A framework for automatic Knowledge Base generation from observation data sets

Agnese Pinto[1 [0000−0002−2502−7467]], Saverio Ieva[1 [0000−0001−8598−5504]], Arnaldo Tomasino[1 [0000−0002−6144−4647]], Giuseppe Loseto[2 [0000−0002−7995−8494]], Floriano Scioscia[1 [0000−0002−7859−9602]⋆], Michele Ruta[1 [0000−0003−2125−327X]], and Francesco De Feudis[1 [0009−0006−1218−9965]]

[1] Polytechnic University of Bari, Bari 70125, Italy
{name.surname}@poliba.it
[2] LUM University "Giuseppe Degennaro", Casamassima BA 70010, Italy
loseto@lum.it

Abstract. In the Semantic Web of Everything, observation data collected from sensors and devices disseminated in smart environments must be annotated in order to produce a Knowledge Base (KB) or Knowledge Graph (KG) which can be used subsequently for inference. Available approaches allow defining complex data models for mapping tabular data to KBs/KGs: while granting high flexibility, they can be difficult to use. This paper introduces a framework for automatic KB generation in Web Ontology Language (OWL) 2 from observation data sets. It aims at simplicity both in usage and in expressiveness of generated KBs, in order to enable reasoning with SWoE inference engines in pervasive and embedded devices. An illustrative example from a precision farming case study clarifies the approach and early performance results support its computational sustainability.

Keywords: Knowledge Representation · Web Ontology Language · Knowledge Graph Construction · Machine Learning

1 Introduction and motivation

The Semantic Web of Everything (SWoE) vision aims to bring interoperable technologies for knowledge representation and reasoning to all scales of computing, from the World Wide Web (WWW) to nanodevices with strict processing, memory, and energy constraints. Distributed infrastructures for data stream gathering, analysis and interpretation can greatly benefit from knowledge-based methods and techniques, as commonly adopted Machine Learning (ML) methods have optimal performance only for very large and well-curated datasets. Dataset transformation into a usable Knowledge Base (KB) or Knowledge Graph (KG) requires overcoming the *impedance mismatch* of the Resource Description Framework (RDF) [26] and Web Ontology Language (OWL) [22] data models with

⋆ Corresponding author Tel.: +39-080-596-3054; fax: +39-080-596-3410

respect to most data sources [20]. Data governance and curation are complex problems for the majority of real-world infrastructures [28] based on wireless sensor networks, embedded devices and wearables, which generate high-volume and high-velocity streams of noisy and uncertain data.

When starting from raw observation data sets, the currently prevalent approaches to the generation of a Knowledge Base (KB) or Knowledge Graph (KG) rely on cloud-based *Data Lakes*. Though flexible, this approach is too complex and cumbersome for SWoE scenarios, not only because it requires huge storage resources, but also because it prevents agents running on pervasive devices from discovering and detecting relevant information autonomously. A wide body of research concerns mapping tabular data to KBs and KGs, which can cover many practical use cases: several methods, systems, and evaluation benchmarks exist [18]. They allow flexible mapping definitions beyond the simple "entity-per-row" assumption, in order to meet the requirements of advanced data management applications. For this reason, they often exhibit a steep learning curve and require significant expertise with Semantic Web technologies. In many SWoE contexts, however, observations represent events gathered from heterogeneous independent sources, such as sensors, Internet of Things (IoT) devices and objects populating a smart environment. Every event –either periodic or triggered by a condition– is an individual entity, with a specific value for each one of its attributes. As data models are kept relatively simple and regular, these contexts can benefit from leaner data processing frameworks for KB/KG construction.

This paper introduces a framework for automatic generation of OWL 2 KBs from observation data sets. The method consists in three phases: (i) data preparation and modeling of an upper ontology including classes for each type of observation (*i.e.*, relevant event in the problem domain) and for each feature; (ii) automatic Terminological Box (TBox) generation; (iii) automatic Assertion Box (ABox) generation, creating an OWL individual for each observation instance. By keeping logical expressiveness relatively simple, the KB generation approach is amenable to event classification and detection problems based on semantic matchmaking, [25] which can be executed on pervasive devices by means of optimized reasoning engines [24]. The correctness of the approach has been validated in a precision farming case study, while preliminary performance tests have evaluated the sustainability of the proposal.

