Datil: Learning Fuzzy Ontology Datatypes

Ignacio Huitzil, Umberto Straccia, Natalia Díaz-Rodríguez, Fernando Bobillo

▶ **To cite this version:**

Ignacio Huitzil, Umberto Straccia, Natalia Díaz-Rodríguez, Fernando Bobillo. Datil: Learning Fuzzy Ontology Datatypes. IPMU 2018: 17th Information Processing and Management of Uncertainty in Knowledge-Based Systems Conference., Jun 2018, Cádiz, Spain. hal-01951785

HAL Id: hal-01951785
https://hal.science/hal-01951785v1

Submitted on 11 Dec 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Datil: Learning Fuzzy Ontology Datatypes

Ignacio Huitzil[1], Umberto Straccia[2], Natalia Díaz-Rodríguez[3], and Fernando Bobillo[1]

[1] I3A, University of Zaragoza, Spain
[2] ISTI-CNR, Pisa, Italy
[3] U2IS, ENSTA ParisTech and INRIA FLOWERS, France
Email: ihuitzil@unizar.es, straccia@isti.cnr.it, natalia.diaz@ensta-paristech.fr, fbobillo@unizar.es

Abstract. Real-world applications using fuzzy ontologies are increasing in the last years, but the problem of fuzzy ontology learning has not received a lot of attention. While most of the previous approaches focus on the problem of learning fuzzy subclass axioms, we focus on learning fuzzy datatypes. In particular, we describe the *Datil* system, an implementation using unsupervised clustering algorithms to automatically obtain fuzzy datatypes from different input formats. We also illustrate the practical usefulness with an application: semantic lifestyle profiling.

Keywords: fuzzy ontologies, machine learning, lifestyle profiling

1 Introduction

Ontologies can nowadays be considered a standard for knowledge representation. An ontology is an explicit and formal specification of the concepts, individuals and relationships that exist in some area of interest, created by defining axioms that describe the properties of these entities [20]. Ontologies can provide semantics to data, making knowledge maintenance, information integration, and reuse of components easier. The current standard language for ontology representation is OWL 2 (Web Ontology Language) [24].

Classical ontologies are not appropriate to deal with imprecise and vague knowledge, inherent to several real world domains. Fuzzy ontologies [3, 21] extend classical ontologies with elements of fuzzy set theory and fuzzy logic [25]. Although fuzzy ontologies have been successfully used in several real-world applications [9–11, 13] and some methodologies [1] and tools [5] supporting their development are available, the cold start problem is still common. It is difficult for ontologists who are experts in a domain but not familiar with fuzzy logic to develop fuzzy ontologies. To overcome this problem, *fuzzy ontology learning techniques* are necessary, but unfortunately there is not been a lot of research in this direction.

The few exceptions are focused on learning fuzzy subclass axioms [4, 15–17, 22]; the three latter references are implemented in the *FuzzyDL-Learner* tool.[4]

[4] www.umbertostraccia.it/cs/software/FuzzyDL-Learner

Starting from the (possibly partial) membership of individuals to classes, it is possible to automatically compute some partial inclusions between (possibly fuzzy) concepts. Fuzzy concept descriptions can be based on fuzzy datatypes.

Because the focus is on learning fuzzy subclass axioms, these approaches usually restrict to a simple case: a uniform partition of the domain into fuzzy datatypes. Instead, we propose to compute fuzzy datatypes from existing real data using clustering algorithms.

In some cases, only attribute values are available and there are no data about the membership to classes. In such cases, we can still learn the fuzzy datatypes, use them to build some preliminary fuzzy subclass axioms to populate the classes, and learn a more complete set of fuzzy subclass axioms using existing approaches.

The main contribution of this paper is the description of the *Datil* system, an implementation of an automatic fuzzy datatype learning algorithm for fuzzy ontologies supporting different input and output formats. We also discuss how to integrate this learning step into existing approaches for fuzzy subclass learning and illustrate it with a real-world use case: semantic lifestyle profiling, i.e., the automatic classification of the lifestyle of people given their digital footprints. The ultimate aim is to help tasks such as long-term human behavior classification and thus improve virtual coaching or customize lifestyle recommendation and intervention programs from free form non-labelled sensor data.

