Mapping Data Sets to Concepts using Machine Learning and a Knowledge based Approach

Andreas Bunte[1], Peng Li[1] and Oliver Niggemann[1,2]
[1]*Institute for Industrial IT, Langenbruch 6, 32657 Lemgo, Germany*
[2]*Fraunhofer IOSB-INA, Langenbruch 6, 32657 Lemgo, Germany*

Keywords: Clustering, Ontology, Knowledge, Reasoning, Classification, Concept Learning.

Abstract: Machine learning techniques have a huge potential to take some tasks of humans, e.g. anomaly detection or predictive maintenance, and thus support operators of cyber physical systems (CPSs). One challenge is to communicate algorithms results to machines or humans, because they are on a sub-symbolical level and thus hard to interpret. To simplify the communication and thereby the usage of the results, they have to be transferred to a symbolic representation. Today, the transformation is typically static which does not satisfy the needs for fast changing CPSs and prohibit the usage of the full machine learning potential. This work introduces a knowledge based approach of an automatic mapping between the sub-symbolic results of algorithms and their symbolic representation. Clustering is used to detect groups of similar data points which are interpreted as concepts. The information of clusters are extracted and further classified with the help of an ontology which infers the current operational state. Data from wind turbines is used to evaluate the approach. The achieved results are promising, the system can identify its operational state without an explicit mapping.

1 INTRODUCTION

Machine learning techniques are getting more and more common in industries. They are able to derive information out of data which can be used for tasks, such as anomaly detection, predictive maintenance and optimization. A major limitation for all these tasks is the sub-symbolic representation of the algorithms' results, so there is no meaning which can be assigned to the data set. So, mapping to meaningful results has to be performed manually, which has to be done for every single type of machine, since the mapping is not generic. This is not feasible for fast changing cyber physical system (CPS). The results have to be represented on a symbolical level, which enable an easy exchange of information without manual adaption. Symbolical information are represented through concepts, which share a common understanding of a specific thing. But how can the machine learning results be transferred to concepts?

To answer the question, a formal definition of concepts is introduced, according to (Cimiano et al., 2005). A concept c is a name for the aggregation of things or objects which share a specific list of common attributes. The assignment of a thing or an object to a concept can be defined as follows. A thing $t \in T$, where T is the set of all things, belongs to a concept $c \in C$, where C is the set of all concepts, if and only if all attributes of c are fulfilled by t. So, to automatically assign data sets to concepts, the attributes of the data sets have to be identified and concepts have to be defined. For example, if the concept *creature* is defined with the attribute *has legs*, then humans are assigned to the concept, because they share the attribute. But the definition is not good, because animals such as fishes or snakes do not have the attribute and thus they are not assigned to the concept.

There are some works which try to learn concepts directly from the data and thus do not need prior knowledge. For example Lake (Lake, 2014) analyzes how to learn concepts from very small sets of training data. Lake uses visual concepts for his work, but it should be transferable to different input data. So, concepts can be learned, but they are independent of each other, e.g. without a hierarchical order, which would be useful for communication e.g. by using superordinate terms. This approach is presented in Figure 1, where two sets of data are aggregated to two concepts.

Since the concepts should be used for an information exchange, it seems not suitable to learn concepts in each devices without prior knowledge. Especially,

Bunte, A., Li, P. and Niggemann, O.
Mapping Data Sets to Concepts using Machine Learning and a Knowledge based Approach.
DOI: 10.5220/0006590204300437
In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART 2018) - Volume 2*, pages 430-437
ISBN: 978-989-758-275-2

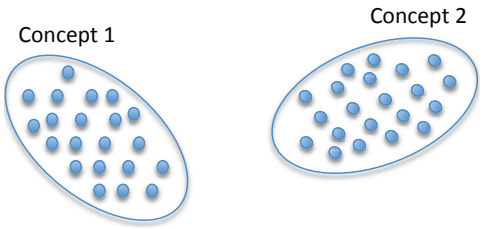

Figure 1: Concepts can be learned, but they are independent of each other.