The remainder of the paper is as follows. Relevant related work is recalled in Section 2. Section 3 describes in detail the framework architecture and processing steps. Section 4 reports on an illustrative example taken from the precision farming case study, while performance results are in Section 5, before conclusion.

2 Related work

The *RDF Mapping Language* (RML) [11] has extended the *Relational database-to-RDF* (R2RML) [8] World Wide Web Consortium (W3C) recommendation for a declarative mapping language from tabular data to RDF. RML adds support for the integration of multiple data sources and for various structured data

formats in addition to relational databases. R2RML and RML are still among the most widely adopted approaches for KG construction. RML has been further extended by *RML-star* [10] in order to support the *RDF-star* [27] language for annotating RDF statements with other RDF statements. These approaches, however, require writing a configuration document manually to specify mapping definitions. *Morph-KGC* [3] is an R2RML and RML interpreter focusing on performance and scalability: it groups rules in the input mapping documents, in order to guarantee the generation of disjoint sets of RDF triples by each group.

Several systems use popular Linked Open Data sources to annotate tabular data automatically. The architecture of *JenTab* [1] consists in a pool of modular processing tasks, which are combined in different pipelines for each type of table semantic annotation problem; it uses *Wikidata* [30] for entity resolution. In addition to Wikidata, *DAGOBAH* [19] queries four other services, including DBpedia [17] and Wikipedia API. Machine learning combined with probabilistic [31] and constraint programming [9] methods have also been exploited to assign a semantic model to tabular data sources automatically.

Whereas the above approaches produce RDF output, systems targeting OWL include *Mapping Master* [21] and *BootOX* [16]. Mapping Master provides a language to define mappings of complex spreadsheets to OWL ontologies. BootOX interprets R2RML mappings by encoding relational database features to OWL 2 axioms: the three phases of the *bootstrapping* problem as formulated in BootOX –vocabulary and ontology generation, mapping generation, importing– are conceptually similar to the processing phases of the approach presented in this paper, although the latter focuses on data-driven applications.

3 Framework architecture

The main goal of the proposed framework is to automate the construction of a Knowledge Base starting from a reference dataset consisting of observation data. In this way, a dataset can be easily mapped into a conceptual model related to a specific domain through which every observation can be annotated. As shown in Figure 1, the process of KB generation consists of three sub-tasks: (a) data preparation and upper ontology modeling; (b) Terminological Box (TBox) generation; (c) Assertion Box (ABox) generation. The following paragraphs provide a top-down description of the approach.

3.1 Data preparation and upper ontology modeling

The first step towards the definition of the Knowledge Base is the dataset analysis, aiming to identify the following information:

- list of *features* of observations, denoted as $\mathcal{F} = \{F_1, F_2, \ldots, F_N\}$; each feature has a (numerical or categorical) domain;
- list of *tuples*, denoted as $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$. Each tuple R_i is a set of attribute values $\langle v_{i,1}, v_{i,2}, \ldots, v_{i,N} \rangle$ representing an observation collected for the specific scenario;

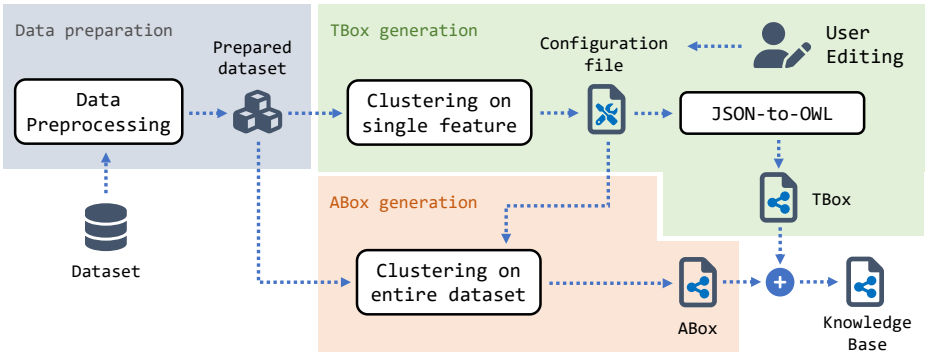

Fig. 1: Workflow of the proposed framework

– list of *events* of interest, denoted as $\mathcal{E} = \{E_1, E_2, \ldots, E_P\}$ and representing the set of class labels associated to the observations. Each tuple R_i can be associated to one or more events.