The rest of this paper is organized as follows. Section 2 provides some background on fuzzy ontologies and clustering algorithms. Next, Section 3 describes the Datil tool. Then, Section 4 illustrates the usefulness of the system with a use case on semantic lifestyle profiling. Finally, Section 5 sets out some conclusions and addresses some ideas for future work.

2 Background

2.1 Fuzzy Ontologies

Fuzzy ontologies extend classical (crisp) ontologies by considering several notions of fuzzy set theory and fuzzy logic [3, 21]. Before going into the details, let us briefly recall the elements of an ontology:

- *Individuals* denote domain elements or objects. For example, john and mary.
- *Datatypes* denote elements that do not belong to the represented domain, but rather to a different domain that is already structured and whose structure is already known to the machine. Data values can be numerical values, textual, or dates, among many other possibilities.
- *Concepts* or classes denote unary predicates and are interpreted as sets of individuals, such as Human. Concept can be simple (atomic) or complex, built up using different types of concept constructors depending on the expressivity of the ontology language.
- *Properties* or roles denote binary predicates relating a pair of elements. There are two types of properties: *object properties* (or abstract roles) link a pair of individuals, whereas *data properties* (or concrete roles) relate an individual

with a data value. For instance, isFriendOf relates two human individuals and is an object property, while hasAge links an individual with an integer number and is a data property.

- *Axioms* are formal statements involving these ontology elements, like a recipe that defines how to combine the previous ingredients to represent the knowledge of some particular domain. The available types of axioms depend on the expressivity of the ontology language, but some typical types are:
  - *Concept assertions* state the membership of an individual to a class. For example, the fact that john belongs to the concept of Human people.
  - *Object property assertions* describe the relation between two individuals. For instance, one can state that john and mary are related via hasChild.
  - *Data property assertions* describe the relation between an individual and a data value. For example, it is possible to express that John's age is 18 by relating john and the number 18 via hasAge.
  - *Subclass* axioms, stating that a concept is more specific (a subclass) of another one. For example, Woman is more specific than Human.

The interested reader can find a complete list of the OWL 2 elements in [24].

In fuzzy ontologies, the elements of an ontology are extended in such a way that concepts, relations, datatypes, and axioms are fuzzy. In particular:

- *Fuzzy concepts* and *fuzzy properties* are interpreted as fuzzy sets of individuals and fuzzy binary relations, respectively. For example, YoungHuman can contain the fuzzy set of young people.
- *Fuzzy axioms* express statements that are not either true or false but hold to some degree. For example, we can state that john belongs to the concept of YoungHuman with at least degree 0.9, meaning that he is rather young.
- *Fuzzy datatypes* generalize crisp values by using a fuzzy membership function. For example, instead of considering the crisp value 18, now it is possible to consider about18. The former datatype is incompatible with the value 17.99, whereas the latter one is not. Some popular membership functions, commonly used to define fuzzy datatypes are the trapezoidal, the triangular, the left-shoulder, and the right-shoulder, depicted in Figure 1.

Although there is not an standard fuzzy ontology language, *Fuzzy OWL 2* [5] is a popular choice. The language extends OWL 2 ontologies with OWL 2 annotations encoding fuzzy information using a XML-like syntax. The key idea of this representation is to start with an OWL 2 ontology created as usual, with a classical ontology editor. Then, it is possible to annotate the elements to represent the features of the fuzzy ontology that OWL 2 cannot directly encode. In particular, it is possible to annotate fuzzy axioms by adding a degree of truth, to represent fuzzy datatypes, and to define specific elements of fuzzy ontologies (such as fuzzy modifiers or aggregated concepts). There is a Protégé plug-in making the syntax of the annotations transparent to the users.[5]

For practical reasons, the range of the fuzzy datatypes is usually restricted to an interval $[k_1, k_2]$, e.g., in the *fuzzyDL* reasoner and Fuzzy OWL 2 [5, 6].

[5] `http://www.umbertostraccia.it/cs/software/FuzzyOWL`

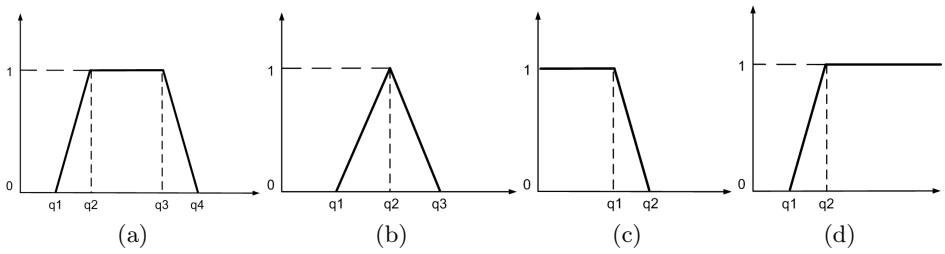

Fig. 1. (a) Trapezoidal; (b) Triangular; (c) Left-shoulder; (d) Right shoulder functions.