the relations are important to interpret the concepts correctly, additionally, they cover knowledge that can be used to infer new facts. Concept learning approaches as in Figure 1 do not learn relations, since their learning is difficult and error-prone (more details are given in section 2). Instead, defined concepts can be used to map data sets to these concepts, as shown in Figure 2. It requires a definition of concepts, but ensures a comprehensible and constant classification to concepts. Nevertheless, it is an abstraction where values of a continuous space are classified to discrete values, so there will be a loss of information.

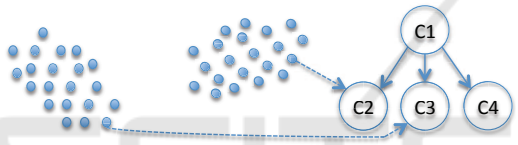

Figure 2: Data sets are mapped to predefined concepts.

This work follows the second approach, which maps data sets to prior defined concepts. The prior defined relations enable a reasoning of new facts, which is an additional benefit. Not only the mapping should be done automatically, it should be realized with unsupervised machine learning techniques. The prior knowledge and the unsupervised machine learning technique enable a handling of new situations. As an example, the operational states of a wind turbine will be automatically determined. It has the obvious advantage, that the states have to be defined once in an abstract manner and can be further used to automatically identify the states of different types of machine.

The contribution of this paper is a novel approach to assign data sets to a corresponding concept. This is achieved by using prior defined concepts which enable the reasoning of new knowledge about the data set. This approach bridges the gap between subsymbolic and symbolic results and thus enables machines to express results in an understandable manner.

The paper is structured as follows: An overview about the state of the art is given in section 2. Section 3 introduces the approach in detail. The results are presented in section 4 by using a concrete use-case. Section 5 summarizes the work.

2 STATE OF THE ART

In this section a literature review of ontology and concept learning is given. Ontology learning is not the main topic but relevant, because it is similar to this work by adding instances to the ontology and classify them. To the best of the authors knowledge, no literature could be found which maps data sets to concepts and thus use an equivalent approach.

In principle, the learning of ontologies is possible, but it is almost used for lightweight ontologies such as taxonomies. Hierachichcal clustering or decision trees are typical methods, which are used for it, because they can be translated directly into lightweight ontologies. Nevertheless, existing approaches often focus on linguistic properties such as (Suma and Swamy, 2016) or (Ocampo-Guzman et al., 2009). That means structured, semi-structured and unstructured texts are used to derive relations between words by using similarity measures such as syntax, properties of word or the probability of occurrence (Drumond and Girardi, 2008). Such approaches are not transferable to CPS, since there are no text bases available.

There is some work in learning more complex ontologies, but these approaches lack of accuracy for a real world applications. For example, Zhu (Zhu et al., 2013) used Bayesian networks to learn more complex ontologies. But even if it is one of the best algorithms that can be found in the literature, the F1 score is between 0.3 and 0.8. Lehmann (Lehmann and Voelker, 2014) concludes in his work that good quality ontologies need a close interaction with humans, so automatic or semi-automatic generated ontologies have a poor quality. This indicates the difficulty of ontology learning.

Concept learning means that patterns in the data should be learned and assigned to concepts. Most of the approaches that can be found in the literature use online methods, which require an interaction with humans. For example, Araki (Araki et al., 2013) developed a Multimodal Latent Dirichlet Allocation (MLDA) algorithm for a robot which enables an online learning. For the evaluation, 120 objects are classified in 24 categories. The algorithm learns a word for every object and categorizes each object to a class, but there is a human in the loop, which should be avoided in this work. Alibeigi (Alibeigi et al., 2017) introduces a method for robots to learn and imitate motions, but there is also a human in the loop. Many approaches are dealing with texts or natural language and learn concepts out of it, such as (Ali et al., 2017) or (Jia et al., 2017). Jia (Jia et al., 2017) uses spoken texts for the identification of concepts. They used an example to reserve a table in a restaurant, where

several information have to be exchanged and the system asks for missing information. But even in this small fixed application scenario, there is much potential to improve the system. A more detailed review about the concept learning is given by Lake (Lake, 2014) or Mahmoodian (Mahmoodian et al., 2013).

