.2732248.

[22] S. Hsiao, D. Kao, Z. Liu and R. Tso, Malware image classification using one-shot learning with, *Siamese Networks* **159** (2019), 1863–1871. doi:10.1016/j.procs.2019.09.358.

[23] M. Hulsebos, K.Z. Hu, M.A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp and C.A. Hidalgo, Sherlock: A deep learning approach to semantic data type detection, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, Anchorage, AK, USA, August 4–8, 2019, ACM, 2019, pp. 1500–1508. doi:10.1145/3292500.3330993.

[24] V. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin and R. Troncy, DAGOBAH: Enhanced scoring algorithms for scalable annotations of tabular, *Data* **2775** (2020), 27–39, http://ceur-ws.org/Vol-2775/paper3.pdf.

[25] Information Technology – Database Languages – SQL Multimedia and Application Packages – Part 3: Spatial, Standard, International Organization for Standardization, 2016. https://www.iso.org/standard/60343.html.

[26] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen and K. Srinivas, in: *SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems*, Vol. 12123, 2020, pp. 514–530, https://doi.org/10.1007/978-3-030-49461-2_30. doi:10.1007/978-3-030-49461-2_30.

[27] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas and V. Cutrona, in: *Results of SemTab 2020, in: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020) Co-Located with the 19th International Semantic Web Conference (ISWC 2020), Virtual Conference (Originally Planned to Be in Athens, Greece)*, November 5, 2020, CEUR Workshop Proceedings, Vol. 2775, CEUR-WS.org, 2020, pp. 1–8, http://ceur-ws.org/Vol-2775/paper0.pdf.

[28] T. Knap, Towards odalic, a semantic table interpretation tool in the ADEQUATe project, in: *Proceedings of the 5th International Workshop on Linked Data for Information Extraction Co-Located with the 16th International Semantic Web Conference (ISWC 2017)*, Vienna, Austria, October 22, 2017, CEUR Workshop Proceedings, Vol. 1946, CEUR-WS.org, 2017, pp. 26–37, http://ceur-ws.org/Vol-1946/paper-04.pdf.

[29] C.A. Knoblock, P.A. Szekely, J.L. Ambite, A. Goel, S. Gupta, K. Lerman, M. Muslea, M. Taheriyan and P. Mallick, Semi-automatically mapping structured sources into the semantic web, in: *The Semantic Web: Research and Applications – 9th Extended Semantic Web Conference, ESWC 2012*, Heraklion, Crete, Greece, May 27–31, 2012, Proceedings, Lecture Notes in Computer Science, Vol. 7295, Springer, 2012, pp. 375–390. doi:10.1007/978-3-642-30284-8_32.

[30] G. Koch, R. Zemel and R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: *Proceedings of the Deep Learning Workshop, International Conference on Machine Learning'15*, Vol. 2, 2015.

[31] F. Lécué, On the role of knowledge graphs in explainable AI, *Semantic Web* **11**(1) (2020), 41–51, https://doi.org/10.3233/SW-190374. doi:10.3233/SW-190374.

[32] J. Lehmann, G. Sejdiu, L. Bühmann, P. Westphal, C. Stadler, I. Ermilov, S. Bin, N. Chakraborty, M. Saleem, A.N. Ngomo and H. Jabeen, Distributed semantic analytics using the SANSA stack, in: *Proceedings, Part II, The Semantic Web – ISWC 2017–16th International Semantic Web Conference*, Vienna, Austria, October 21–25, 2017, Lecture Notes in Computer Science, Vol. 10588, Springer, 2017, pp. 147–155. doi:10.1007/978-3-319-68204-4_15.

[33] X. Luo, K. Luo, X. Chen and K.Q. Zhu, Cross-lingual entity linking for web tables, in: *Proceedings of the Thirty-Second Conference on Artificial Intelligence (AAAI)*, AAAI Press, 2018, pp. 362–369.

[34] J. Mitlöhner, S. Neumaier, J. Umbrich and A. Polleres, Characteristics of open data CSV files, in: *Proceedings of the 2nd International Conference on Open and Big Data, OBD 2016*, Vienna, Austria, August 22–24, 2016, IEEE Computer Society, 2016, pp. 72–79. doi:10.1109/OBD.2016.18.

[35] S. Neumaier and A. Polleres, Enabling spatio-temporal search in open data, *J. Web Semant.* **55** (2019), 21–36. doi:10.1016/j.websem.2018.12.007.

[36] S. Neumaier, J. Umbrich and A. Polleres, Automated quality assessment of metadata across open data portals, *ACM J. Data Inf. Qual.* **8**(1) (2016), 2:1–2:29, https://doi.org/10.1145/2964909. doi:10.1145/2964909.