The upper layers of the reference ontology $\mathcal{T}$ should model the domain conceptualization along the specific patterns detailed hereinafter, in order to support semantic-based data annotation and interpretation. $\mathcal{T}$ is assumed as acyclic and expressed in the moderately complex $\mathcal{ALN}$ (Attributive Language with unqualified Number restrictions) Description Logic [4]. This is required to be compliant with nonstandard, nonmonotonic inferences provided by reasoning engines designed for SWoE applications (*e.g.*, *Tiny-ME* [24]). For each feature in $\mathcal{F}$, $\mathcal{T}$ must include a hierarchy of concepts derived from a reference class, selected by the user typically by referring to a well-known upper ontology, forming a partonomy of the topmost concept. For example, in Figure 2 the *FeatureOfInterest* class defined in the *SOSA (Sensor, Observation, Sample, and Actuator)* ontology [15] is used as an upper layer for the TBox section related to the features. This subphase must deal with any *impedance mismatch* between the dataset model and (the available expressiveness of) the selected OWL sublanguage [23]. In this way, each dataset attribute F_i is represented by means of a class/subclass taxonomy featuring all significant value ranges and configurations it can take in the domain of interest. The breadth of each sublevel will be determined automatically during the TBox generation task described in Section 3.2. In that step, each subclass $F_{i,j}$ will be also associated with contextual parameters by means of specific *OWL Annotation Properties*. Similarly, the events in $\mathcal{E}$ are modeled as subclasses of an output concept identified by the user (*e.g.*, the *Observation* class in Figure 2).

3.2 TBox generation

The next step of the framework consists in processing the prepared dataset to generate the TBox. This task is composed of two distinct sub-tasks:

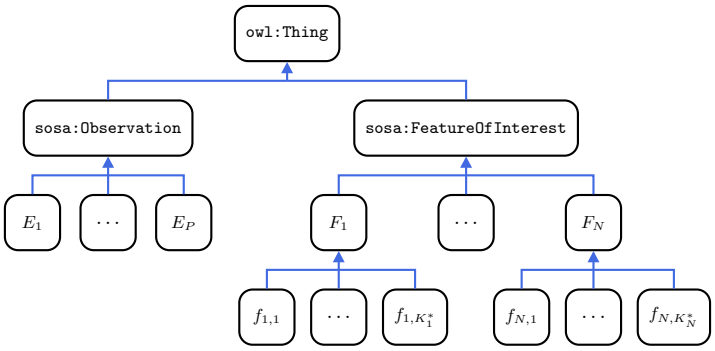

Fig. 2: TBox hierarchy

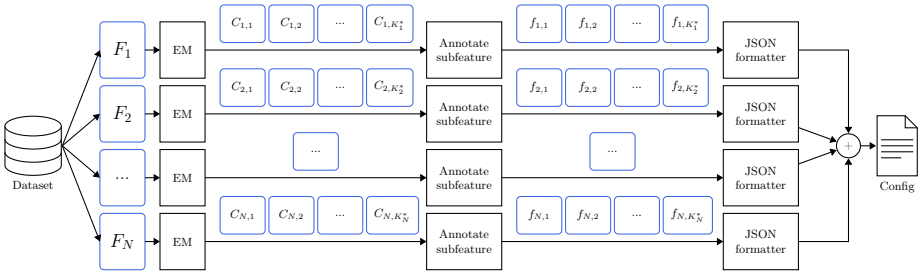

Fig. 3: Generation of the configuration file

1. generation of a configuration file, storing the ontology metadata and all features parameters required to create and characterize the concept hierarchy;
2. serialization of the TBox, according to one of the available OWL 2 syntaxes, guided by the configuration file.