The difference of the approaches in literature is that we use a knowledge base. That has the advantage, that the resulting concepts are known and thus they can be used for communication, because everybody, who uses that knowledge base, has the same understanding, e.g. of the concept *error state*. However, it requires a knowledge base which has to be slightly adapted to new types of machines, but there is not much knowledge required and some of the knowledge can be reused.

3 APPROACH

The idea is to map data sets to previously defined concepts as presented in Figure 2. This work uses clustering as a well known machine learning technique. Concepts are defined within ontologies. Ontologies are suitable, because they enable a formal concept description, support reasoning and there are tools available which ensure an easy usage. The aim is to assign data to concepts and thus bridge the gap between sub-symbolic and symbolic layer. This is achieved by five steps (see Figure 3), which are described in more detail in this section:
(i) Extraction of data about the clusters;
(ii) Data pre-processing;
(iii) Data discretisation;
(iv) Instance creation for the current state;
(v) Reasoning of the final operational state;

Step *(i)* is closely related to the used machine learning technique. Clustering is used in this approach, but it is also possible with other classification algorithms. The so called mapping unit consists of step *(ii)* and *(iii)*. The steps *(iv)* and *(v)* performed in the knowledge base, which is an ontology in this approach. The mapping unit must be able to query the knowledge base, to get knowledge about the signals.

3.1 Data Extraction

For more complex CPS, the system behavior might consist of multiple operational states that depend on different factors, e.g. work environments or operations of the systems. For example, a wind turbine has various operational states, namely idle, part load, full load or error state. Therefore, data from all sensors and actuators is acquired during the operational phase. When cluster analysis is performed on a data set, multiple clusters can be recognized. Each cluster corresponds to a particular operational state of a given CPS.

Typically, clustering provides data about the position of the data point or about the cluster where the data point is assigned to. This is suitable for applications such as anomaly detection, but for this use case detailed information about the clusters is necessary. This first approach just focuses on clusters and does not consider single outlier. In this step, the information that describes the clusters sub-symbolically will be extracted in the following manner. The names of the variables in the data sets will be passed to the mapping unit, so that the semantic information of each cluster can be retained as much as possible. All possibly relevant data is captured, in order to describe the clusters as good as possible with statistical values. Therefore, clusters are described in all dimensions (dimensions are equal to the number of input signals), so the maximum, minimum and mean value of all signals are extracted as well as the variance of each cluster. These data is provided to the mapping unit, which does the further processing.

3.2 Data Preprocessing

As known from every data mining application, the data preprocessing is an important step. Since there are different scales of variables in the CPS, their impact has to be adjusted. To compare the influence of these variables on the mapping task, each signal will firstly be normalized in the range of [0, 1] overall clusters.

The task is to identify one representative value for each signal of each cluster. Therefore, all the information (minimum, mean, maximum and variance) has to be processed to one significant value. It is suggested that the mean value provides a good representation, if the variance of the signal is not to high. If there is a huge variance (in extreme it could cover the whole range between 0 and 1), it is difficult to identify one characteristic value for the signal, thus the signal is suggested as *not characteristic*. The not characteristic values are determined as all other values, but they are marked with a minus. Depending on the needs, the threshold should be adapted. Generally speaking: If there are many dimension and few states, then the boundary to set a dimension as *not characteristic* should be lower. However, the presented approach focuses on determining operational states. For purpose of configuration it might be interesting which