2.2 Clustering

This section recaps three well-known unsupervised clustering algorithms, namely k-means, fuzzy c-means, and mean-shift. These learning algorithms cluster a collection of n real data values (or points) denoted x_j into a set of classes or clusters C_i described by means of their centroids (one per cluster) c_i.

K-means groups a set of data into k clusters [18]. The algorithm starts by computing randomly the k initial centroids c_i. Then, it repeats two steps: first, each point x_j is assigned to its nearest cluster, denoted $C(x_j)$, according to the Euclidean distance: $C(x_j) = C_k$ if $\arg\min_i ||x_j - c_i||^2 = k$. Second, the centroids are updated: $c_i = (\sum_{x_j \in C_i} x_j)/|C_i|$. The algorithm aims at minimizing a squared error function and finishes when a stopping criteria is met (typically, after a total number of iterations or when there are no further changes in the centroids).

Fuzzy c-means [2] is an extension where every point can belong to several clusters with different degrees of membership. To this end, the algorithm considers c fuzzy clusters and a matrix of membership degrees μ, where $\mu_{ij} \in [0, 1]$ denotes the membership degree of the datum x_j to the i-th cluster. The positions of the centroids are computed as $c_i = (\sum_{j=1}^{n} \mu_{ij}^m x_j)/\sum_{j=1}^{n} \mu_{ij}^m$, where $m \geq 1$ is a parameter indicating a fuzziness degree. The membership degrees are then updated as $\mu_{ij} = \left(\sum_{k=1}^{c} \frac{\|x_j - c_i\|^{2/(m-1)}}{\|x_j - c_k\|^{2/(m-1)}} \right)^{-1}$.

Mean-shift [7, 8] is widely used in clustering but also in image segmentation. It seeks modes or local maxima of density in a feature space by computing a mean-shift vector $m(x)$. The algorithm defines a window around each point, computes the mean of the data points in the window and then shifts its center to the mean. It uses a Gaussian Kernel K_g to keep track of the nearest neighbors of each x_i according to a bandwidth or window size h. To compute the bandwidth we use the rule of thumb proposed in [23]. This rule can be used to compute a quick estimation of h for a given K_g, and allows to define a local seeking distance $l = \frac{h}{2}$. At the end of the process, the mean-shift vector converges into a set of centroids after removing data points at a too close distance.

The main advantage of fuzzy c-means when compared to k-means is that it is more robust to the random initialization of the centroids. The main advantage

of mean-shift is that it does not require to fix a priori the number of clusters, as both k-means and fuzzy c-means do.

3 The Datil System

Overview. Datil[6] (DATatypes with Imprecision Learner) is a software that automatically learns fuzzy datatypes for fuzzy ontologies from different types of inputs. Datil implements several unsupervised clustering algorithms: k-means, fuzzy c-means, and mean-shift (see Section 2.2). The tool is publicly available.[7].

For each data property in an ontology with a numerical range (or with assertions involving numerical values), Datil collects an array of real numbers corresponding to the values of the property for different individuals. A clustering algorithm provides a set of centroids from these array of values. These centroids are used as the parameters to build fuzzy membership functions partitioning the domain, as illustrated in Figure 2.

Learning the fuzzy datatypes. Assuming that $k \geq 2$, Datil creates the following datatypes using a set of centroids $\{c_1, c_2, \ldots, c_k\}$:

- a left-shoulder function with parameters c_1 and c_2,
- a right-shoulder function with parameters c_{k-1} and c_k, and
- $k-2$ triangular functions, where the i-th triangular function has parameters c_i, c_{i+1}, and c_{i+2}.