[37] P. Nguyen, N. Kertkeidkachorn, R. Ichise and H. Takeda, MTab: Matching tabular data to knowledge graph using probability models, in: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference, SemTab@ISWC 2019*, CEUR Workshop Proceedings, Auckland, New Zealand, October 30, 2019, Vol. 2553, CEUR-WS.org, 2019, pp. 7–14. http://ceur-ws.org/Vol-2553/paper2.pdf.

[38] P. Nguyen, K. Nguyen, R. Ichise and H. Takeda, EmbNum+: Effective, efficient, and robust semantic labeling for numerical values, *New Gener. Comput.* **37**(4) (2019), 393–427, https://doi.org/10.1007/s00354-019-00076-w. doi:10.1007/s00354-019-00076-w.

[39] M. Pham, S. Alse, C.A. Knoblock and P.A. Szekely, Semantic labeling: A domain-independent approach, in: *Proceedings, Part I, The Semantic Web – ISWC 2016–15th International Semantic Web Conference*, Kobe, Japan, October 17–21, 2016, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 446–462. doi:10.1007/978-3-319-46523-4_27.

[40] A. Pomp, A. Paulus, A. Kirmse, V. Kraus and T. Meisen, Applying semantics to reduce the time to analytics within complex heterogeneous infrastructures, *Technologies* **6**(3) (2018), 86. doi:10.3390/technologies6030086.

[41] S.K. Ramnandan, A. Mittal, C.A. Knoblock and P.A. Szekely, Assigning semantic labels to data sources, in: *The Semantic Web. Latest Advances and New Domains – 12th European Semantic Web Conference, ESWC 2015*, Portoroz, Slovenia, May 31–June 4, 2015, Proceedings, Lecture Notes in Computer Science, Vol. 9088, Springer, 2015, pp. 403–417. doi:10.1007/978-3-319-18818-8_25.

[42] D. Ritze and C. Bizer, Matching web tables to DBpedia – a feature utility study, in: *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017*, Venice, Italy, March 21–24, 2017, OpenProceedings.org, 2017, pp. 210–221. doi:10.5441/002/edbt.2017.20.

[43] 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, WIMS 2015*, Larnaca, Cyprus, July 13–15, 2015, ACM, 2015, pp. 10:1–10:6. doi:10.1145/2797115.2797118.

[44] D. Ritze, O. Lehmberg, Y. Oulabi and C. Bizer, 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, April 11–15, 2016, ACM, 2016, pp. 251–261. doi:10.1145/2872427.2883017.

[45] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Bießmann and A. Grafberger, Automating large-scale data quality verification, *Proc. VLDB Endow.* **11**(12) (2018), 1781–1794. doi:10.14778/3229863.3229867.

[46] G. Sejdiu, A. Rula, J. Lehmann and H. Jabeen, A scalable framework for quality assessment of RDF datasets, in: *Proceedings, Part II, the Semantic Web – ISWC 2019 – 18th International Semantic Web Conference*, Auckland, New Zealand, October 26–30, 2019, Lecture Notes in Computer Science, Vol. 11779, Springer, 2019, pp. 261–276. doi:10.1007/978-3-030-30796-7_17.

[47] B. Steenwinckel, G. Vandewiele, F. De Turck and F. Ongenae, CSV2KG: Transforming tabular data into semantic knowledge, in: *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Co-Located with the 18th International Semantic Web Conference (ISWC 2019)*, 2019, http://ceur-ws.org/Vol-2553/paper5.pdf.

[48] M. Taheriyan, C.A. Knoblock, P.A. Szekely and J.L. Ambite, Leveraging linked data to discover semantic relations within data sources, in: *Proceedings, Part I, The Semantic Web – ISWC 2016–15th International Semantic Web Conference*, Kobe, Japan, October 17–21, 2016, Lecture Notes in Computer Science, Vol. 9981, 2016, pp. 549–565, https://doi.org/10.1007/978-3-319-46523-4_33. doi:10.1007/978-3-319-46523-4_33.

[49] M. Taheriyan, C.A. Knoblock, P.A. Szekely and J.L. Ambite, Learning the semantics of structured data sources, *J. Web Semant.* **37–38** (2016), 152–169. doi:10.1016/j.websem.2015.12.003.

[50] S. Zhang and K. Balog, Web table extraction, retrieval, and augmentation: A survey, *ACM Transactions on Intelligent Systems and Technology (TIST)* **11**(2) (2020), 1–35. doi:10.1145/3372117.

[51] S. Zhang, E. Meij, K. Balog and R. Reinanda, Novel entity discovery from web tables, in: *Proceedings of the Web Conference 2020 (TheWebConf)*, Taipei, Taiwan, April 20–24, 2020, 2020, ACM/IW3C2. pp. 1298–1308. doi:10.1145/3366423.3380205.

[52] Z. Zhang, Effective and efficient semantic table interpretation using TableMiner^{+}, *Semantic Web* **8**(6) (2017), 921–957. doi:10.3233/SW-160242.