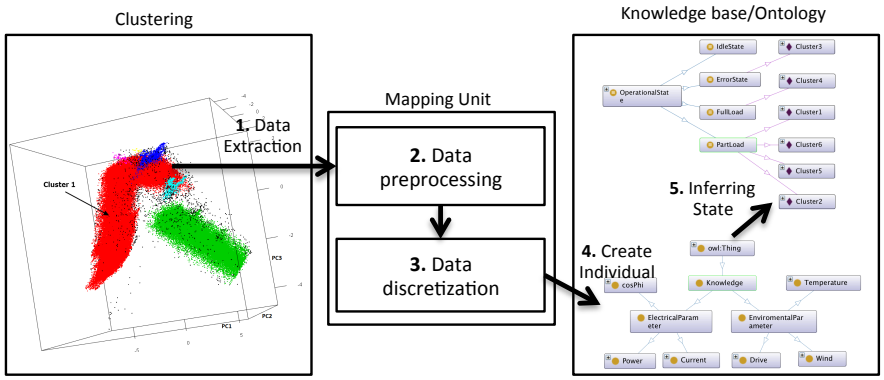

Figure 3: The mapping is done in five steps.

signals have a huge variance, or for optimization the minimum and maximum values might be more interesting. In such cases, the data preprocessing has to be adapted according to the needs. Nevertheless, the output of the step is a list where each of the dimensions have a dedicated value.

3.3 Data Discretization

In this section it is described how the discretization of the cluster information is performed. The discretization bases on signal values determined in the previous step. Every signal value $s \in S$, where S is the set of all signals, is assigned to a category $y \in Y$, where Y is the set of all categories, using the function f. Six categories Y are used to represent the ordinale scale, named *no* (α), *low* (β), *mean* (γ), *high* (δ), *very high* (ε) and *not characteristic* (ζ). The number of categories should be chosen according to the application, but for many applications six categories seems to be suitable.

Some signals do not use the whole range between 0 and 1 which leads to a wrong discretization, because the categories are ordered through the whole range between 0 and 1. If the signal just sligthly change, e.g. between 0.8 and 1, values would only alternate between two categories. For example, the power factor should be higher than 0.9 and a value near 0 would immediately lead to a blackout, so the category *mean* should be significantly bigger than a value of 0.5. One could suggest that it is fixed by the normalization, if the value just varies between 0.8 and 1, but it is mostly not the case for long data records, caused by e.g. a disconnect of the grid, a blackout or measurement error. To prevent such wrong classifications, every signal has a parameter to choose the mean value μ. This value μ can be defined in the knowledge base which influences the discretization of signals. If the parameter is not defined, μ is set to 0.5 as default value. But also the variance of some signals can be smaller, e.g. again the power factor varies between 0.8 and 1, but 1 should be very high and 0.8 should be low. Therefore another parameter λ can be set, which scales the range down. Again, this is used to cover the whole range of categories and does not end up in a single class over the whole range. If there is no parameter defined, λ is set to 1. So, to perform this task, an access to the knowledge base is required, to query the values μ and λ for each signal.

It could happen that a signal which is mandatory for the classification of operational states has a high variance and is thus classified as *not characteristic*. This would cause some trouble, because without such signals, it is not possible to classify the state. To prevent this, an additional parameter *relevance* can be defined for every signal in the knowledge base.[1] The *relevance* can be manually set to a value between 0 and 1, where 1 means that the signal is mandatory for the cluster description and 0 indicate no relevance of the signal (which will probably occur rarely in practical application). So, all values with a relevance higher than 0.5 are changed to a positive value. Afterwards, the borders for the values are defined as follows:

$$f(s) = \begin{cases} \alpha & \text{if } 0 \leq s \leq \frac{\lambda}{10} \\ \beta & \text{if } \frac{\lambda}{10} < s \leq \mu - \frac{(1-\mu)}{3\cdot\lambda} \\ \gamma & \text{if } \mu - \frac{(1-\mu)}{3\cdot\lambda} < s \leq \mu + \frac{(1-\mu)}{3\cdot\lambda} \\ \delta & \text{if } \mu + \frac{(1-\mu)}{3\cdot\lambda} < s \leq 1 - \frac{(1-\mu)}{3\cdot\lambda} \\ \varepsilon & \text{if } 1 - \frac{(1-\mu)}{3\cdot\lambda} < s \leq 1 \\ \zeta & \text{if } 0 > s \end{cases} \quad (1)$$