As already mentioned, the fuzzy datatypes are defined over a range $[k_1, k_2]$ and not over $(-\infty, \infty)$. For each data property d, Datil uses several strategies to compute such k_1, k_2 for all the fuzzy datatypes defined for d:

- First, checking if the range of dp is of the form [>= k1, <= k2], where >= and <= denote xsd:minInclusive and xsd:maxInclusive OWL 2 facets, respectively, that constrain the possible values of an OWL 2 numerical datatype.
- Otherwise, it computes the minimum (min) and the maximum (max) of the array of real numbers formed by the values of dp and defines $k1 = min - \sigma$ and $k2 = max + \sigma$ for some σ. In the case of mean-shift, $\sigma = h/2$.

For small numbers of clusters ($k \leq 7$), Datil automatically provides readable names for the fuzzy datatype labels in an automatic way. For example, for a data property SkinTemperature, the tool can create 7 fuzzy datatypes VeryVeryLowSkinTemperature, VeryLowSkinTemperature, LowSkinTemperature, NeutralSkinTemperature, HighSkinTemperature, VeryHigSkinTemperature and VeryVeryHigSkinTemperature. If the number of clusters was 6, NeutralSkinTemperature would be omitted. For an arbitrary number of clusters $k > 7$, label names are formed by concatenating the name of the data property dp and an integer number (the order of the fuzzy datatype according to an increasing value of the smaller centroid).

If the clustering algorithm provides a unique centroid c, Datil only creates one triangular function with parameters $c - \sigma$, c, and $c + \sigma$.

[6] Dátil is the Spanish for the date fruit.

[7] `http://webdiis.unizar.es/~ihvdis/Datil`

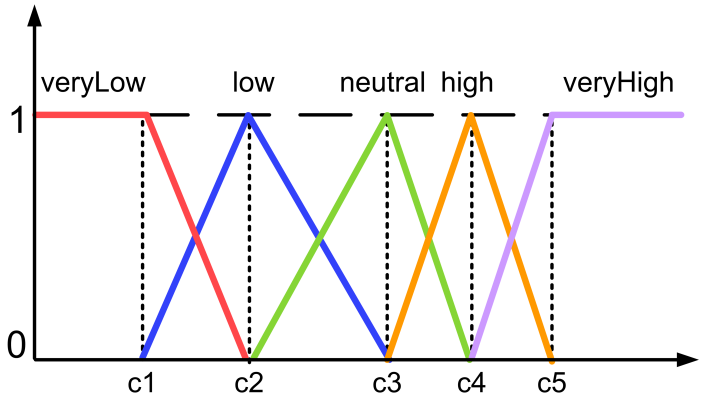

Fig. 2. Some fuzzy membership functions built from the centroids.

Input formats. Datil supports 3 possible input formats: .owl, .fdl, and .csv.

- .owl format correspond to ontologies in the standard language OWL 2. Files can be classical ontologies but also fuzzy ontologies in Fuzzy OWL 2; in the latter case the annotations with the fuzzy information are discarded. Datil restricts itself to data property assertions and range restrictions. A semantic reasoner is used to retrieve both explicit and implicit axioms.
- .fdl is the own syntax of the fuzzyDL reasoner to define fuzzy ontologies [6]. As in previous case, Datil restricts itself to data property assertions and range restrictions and does not consider any fuzzy information (not even the degree of truth of the assertions).
- .csv (Comma-Separated Values) format consists of large data (numbers and text) in plain text. Each record (row) in the file contains one or more fields (columns) separated by commas. In this case, the clustering algorithm takes as an input all the values for a given column. Typically, the first line of the file is special and contain the column names, which should correspond to datatype properties from an ontology.

Output format. Datil supports 2 possible output formats: .owl, and .fdl. The output is a fuzzy ontology with some fuzzy datatype definitions that can be represented as OWL 2 annotations (as specified in Fuzzy OWL 2) or as fuzzyDL (.fdl) axioms. If the output is a .fdl file, apart from the definition of the fuzzy datatypes, Datil adds further axioms required by fuzzyDL reasoner (functional and range data property axioms).

If the input was an ontology (.owl or .fdl), the output is an extension with the new elements. If the input was a .csv file, the output ontology is created from scratch, and the user can import it later on from another ontology file.