The main advantage of this two-step approach lies in the fact that the user can customize the configuration file (*e.g.*, to include further contextual information) before proceeding with the generation of the TBox. In this way, the proposed workflow can be adapted and reused for the generation of ontologies in a wide variety of domains of interest. Given a properly prepared dataset as input, the process of building the configuration file is illustrated in Figure 3. The prepared dataset is processed via the *K-means* clustering algorithm, which has been chosen due to computational complexity amenable to SWoE contexts [2]. For each feature $F_i \in \mathcal{F}$, the *Elbow Method* (EM) [29] is used to compute the optimal number K_i^* of clusters to partition the values contained in the dataset for each i-th feature. Denoting as $\mathcal{C}_i = \{C_{i,1}, C_{i,2}, \ldots, C_{i,K_i^*}\}$ the set of clusters obtained from the feature F_i, for each cluster $C_{i,j} \in \mathcal{C}$ a new subclass $f_{i,j}$ (for $i = 1..N$ and $j = 1..K_i^*$) is added to the TBox. The name of each subclass of a given feature is obtained by combining a cluster *prefix* with the feature *name*, *e.g.*, *Low* + *Temperature* = *LowTemperature*). Up to 7 different prefixes are currently supported to partition a feature: *ExtremelyLow*, *VeryLow*, *Low*, *Medium*,

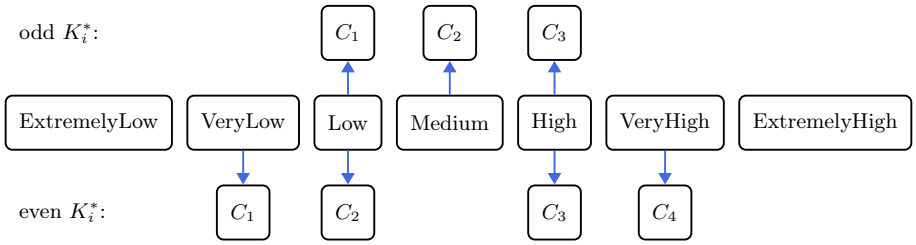

Fig. 4: Example of prefix assignment

Metadata	**Annotation Property IRI**	**Short IRI**	**Range**
Title	`http://purl.org/dc/terms/title`	`dcterms:title`	`xsd:string`
Description	`http://purl.org/dc/terms/description`	`dcterms:description`	`xsd:string`
Creator	`http://purl.org/dc/terms/creator`	`dcterms:creator`	`xsd:string`
Version	`http://www.w3.org/2002/07/owl#versionIRI`	`owl:versionIRI`	`xsd:string`
Centroid	`http://swot.sisinflab.poliba.it/onto/kbgen/centroid`	`kbg:centroid`	`xsd:float`
Min Value	`http://schema.org/minValue`	`schema:minValue`	`xsd:float`
Max Value	`http://schema.org/maxValue`	`schema:maxValue`	`xsd:float`

Table 1: List of supported OWL annotation properties

High, *VeryHigh*, *ExtremelyHigh*. As shown in Figure 4, if the number of clusters K_i^* associated to F_i is odd, then the prefix *Medium* will be assigned to the subclass $f_{i,m}$ with $m = (K_i^* + 1)/2$. On the contrary, if K_i^* is even then the prefix *Medium* will not be assigned to any subclass.

Each subclass $f_{i,j}$ is also characterized by the *centroid* ($c_{i,j}$) and the *range* of values associated to the $C_{i,j}$ cluster computed via K-means. The *kbg:centroid* annotation property associates the centroid to the subclass, while minimum and maximum values are specified as a pair of annotation properties –borrowed from the *Schema.org* vocabulary [12]– named *schema:minValue* and *schema:maxValue*, respectively. This facilitates associating the features of individual dataset observations to specific clusters and mapping each of them to the related subclass. The final configuration is then created by including the definition of all computed OWL classes and their annotated values. Basic ontology metadata (detailed in Table 1) is also specified by the user to further characterize the reference Knowledge Base. Finally, the configuration file is generated according to JSON (JavaScript Object Notation) syntax [5] representing a self-describing and easy-to-parse data format.

In the second step, domain experts can modify the configuration file both to refine results obtained by the clustering procedure and introduce additional elements (*e.g.*, classes, properties) not included in the original dataset. The refined configuration is parsed by a dedicated software module implemented in Java in order to generate the corresponding ontology. OWL API (version 3.4.10) library [13] is used as reference implementation providing several functionalities for creating, manipulating and serializing OWL ontologies.