The formula 1 assigns each signal of a cluster to a discrete class. This enables to name properties of clusters, such as "The *wind speed* in cluster 3 is *high*."

[1]The signals that are needed for the state classification could also be determined by the knowledge base with a simple query, but the *relevance* is also used for a unique and meaningful naming of the cluster, as introduced in (Bunte et al., 2017).

So, this can already be used for communication, but it should be transformed to a more abstract level by combining different characteristic combinations to an operational state.

3.4 Create Individuals

Ontologies capture the prior knowledge, but they are also used to capture the knowledge which came up during the operation. They can be modeled with the web ontology language (OWL). The ontology defines individuals (instances) and concepts (classes) which can be ordered hierarchically and restricted class definitions. Individuals are described through object and data properties. Object properties are describing relations between classes or between individuals. It is possible to define relations between classes and instances, but the ontology is getting undecidable with it (Antoniou et al., 2003), which can influence the state inferring. So these kinds of relations are forbidden for this approach, since the decidable profile OWL-DL is used. Data properties are defining values (strings, dates, floats,...) which are used, e.g. for the *relevance* parameter.

The initial ontology has a class *Cluster* where an instance for each cluster is created. The ontology contains additional classes which describe the operational states and individuals which represent all input signals of the machine learning. By creating an individual for each cluster, all information should be stored. The main information is the discretized input signal of each cluster. So, all discrete categories (α...ζ) are modeled as an object property and they are connected to the particular input signals, which are modeled as individuals. Additionally, all continuous values are also stored as data properties to not lose the precise information.

Finally, all information of the clusters is stored as individuals of the class *Cluster* in the ontology. This is just a different representation, but ontologies provide reasoning capabilities, which enable to find logical conclusions based on the formal descriptions, which is done in the next step.

3.5 State Inferring

The reasoning requires prior knowledge to infer new facts, e.g. to infer the operational state. To enable this, all possible states have to be defined formally in the knowledge base, as well as a description of signal types, e.g. power or temperature. The main challenge is to model the knowledge in a way that allows inferences for all possible combinations to exactly one state. It requires some experience and an understanding of available modeling constructs, but then it is feasible and does not take a long time. The section 4 shows exemplary how a state definition looks like. If it has to be adapted to another type of machine, some knowledge can be reused. Signal types hold generally, so just the state definition has to be adapted and the reusable knowledge depends on the similarity of the machine types.

The inferring itself is done by a reasoner, which is used to infer knowledge based on the formal descriptions. Among other things, the reasoner checks all individuals and identifies class assignments for them. Based on the object properties the reasoner checks which cluster fits to which class. The new type assignments are made according to the description and this represents the current operational state. These states can be used for communication between machines, but also for the communication with humans. The concepts defined in the ontology fit to the humans' understanding, so if there is an error state, it can be shown to humans and they would understand the current situation.

4 RESULTS

The results for a concrete use-case are presented in this section. Clusters detected in the wind power plant data should be automatically determined to an operational state. The data set consists of 11 continuous signals with a time resolution of 10 minutes. Over 230,000 data points generate six clusters, but most of the data is represented in two clusters, (see the clustering of Figure 3). To provide a more detailed understanding of the approach, the results of all five steps are presented for this example.