Dependencies. Datil is implemented in Java and uses some external libraries:

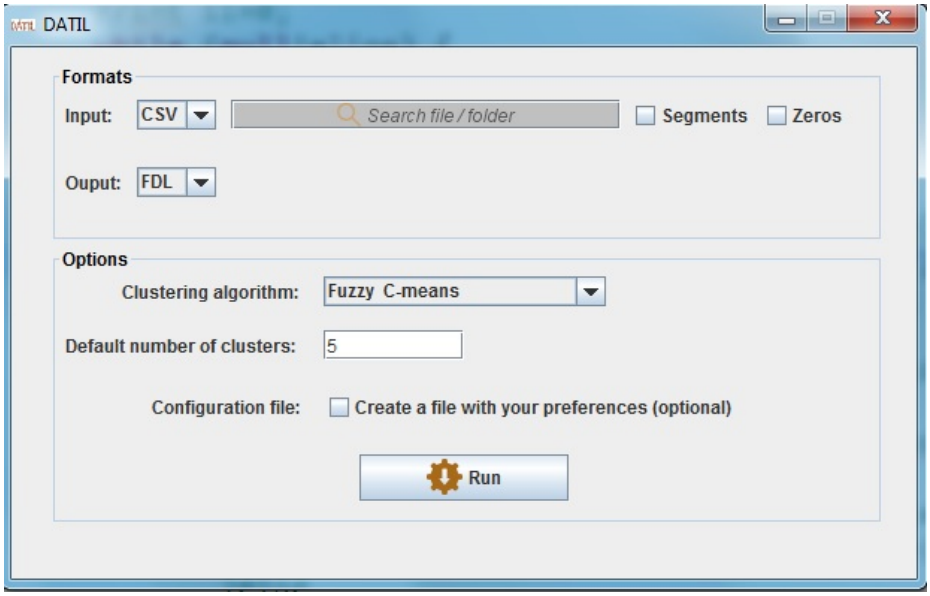

Fig. 3. Snapshot of the main user interface of Datil.

- *OWL API*[8] [14] is an ontology API to manage OWL 2 ontologies in Java applications and provides a common interface to interact with DL reasoners. It can be considered as a de facto standard, as the most recent versions of most of the semantics tools and reasoners use the OWL API to load and process OWL 2 ontologies.
- *HermiT*[9] is an OWL 2 ontology reasoner [12]. It completely supports the language and implements several optimization techniques. HermiT is implemented in Java, and is accessible through several interfaces, including the OWL API. We use it to retrieve all the data property assertions, not only those explicitly represented in the ontology but also the implicit ones.
- *Java-ML* (Java Machine Learning Library)[10] is a collection of machine learning algorithms and a common Java interface for those algorithms. Although Java-ML provides an implementation of k-means, we have implemented our own version the algorithm. However, we do use its Java data structures in all of our clustering algorithms.
- *fuzzyDL*[11] is a fuzzy ontology reasoner [6]. It supports a fuzzy extension of a significant fragment of OWL 2 and supports a notable plethora of reasoning services. The possible input formats are Fuzzy OWL 2, its own syntax in FDL

[8] `http://owlapi.sourceforge.net`
[9] `http://www.hermit-reasoner.com`
[10] `http://java-ml.sourceforge.net`
[11] `http://www.umbertostraccia.it/cs/software/fuzzyDL/fuzzyDL.html`