3.3 ABox generation

The ABox generation process is shown in Figure 5. The same dataset used to generate the TBox is annotated to generate OWL individuals. In particular, an instance of the ABox can represent:

- a single tuple of the dataset which values are mapped to elements of the TBox. In this way, a generic data corpus can be translated to an OWL KB, where each record corresponds to an instance;
- an aggregate event description derived from a clustering procedure considering all attributes simultaneously rather than individual features.

In the latter case, K-means algorithm takes the whole dataset in input, in order to identify different partitions of collected observations into K clusters. Each cluster represents an OWL individual to be modeled within the KB with a reference class expression, which can be used to label new observations. The EM is used also in this case for defining the optimal number K^* of clusters to partition the whole dataset. Since the clustering was performed on a dataset with N attributes, each cluster $D_i \in \mathcal{D} = \{D_1, D_2, \ldots, D_{K^*}\}$ will be associated to a list of centroids $d_i = (d_{i,1}, d_{i,2}, \ldots, d_{i,N}) \in \mathbb{R}^N$, where the value $d_{i,j}$ represents the reference centroid for the feature $F_j \in \mathcal{F}$. In particular, let ϕ_j be the set of partitions $\{f_{j,1}, f_{j,2}, \ldots, f_{j,K_j^*}\}$ computed for the feature F_j; each value $d_{i,j} \in d_i$, $\forall j = 1..N$ is mapped to the corresponding subclass defined in the TBox associated with the partition $f_{j,h} \in \phi_j$ if and only if $minValue(f_{j,h}) < d_{i,j} \leq maxValue(f_{j,h})$. After mapping all values $d_{i,j} \in D_i$ into OWL classes defined in the TBox, an OWL named individual will be created as a conjunctive expression of ontology axioms. Two different modeling approaches have been investigated to represent the generated instances, which can be selected by means of the configuration file. In the first approach, each individual description is modeled as a simple conjunction of atomic concepts corresponding to the subclasses $f_{j,h}$. In the other one, individuals are modeled combining OWL object properties with each subclass. The latter approach preserves the hierarchy of concepts, a further explicit semantics is given and each ontology axiom is expressed by means of a universal and an existential quantifier. Finally, individuals are associated to a specific event selected among the output classes in the ontology. This process is applied for each cluster D_i to populate the ABox of the KB.

4 Illustrative example in precision farming

In this section, a simple example is described in order to highlight peculiarities of the proposed framework. The reference dataset derives from on-site measurements of several parameters characterizing wheat, olive and grapevine crops in the Apulia region, exploiting Internet of Things (IoT) device networks as well as hyperspectral images taken from Unmanned Aerial Vehicle (UAV), manned aircraft, and satellite constellations. Each crop is described by a set of attributes related to health status, growth factors and phenological phases. Quantitative

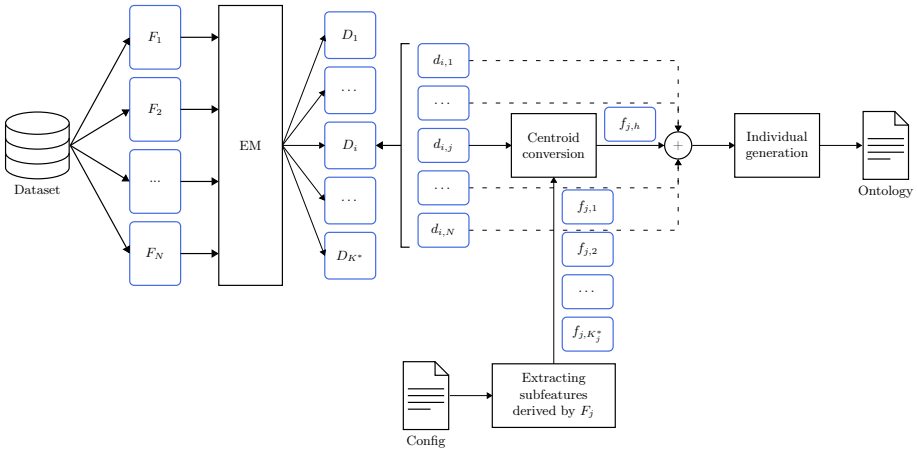

Fig. 5: ABox generation process

data are processed according to the workflow in Figure 1 and the generated KB is reported in what follows. For the sake of readability, OWL statements are expressed in *Manchester* syntax [14] but all OWL 2 serializations are supported and the reference syntax can be selected in the configuration file.