In the **first step**, data about the maximum, mean, minimum value and variance of every signal in every cluster center is extracted. It is represented as follows:
Cluster1 windSpeed 12.0 5.2 0.0 3.1 rotorSpeed 16.9 9.7 0.0 5.94...
Cluster2 windSpeed 16.1 10.7 5.7 4.2 rotorSpeed 18.0 17.3 16.6 0.72...
...

In the **second step**, the data is preprocessed, so at first it is normalized. The further processing can be adapted to the specific use case. In this use case, the variance in most of the clusters is low, such that the mean values are suitable to describe the cluster. The following data is provided to the discretization step:
Cluster1 windSpeed 0.20 rotorSpeed 0.54 ...
Cluster2 windSpeed 0.41 rotorSpeed 0.96 ...
...

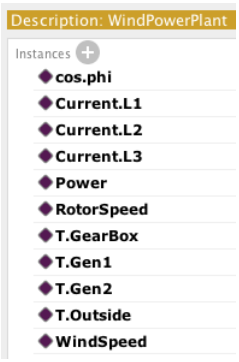

Figure 4: All signals of the plant have to be modeled as individual, before.

The step **data discretization** transforms all values into a category between *no* and *very high* or to *not characteristic*. As described, the parameters λ and μ can be used to adapt the categorization. This is done e.g. for the signal *rotorSpeed*, because even at low wind speeds the rotor has a fast drive. μ was set to 0.8. The parameter *relevance* is not needed for this use case. To increase the readability, every signal can have a name that is more understandable or common in the community. In this example, the signal *windSpeed* has the colloquial name WIND which is stored in the individual that represent the input signal windSpeed. The resulting representation is as follows:
Cluster1 windSpeed LOW WIND *rotorSpeed* LOW ROTORSPEED...
Cluster2 windSpeed MEDIUM WIND *rotorSpeed* VERYHIGH ROTORSPEED...

The **fourth step** is to integrate the information from above to the ontology. All signals of the plant must be represented by an individual in the ontology, as shown in Figure 4. Additionally, the categories have to be represented by an object property, as shown in Figure 5. Both are preliminaries, so this is done during the configuration beforehand. Since this is done, the information can be transferred to an individual, which is an automatic process. For every cluster, a new individual of the generic type *cluster* is created and the name is generated by the word *cluster* and a consecutive numbering. All eleven signals are combined to one individual by using the categories as object property which connects it with the individual that represent the signal. This is presented in Figure 6. Additionally, the original values are stored as data property. This is not needed, but can provide additional information e.g. if the categories are reorganized.

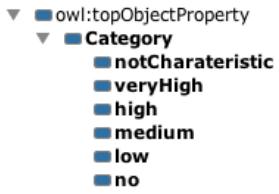

Figure 5: The categories are modeled as object properties.

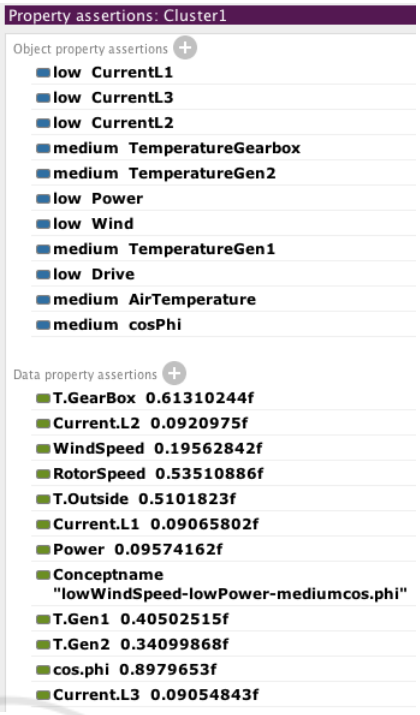

Figure 6: Example of the individual cluster 1.

In **step five**, the reasoner will infer the states by assigning individuals to classes, the classes represent operational states in this example. Table 1 shows the definition of states, which are modeled during the configuration. They only base on the two properties wind speed and power, in this example. The operational states are not defined very strictly, e.g. no wind and low power is defined as idle state, because transitions between two categories are always critical. Since two signals are classified, it is not known which one switches first to another category, e.g. if wind and power rise.