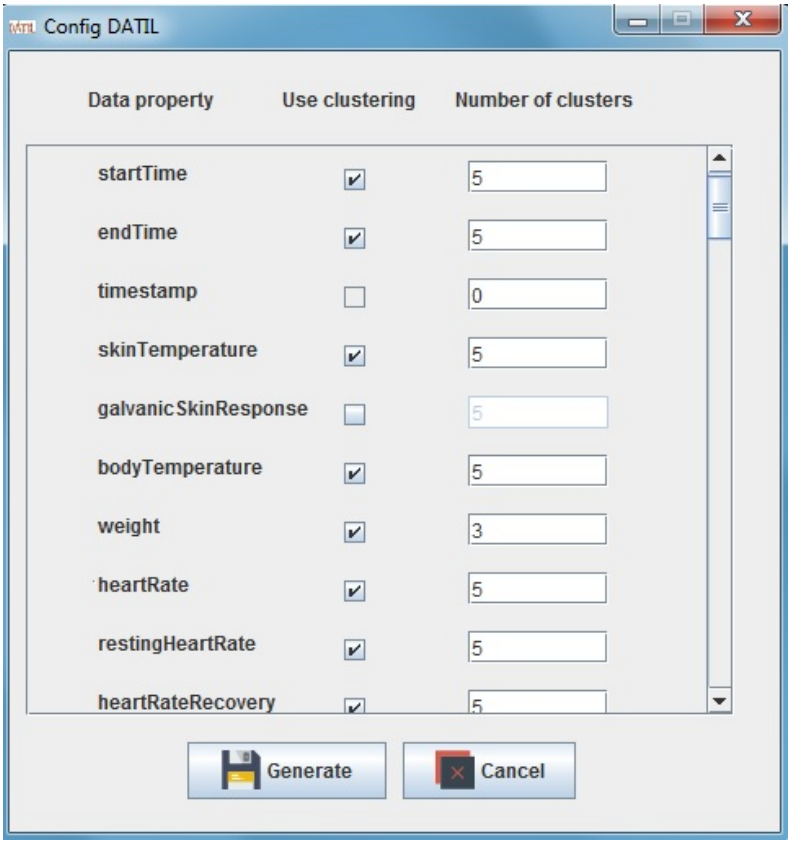

Fig. 4. GUI to create a configuration file in Datil.

format, and a Java API. We use fuzzyDL to translate fdl fuzzy ontologies into Fuzzy OWL 2.

Configuration options. Datil requires several parameters:

- The input and output formats.
- The input file. The output file is not a parameter; Datil uses the same filename (with a different filename extension if there is a format change).
- The selected clustering algorithm.
- The number of clusters (only for k-means and fuzzy c-means) for all the properties, or a different number for each property.
- The properties for which to learn the fuzzy datatypes.
- Use of zeros (only for .csv files): zero values can be either taken into account or skipped (in practice, they are often used just to represent empty data).
- Use of segments (only for .csv input files), i.e., the first column can have a special meaning classifying each row as belonging to a different category.

User interface Figure 3 shows a snapshot of the main user interface, where the user can configure most of the previously mentioned parameters: input and output formats, input file, use of zeros and segments, clustering algorithm, and global number of clusters. In case of .csv files, the user can select a folder with several files rather than a single one. By default, Datil learns fuzzy datatypes for all data properties with a numerical range.

It is also possible to use a *configuration file* to select a subset of the data properties and/or select a different number of clusters for each of them if the clustering algorithm is not mean-shift. Figure 4 shows how Datil supports the creation of the file by making its syntax transparent to the user. Thanks to the configuration file the user does not need to repeat the selection in future executions. If the system does not find it, it runs with the default values.

4 Use Case: Semantic Lifestyle Profiling

Lifestyle can be defined as a collection of routines and behaviors shaped by the social, economic, and environmental structure around a person. In a computational application the behaviors are represented by measurements from wearable sensors. The lifestyle of an individual can then be modeled by the statistics of the measurements conditioned by the elements of the surrounding structure. The percentage of day time spent doing certain activities, the locations where a person spends his time, and the amount of times or frequency with which a person performs an activity or visits a place, are examples of data that provide a good idea of the person's type of lifestyle.

A model of a lifestyle can be based on matching a predefined semantic template to the data. We propose discovering these templates blindly from the data using machine learning techniques [9]. Semantically meaningful and interpretable models to better understand the underlying statistics of individual lifestyle patterns of people is not a trivial task because of the variability of the individuals. Even if technology allows for a broad spectrum of sensors, it is not straightforward to choose the most appropriate data acquisition, data imputation and data fusion techniques [19]. Semantics can enhance data-driven processes and improve accuracy and precision of recognition in human activities [10, 11].

To serve application development in Ambient Intelligence scenarios ranging from activity monitoring in smart homes to active healthy aging or lifestyle profiling, we have developed a fuzzy ontology that allows to describe the lifestyle of a user given its digital footprints such as wrist-born activity trackers, GPS and mobile phone applications[12]. The ontology includes information such as height, weight, locations, cholesterol, sleep, activity levels, activity energy expenditure, heart rate, or stress levels, among many other aggregated features.

In order to populate the ontology, the only information that we have used are real data obtained from digital traces such as sleep and activity sensors and other wearable devices. In particular, we used 40 records of volunteers of middle age

[12] `https://github.com/NataliaDiaz/Ontologies`

living in the Eindhoven area (The Netherlands) [9]. These data were provided by a private company and are confidential (little details are thus given in this paper for privacy reasons). However, we would like to point out that this scenario is a typical case where we do not have data about the membership to classes but we do know the values of several data properties.