SAREF4AGRI[3] is used as upper ontology representing an extension of the *Smart Applications REFerence ontology* (SAREF) [7] for the agriculture and food domain. This ontology includes two concepts *saref:Property* and *s4agri:Crop* configured as reference class for the subtrees related to crop parameters and observed events, respectively.

```
Prefix: : <https://www.example.com/crop-onto#>
Prefix: saref: <https://saref.etsi.org/core/>
Prefix: s4agri: <https://saref.etsi.org/saref4agri/>
Class: saref:Property
Class: s4agri:Crop
```

The TBox generation task produces the following results. For each feature F_i within the dataset (*e.g.*, petiole height), a new OWL class is created as subclass of *saref:Property*. As mentioned in Section 3.2, F_i is also partitioned in multiple subclasses $f_{i,j}$ (*e.g.*, low petiole height), according to the clustering algorithm executed on the single attribute, and characterized by means of OWL annotation properties. An OWL object property is also generated as subproperty of *saref:hasProperty*.

```
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: schema: <https://schema.org/>
Prefix: kbg: <http://swot.sisinflab.poliba.it/onto/kbgen/>
Class: PetioleHeight
  SubClassOf: saref:Property
```

[3] https://saref.etsi.org/saref4agri/

```
Class: LowPetioleHeight
  SubClassOf: PetioleHeight
  Annotations: kbg:centroid "3.57"^^xsd:float,
    schema:minValue "3.1"^^xsd:float,
    schema:maxValue "3.8"^^xsd:float
ObjectProperty: hasPetioleHeight
  SubPropertyOf: saref:hasProperty
  Range: PetioleHeight
```

Three OWL classes are defined as subclasses of *Crop* to model the cultivations observed during the measurement procedure: *Grapevine*, *OliveTree* and *Wheat*.

```
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Class: GrapeVine
  SubClassOf: s4agri:Crop
Class: OliveTree
  SubClassOf: s4agri:Crop
Class: Wheat
  SubClassOf: s4agri:Crop
AnnotationProperty: cropType
  SubPropertyOf: rdf:value
  Range: s4agri:Crop
```

Finally, the ABox is generated as described in Section 3.3. The following example reports an OWL named individual representing an olive tree observation. The individual description is obtained by means of the K-means algorithm applied on the whole dataset. All numeric values in the cluster are mapped to concepts and properties defined during the previous step.

```
Individual: OliveTreeObservation
  Annotations: cropType OliveTree
  Types: (hasPetioleHeight only LowPetioleHeight) and (hasPetioleArea
    only MediumPetioleArea) and (hasTemperature only VeryLowTemperature)
    and (hasHumidity only HighHumidity) ... and (hasTreeDensity only
    HighTreeDensity)
```

5 Performance evaluation

As a preliminary feasibility assessment of the proposed approach, computational performance has been evaluated exploiting a reference dataset [6] consisting of 10000 observations, each characterized by 48 features (specifically, 22 Boolean, 3 continuous numeric, 16 discrete numeric and 7 categorial features). Tests have been carried out on a desktop PC equipped with Intel Core i7-4790 quad-core CPU at 3.6 GHz, 16 GB DDR3 RAM at 1600 MT/s, 2 TB SATA storage memory at 7200 RPM, Windows 10 Home 64-bit. Performed tests regard the turnaround time of the KB generation procedure, which is composed of the following subtasks: (a) dataset preprocessing; (b) features clustering; (c) configuration file creation; (d) TBox generation; (e) dataset clustering; (f) ABox generation (atomic

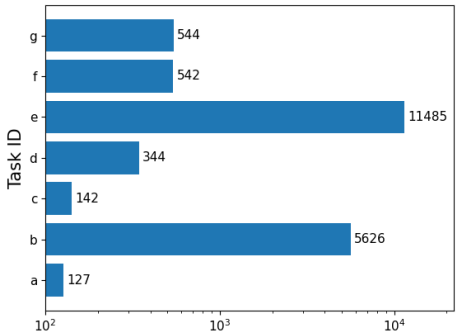

Fig. 6: Processing time (ms)