The class description of *FullLoad* is shown in Figure 7. It is defined as subclass of *OperationalState* and disjoint with the classes *PartLoad*, *IdleState* and *ErrorState*, which indicates that every cluster can have only one operational. But the important definition for the reasoning is the *Equivalent To* property, which is defined regarding table 1 for FullLoad. So all created individuals, which represent a cluster, will be assigned to a class, depending on the attributes they fulfill, by the reasoner. In this example *Cluster4* has all attributes of and thus it is assigned to the class. The yellow background of individual (in Figure 7) indicates that the class assignment was inferred automatically.

Table 1: Definition of operational states.

Wind speed \ Power	no	low	medium	high	very high
no	Idle state	Idle state	Error state	Error state	Error state
low	Idle state	Part load	Part load	Error state	Error state
medium	Error state	Part load	Part load	Full load	Error state
high	Error state	Error state	Part load	Full load	Full load
very high	Error state	Error state	Error state	Full load	Full load

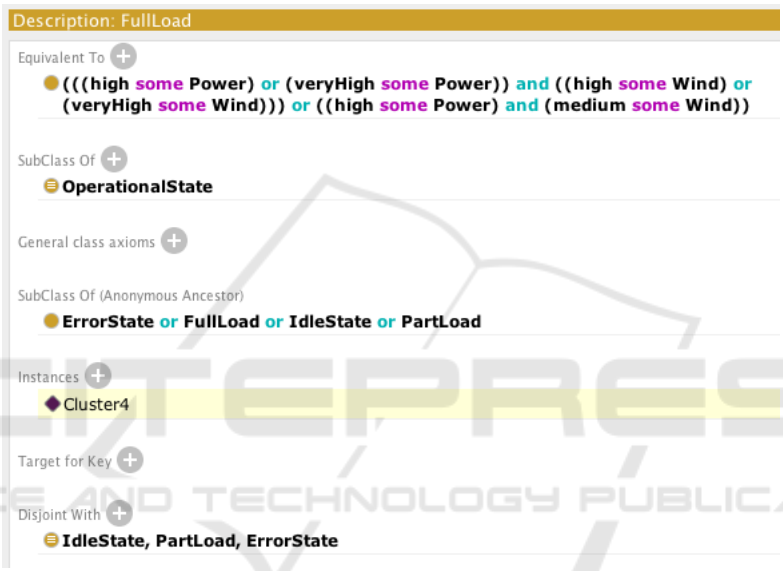

Figure 7: Description of the class full load including one inference.

The overall results are promising. All clusters are classified correctly. *Cluster 3* is correctly classified as error state, because there is high wind, but no power. So it is obviously for humans that it is an error state, but here the machine also has determined it automatically and thus shares the concepts with the humans. *Cluster 4* is detected as full load state, which is correct, with a mean value of 0.95 for power, the wind power plant is nearly at its rated power. *Cluster 1* is classified as idle state, which can be argued about, since the mean value of 0.095 for power is not fully idle, but it is the lowest value of all clusters and includes lots of idle times. All other clusters (cluster 2, 5 and 6) are correctly classified as part load. They have different combinations of *low*/*medium* and *power*/*wind*, additionally the *cos* φ differs.

5 CONCLUSION

This paper introduces an approach for the mapping of data sets to concepts. It maps sub-symbolic data to symbolic information. This approach requires some prior knowledge about signals and operational states. The signal names are mandatory, three parameters are optional and just needed for some signals, which are mostly the same even in different types of machines, such as cos phi, which has always the same characteristic. This is additional expert knowledge that is provided to the system.