The next step is to learn some fuzzy datatypes for each of the 68 data properties and 14 day segments by using Datil. If we use k-means with $k = 5$ fuzzy datatypes, we end up trying to learn 4760 fuzzy datatypes, although for some combinations of data property and day segment there were no data. For instance, one of the learnt fuzzy datatypes for the data property HighSkinTemperature restricted to the day segment day AtWork is HighSkinTemperatureAtWork, defined (in Fuzzy OWL 2 Manchester syntax) as follows:

```
Datatype: HighSkinTemperatureAtWork
  Annotations:
    fuzzyLabel "<fuzzyOwl2 fuzzyType="datatype">
      <Datatype type="triangular" a="22.54" b="27.97" c="30.22" />
       </fuzzyOwl2>"

  EquivalentTo:
    (xsd:double[>= "-6.71"^^xsd:double] and xsd:double[<= "37.97"^^xsd:double])
```

Note that indeed the fuzzy datatypes add more knowledge, in the sense that we can make new inferences. For example, two individuals with slightly different skin temperatures at work, even if such values are different than the center of the triangular function 27.97, would be compatible with the fuzzy datatype HighSkinTemperatureAtWork with different degrees of truth.

To compute the categorization of a person into some lifestyle pattern, we propose the following process:

1. Build a crisp ontology with the features of interest, using domain data scientists with diet and specialists that monitor cardiac disease patients. At this point, experts should identify lifestyle patterns like MediterranLuncher.
2. Populate it with data property assertions obtained from wearable devices.
3. Learn some fuzzy datatypes from the data property assertions using Datil.
4. Define some preliminary rules (concept equivalence or fuzzy subclass axioms) with the help of an expert. Some of the concepts will have complex definitions, being defined in terms of the learnt datatypes. For example, one can define the concept of MediterranLuncher from the (late) time and the (high) duration of the lunch breaks.

   ```
   (define-concept MediterraneanLuncher (g-and
     (some StartTimeFixedAfternoon HighStartTimeFixedAfternoon)
     (some ActivityDurationFixedAfternoon HighActivityDurationFixedAfternoon)
   )
   ```

5. Ask a fuzzy semantic reasoner to retrieve all the instances of each of the defined concepts together with the degrees of membership.
6. Represent it as fuzzy concept assertions and add them to the fuzzy ontology.
7. Run a learning algorithm starting from memberships to fuzzy classes (Fuzzy DL-Learner) to get the final rules. Final rules are complex definitions of lifestyle pattern concepts, similar to the preliminary rules but automatically derived from the real data.

5 Conclusions and Future Work

This paper has presented Datil, a novel tool which is able to learn fuzzy datatypes automatically. Datil supports different input and output formats, allowing both enriching existing fuzzy ontologies and helping in the extension of crisp ontologies to the fuzzy case. Three well-known clustering algorithms are implemented to compute a set of centroids from available data, and then the fuzzy datatypes are defined after them. We have also discussed how fuzzy datatype learning has been applied to a real-world application: semantic lifestyle profiling. In order to characterize the lifestyle of people given their digital footprints, we start from numerical data obtained from different sensors and use Datil to cluster them into fuzzy datatypes interpretable by human users.

Fuzzy datatype learning is a complementary technique to other approaches for fuzzy ontology learning. In particular, our implementation could be used to extend Fuzzy DL-Learner, a system learning fuzzy subclass axioms. Other ideas for future work include the implementation of more sophisticated clustering algorithms. Last but not least, we plan to apply Datil to learn fuzzy datatypes in some more real-world domains; this practical experience will surely provide more ideas to extend our tool.

Acknowledgment I. Huitzil was partially funded by Universidad de Zaragoza - Santander Universidades (Ayudas de Movilidad para Latinoamericanos - Estudios de Doctorado). N. Díaz-Rodríguez acknowledges AAPELE.eu EU COST Action IC1303 and EU Erasmus+ Funding; part of her work was done during internship at Philips Research. I. Huitzil and F. Bobillo were partially supported by the project TIN2016-78011-C4-3-R. Special thanks are due to Aki Härmä and Rim Helaoui (Philips Research) for their invaluable help with lifestyle real data.