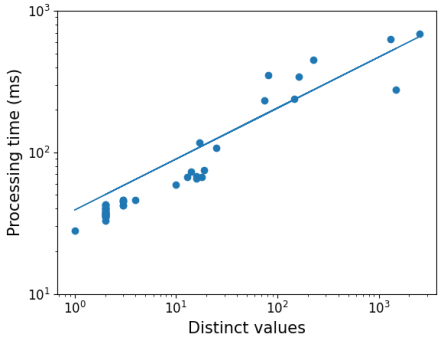

Fig. 7: Clustering time - Task *b*

concepts conjunction); (g) ABox generation (atomic concepts with object properties). Tests have been repeated five times and average values have been reported. As shown in Figure 6, steps (b) and (e) require on average a higher processing time due to the clustering procedures. In particular, the execution time of task (b) strongly correlates with on the number of distinct values collected for each feature (Figure 7). The two modeling approaches (f) and (g) for the generation of the ABox require similar processing time. Users can select the most suitable approach according to the tasks that will be performed on the generated output without worrying about particular performance issues.

6 Conclusion and future work

This paper has presented an approach for generation of OWL 2 KBs from observation data sets. The proposal aims to automate annotation of data gathered by sensing devices in pervasive computing contexts, fostering usage of semantic-enhanced machine learning in the SWoE. K-means clustering is exploited on each data set feature to create the TBox along partonomy patterns; K-means is applied to individuals as well, in order to model (i) clusters across all features as concept expressions, representing the reference model in a training set, and (ii) single records in a test set for evaluation. The correctness of the approach has been validated empirically in a case study on precision farming, while preliminary performance of a prototype implementation supports its computational feasibility.

Future work includes adapting the framework implementation to embedded, wearable and single-board computer platforms, followed by experimental evaluations of the computational efficiency and the quality of the generated KB w.r.t. state-of-the-art methods on available reference benchmarks. Moreover, the exploitation of generated KBs in a variety of real SWoE case studies will allow to fully validate the suitability of the approach. Further investigation includes integration with RML or RML-derived languages to provide both ease-of-use for typical applications and flexibility when needed.

Acknowledgments

This work has been supported by project TEBAKA (TErritorial BAsic Knowledge Acquisition), funded by the Italian Ministry of University and Research.

References

1. Abdelmageed, N., Schindler, S.: Jentab: Matching tabular data to knowledge graphs. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Workshop, International Semantic Web Conference. pp. 40–49 (2020)
2. Ahmed, M., Seraj, R., Islam, S.M.S.: The K-means algorithm: a comprehensive survey and performance evaluation. Electronics **9**(8), 1295 (2020)
3. Arenas-Guerrero, J., Chaves-Fraga, D., Toledo, J., Pérez, M.S., Corcho, O.: Morph-KGC: Scalable knowledge graph materialization with mapping partitions. Semantic Web pp. 1–20 (2022)
4. Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al.: The description logic handbook: Theory, implementation and applications. Cambridge university press (2003)
5. Bray, T.: RFC 8259: The JavaScript object notation (JSON) data interchange format (2017)
6. Chiew, K.L., Tan, C.L., Wong, K., Yong, K.S., Tiong, W.K.: A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Information Sciences **484**(C), 153–166 (may 2019)
7. Daniele, L., den Hartog, F., Roes, J.: Created in Close Interaction with the Industry: The Smart Appliances REFerence (SAREF) Ontology. In: Cuel, R., Young, R. (eds.) Formal Ontologies Meet Industry. pp. 100–112. Springer International Publishing, Cham (2015)
8. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. Recommendation, W3C (Sep 2012), `https://www.w3.org/TR/r2rml/`
9. De Una, D., Rümmele, N., Gange, G., Schachte, P., Stuckey, P.J.: Machine learning and constraint programming for relational-to-ontology schema mapping. In: International Joint Conference on Artificial Intelligence. pp. 1277–1283 (2018)
10. Delva, T., Arenas-Guerrero, J., Iglesias-Molina, A., Corcho, O., Chaves-Fraga, D., Dimou, A.: RML-star: A declarative mapping language for RDF-star generation. In: ISWC2021, the International Semantic Web Conference. vol. 2980. CEUR (2021)
11. Dimou, A., Vander Sande, M., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: A generic language for integrated rdf mappings of heterogeneous data. In: Workshop on Linked Data on the Web, 23rd International World Wide Web Conference. pp. 1–5 (2014)
12. Guha, R.V., Brickley, D., Macbeth, S.: Schema.Org: Evolution of Structured Data on the Web. Commun. ACM **59**(2), 44–51 (jan 2016)
13. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL Ontologies. Semantic Web **2**(1), 11–21 (2011)
14. Horridge, M., Patel-Schneider, P.: OWL 2 Web Ontology Language Manchester Syntax (Second Edition). W3C note, W3C (Dec 2012), http://www.w3.org/TR/owl2-manchester-syntax/
15. Janowicz, K., Haller, A., Cox, S.J., Le Phuoc, D., Lefrançois, M.: SOSA: A lightweight ontology for sensors, observations, samples, and actuators. Journal of Web Semantics **56**, 1–10 (2019)