In a first step a clustering algorithm is performed to generate clusters from data points. These clusters can be interpreted as concepts. The information about the clusters is extracted and classified to one of six categories, namely *no, low, medium, high, very high* or *not characteristic*. This symbolic representation is added to the ontology. Reasoning is performed in the

ontology as a last step. A reasoner assigns the clusters to classes, which represent the operational state based on its features. The approach was tested at a wind power plant data set with six clusters. All clusters are assigned correctly to the operational modes.

Therefore, the aim, to determine the operational state without explicitly defining it for a use case, is achieved. There are just generic definitions used, which are suitable for similar applications. If the application changes, it has to be adapted only once. But since a classification is made of many continuous signals, it can happen that really small changes lead to another operational state, but this is quite normal since it is an abstraction.

Further work can deal with the generic part of the data preprocessing, since it has to be adapted manually regarding the use case. In particular the normalization can cause some trouble, if there are uncommon values, which deform the range of the values and lead to wrong classification, which should be handled. Furthermore, additional machine learning techniques can be integrated and maybe combined to achieve a better results.

ACKNOWLEDGEMENT

The work was supported by the German Federal Ministry of Education and Research (BMBF) under the projects "Semantics4Automation" (funding code: 13FH020I3) and "Provenance Analytics" (funding code: 03PSIPT5B).

REFERENCES

Ali, I., Madi, N. A., and Melton, A. (2017). Using text comprehension model for learning concepts, context, and topic of web content. In *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, pages 101–104.

Alibeigi, M., Ahmadabadi, M. N., and Araabi, B. N. (2017). A fast, robust, and incremental model for learning high-level concepts from human motions by imitation. *IEEE Transactions on Robotics*, 33(1):153–168.

Antoniou, G., , Antoniou, G., Antoniou, G., Harmelen, F. V., and Harmelen, F. V. (2003). Web ontology language: Owl. In *Handbook on Ontologies in Information Systems*, pages 67–92. Springer.

Araki, T., Nakamura, T., and Nagai, T. (2013). Long-term learning of concept and word by robots: Interactive learning framework and preliminary results. In *International Conference on Intelligent Robots and Systems*, pages 2280–2287.

Bunte, A., Li, P., and Niggemann, O. (2017). Learned abstraction: Knowledge based concept learning for cyber physical systems. In *3rd Conference on Machine Learning for Cyber Physical Systems and Industry 4.0 (ML4CPS)*.

Cimiano, P., Hotho, A., and Staab, S. (2005). Learning concept hierarchies from text corpora using formal concept analysis. *J. Artif. Int. Res.*, 24(1):305–339.

Drumond, L. and Girardi, R. (2008). A survey of ontology learning procedures. In *WONTO*, volume 427 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Jia, R., Heck, L., Hakkani-Tür, D., and Nikolov, G. (2017). Learning concepts through conversations in spoken dialogue systems. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5725–5729.

Lake, B. M. (2014). *Towards more human-like concept learning in machines: Compositionality, causality, and learning-to-learn*. PhD thesis, Massachusetts Institute of Technology.

Lehmann, J. and Voelker, J. (2014). An introduction to ontology learning. In Lehmann, J. and Voelker, J., editors, *Perspectives on Ontology Learning*, pages ix–xvi. AKA / IOS Press.

Mahmoodian, M., Moradi, H., Ahmadabadi, M. N., and Araabi, B. N. (2013). Hierarchical concept learning based on functional similarity of actions. In *First International Conference on Robotics and Mechatronics (ICRoM)*, pages 1–6.

Ocampo-Guzman, I., Lopez-Arevalo, I., and Sosa-Sosa, V. (2009). Data-driven approach for ontology learning. In *2009 6th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, pages 1–6.

Suma, T. and Swamy, Y. S. K. (2016). Email classification using adaptive ontologies learning. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 2102–2106.

Zhu, M., Gao, Z., Pan, J. Z., Zhao, Y., Xu, Y., and Quan, Z. (2013). Ontology learning from incomplete semantic web data by belnet. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, pages 761–768.