References

1. Alexopoulos, P., Wallace, M., Kafentzis, K., Askounis, D.: Ikarus-onto: a methodology to develop fuzzy ontologies from crisp ones. Knowledge and Information Systems 32(3), 667–695 (2012)
2. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Advanced Applications in Pattern Recognition, Plenum Press, 2nd edn. (1987)
3. Bobillo, F., Cerami, M., Esteva, F., García-Cerdaña, À., Peñaloza, R., Straccia, U.: Fuzzy description logics. In: Cintula, P., Fermüller, C., Noguera, C. (eds.) Handbook of Mathematical Fuzzy Logic Volume III, Studies in Logic, Mathematical Logic and Foundations, vol. 58, chap. XVI, pp. 1105–1181. College Publications (2015)
4. Bobillo, F., Ruiz, M.D., Gómez-Romero, J., Sánchez, D.: On the application of data mining techniques to graded ontology building. In: Actas del XVIII Congreso Español sobre Tecnologías y Lógica Fuzzy (ESTYLF 2016). pp. 142–143 (2016)
5. Bobillo, F., Straccia, U.: Fuzzy ontology representation using OWL 2. International Journal of Approximate Reasoning 52(7), 1073–1094 (2011)
6. Bobillo, F., Straccia, U.: The fuzzy ontology reasoner fuzzyDL. Knowledge-Based Systems 95, 12–34 (2016)

7. Cheng, Y.: Mean shift, mode seeking, and clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(8), 790–799 (1995)
8. Comaniciu, D., Meer, P., Member, S.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24, 603–619 (2002)
9. Díaz-Rodríguez, N., Härmä, A., Helaoui, R., Huitzil, I., Bobillo, F., Straccia, U.: Couch potato or gym addict? Semantic lifestyle profiling with wearables and knowledge graphs. In: Proceedings of the 6th NIPS Workshop on Automated Knowledge Base Construction (AKBC 2017) (December 2017)
10. Díaz-Rodríguez, N., León-Cadahía, O., Pegalajar-Cuéllar, M., Lilius, J., Delgado, M.: Handling real-world context-awareness, uncertainty and vagueness in real-time human activity tracking and recognition with a fuzzy ontology-based hybrid method. Sensors 14(10), 18131–18171 (2014)
11. Díaz-Rodríguez, N., Pegalajar-Cuéllar, M., Lilius, J., Delgado, M.: A fuzzy ontology for semantic modelling and recognition of human behaviour. Knowledge-Based Systems 66, 46–60 (2014)
12. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: HermiT: An OWL 2 reasoner. Journal of Automated Reasoning 53(3), 245–269 (2014)
13. Gómez-Romero, J., Bobillo, F., Ros, M., Molina-Solana, M., Ruiz, M.D., Martín-Bautista, M.J.: A fuzzy extension of the semantic building information model. Automation in Construction 57, 202–212 (2015)
14. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web Journal 2(1), 11–21 (2011)
15. Iglesias, J., Lehmann, J.: Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In: Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011). pp. 1323–1328 (2011)
16. Lisi, F.A., Straccia, U.: A logic-based computational method for the automated induction of fuzzy ontology axioms. Fundamenta Informaticae 124(4), 503–519 (2013)
17. Lisi, F.A., Straccia, U.: Learning in description logics with fuzzy concrete domains. Fundamenta Informaticae 140(3–4), 373–391 (2015)
18. Lloyd., S.P.: Least squares quantization in PCM. IEEE Transactions on Information Theory 28(2), 129–137 (1982)
19. Pires, I.M., Garcia, N.M., Pombo, N., Flrez-Revuelta, F.: From data acquisition to data fusion: A comprehensive review and a roadmap for the identification of activities of daily living using mobile devices. Sensors 16(2), 184 (2016)
20. Staab, S., Studer, R. (eds.): Handbook on Ontologies. International Handbooks on Information Systems, Springer (2004)
21. Straccia, U.: Foundations of Fuzzy Logic and Semantic Web Languages. CRC Studies in Informatics Series, Chapman & Hall (2013)
22. Straccia, U., Mucci, M.: pFOIL-DL: Learning (fuzzy) $\mathcal{EL}$ concept descriptions from crisp OWL data using a probabilistic ensemble estimation. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC-15). pp. 345–352. ACM, Salamanca, Spain (2015)
23. Turlach, B.A.: Bandwidth selection in kernel density estimation: A review. In: CORE and Institut de Statistique (1993)
24. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview (2008), [Online] Available: `http://www.w3.org/TR/owl2-overview`
25. Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)