16. Jiménez-Ruiz, E., Kharlamov, E., Zheleznyakov, D., Horrocks, I., Pinkel, C., Skjæveland, M.G., Thorstensen, E., Mora, J.: BootOX: Practical mapping of RDBs to OWL 2. In: Proceedingsof the 14th International Semantic Web Conference, Part II. pp. 113–132. Springer (2015)
17. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., Bizer, C.: Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. Semantic web **6**(2), 167–195 (2015)
18. Liu, J., Chabot, Y., Troncy, R., Huynh, V.P., Labbé, T., Monnin, P.: From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods. Journal of Web Semantics p. 100761 (2022)
19. Liu, J., Troncy, R.: Dagobah: an end-to-end context-free tabular data semantic annotation system. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Workshop, International Semantic Web Conference. pp. 41–48 (2019)
20. Modoni, G.E., Sacco, M.: Discovering Critical Factors Affecting RDF Stores Success. In: Pandey, R., Paprzycki, M., Srivastava, N., Bhalla, S., Wasielewska-Michniewska, K. (eds.) Semantic IoT: Theory and Applications: Interoperability, Provenance and Beyond, pp. 193–206. Springer (2021)
21. O'Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping master: a flexible approach for mapping spreadsheets to OWL. In: International Semantic Web Conference. pp. 194–208. Springer (2010)
22. Parsia, B., Rudolph, S., Krötzsch, M., Patel-Schneider, P., Hitzler, P.: OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation, W3C (Dec 2012), `http://www.w3.org/TR/owl2-primer`
23. Pinkel, C., Binnig, C., Jiménez-Ruiz, E., Kharlamov, E., May, W., Nikolov, A., Sasa Bastinos, A., Skjæveland, M.G., Solimando, A., Taheriyan, M., et al.: RODI: benchmarking relational-to-ontology mapping generation quality. Semantic Web **9**(1), 25–52 (2018)
24. Ruta, M., Scioscia, F., Bilenchi, I., Gramegna, F., Loseto, G., Ieva, S., Pinto, A.: A multiplatform reasoning engine for the Semantic Web of Everything. Journal of Web Semantics **73**, 100709 (2022)
25. Ruta, M., Scioscia, F., Loseto, G., Pinto, A., Di Sciascio, E.: Machine learning in the Internet of Things: A semantic-enhanced approach. Semantic Web **10**(1), 183–204 (2019)
26. Schreiber, G., Gandon, F.: RDF 1.1 XML syntax. Recommendation, W3C (Feb 2014), `http://www.w3.org/TR/rdf-syntax-grammar/`
27. Schreiber, G., Gandon, F.: RDF-star and SPARQL-star. Draft community group report, W3C (June 2023), `https://w3c.github.io/rdf-star/cg-spec/editors_draft.html`
28. Talburt, J.R., Ehrlinger, L., Magruder, J.: Automated data curation and data governance automation. Frontiers in Big Data **6** (2023)
29. Thorndike, R.: Who belongs in the family? Psychometrika **18**(4), 267–276 (1953)
30. Van Veen, T.: Wikidata: fron "an" identifier to "the" identifier. Information technology and libraries **38**(2), 72–81 (2019)
31. Vu, B., Knoblock, C., Pujara, J.: Learning semantic models of data sources using probabilistic graphical models. In: The World Wide Web Conference. pp. 1944–1953 (2019)