DE GRUYTER OPEN
Information Technology and Management Science doi: 10.1515/itms-2015-0006
2015 / 18

Ontology Building Using Classification Rules and Discovered Concepts

Henrihs Gorskis[1], Arkady Borisov[2]
[1, 2] *Riga Technical University*

***Abstract* – Building an ontology is a difficult and time-consuming task. In order to make this task easier and faster, some automatic methods can be employed. This paper examines the feasibility of using rules and concepts discovered during the classification tree building process in the C4.5 algorithm, in a completely automated way, for the purposes of building an ontology from data. By building the ontology directly from continuous data, concepts and relations can be discovered without specific knowledge about the domain. This paper also examines how this method reproduces the classification capabilities of the classification three within an ontology using concepts and class expression axioms.**

***Keywords* – Building, classification tree, ontology.**

I. Introduction

Ontological descriptions can help in understanding large and complex domains, by explaining their concepts. This is useful for communication between different agents and users using the same set of concepts in a given domain. Ontological descriptions for all possible domains do not exist yet. Since building any ontology is a difficult task, the desire to automate this process has existed for some time [1], [2]. In order to build an ontology, one needs to list any seemingly important concepts and relations between concepts. This is often insufficient because the ontology designer may forget some important concept or add a concept, which holds important meaning for the designer but later turns out not to be important for the description of the domain. Once the list of concepts is created, the next step is to find possible descriptions for the concepts. One concept can be the result of a combination of other concepts, or the equivalent of any combination of relationships among concepts. Once the designer of the ontology started to think of definitions, he must add relationships to the ontology in order to use them for the description of the concepts. Other tools are available for the description of concepts, those are, for example, data properties. Data properties allow the ontology designer to use data attributes, which an individual of a concept might hold, for the purposes of classification with a related concept. It is this ability of using data for the description of concepts that enables translation of classification models into ontological descriptions. This paper presents one possible approach aimed at translating a classification tree obtained from the C4.5 algorithm into an ontology, while maintaining its classification abilities.

The data used for building the classification tree was obtained from medical statistics of gastric cancer patients and patients of other gastric ailments [3], and the purpose of the classification tree was to classify data into possible patients of gastric cancer.

Since one of the purposes of an ontology is to classify individuals and connect them to concepts, it seemed logical to try to translate the findings of the classification tree into an ontological description.

II. Difference from Existing Approaches

There are many similar methods and approaches to automatic or semi-automatic ontology building. Most of them follow the same basic principle of using rules as equality statements for concepts. One such approach is described in the paper "New Algorithm for Building Ontology from Existing Rules: A Case Study" [4]. Translation from a rule-based model is achieved by assigning names to discrete attribute values, such as "High", "Mid" and "Low"; numeric rules can then be translated into human readable descriptions. By combining the concepts of the rules with the generated descriptions of attribute values, using equality expressions, automatic classification of individuals can also be achieved. This approach is very user-friendly, but requires that new data are described using the established descriptions, and it loses some accuracy because of the possibly broad meanings of the descriptions.

Another approach to creating ontology descriptions specifically from classification trees is described in "Applying Data Mining for Ontology Building" [5], where concept creation is based directly on the tree nodes. This approach also mostly ignores numeric values and uses named individuals instead. Rule representation is given on an instance level, while the concept layer is reserved for the description of the elements used to create the rules.

The proposed method should not be confused with similarly sounding but very different approaches to classifying data using ontology-aware or ontology-based methods. There are many ontology-based classification methods, which are used to improve classification by using domain knowledge [6]–[9]. There are methods designed for the medical domain [10], [11], chemistry [12] and others. These methods use already existing knowledge described in an ontology language. Improvement of the ontology is only optional in these methods.

The main difference of the approach described in this paper from similar methods is the importance, which is assigned to the attribute value spans found by the C4.5 algorithm. Instead of using whole nodes or only the division point in intervals, we find and use value spans as the basis for concepts, which might hold possibly important ideas. The classification tree

determined important split points for every attribute in the tree. These splitting points emerge in different nodes. However, if one looks at the final leaves of the classification tree, he realises that every leaf operates with intervals or spans of values. We propose that since these spans were of importance to the tree, they might indicate important domain concepts. These values can indicate concepts as simple as "high" or "low". However, since they are generated automatically, other values might have to be considered, for example: "very high", "low average", "in-between" and others.

The concepts obtained by this approach are named by a generator, and an expert is still needed in order to provide human readable and sensible names to the concepts and relations.

III. The Translation Algorithm

The translation algorithm uses the finished classification tree and translates its data, first into rules, later into ontology description. First, a list of all the leaf nodes from the classification tree is obtained. For each leaf, a list of minimal criteria of the leaf is generated. Minimum criteria are obtained by traversing all tree nodes leading to the leaf and noting the minimum and maximum values for every attribute based on the direction of the condition. For example, while traversing the branch (chain of inner nodes) of the classification tree, leading to the leaf, the attribute "A1" is compared multiple times. The first time it is tested for the condition "A1 ≤ 10", later it is tested again for "A1 ≤ 8", and again for "A1 ≤ 5". Only the very last condition will be used for the minimum criteria, since less or equal to five is both a necessary condition and fulfils all other conditions. For every attribute this must be done twice, once for "less or equal to", and another time for "more than". Ignoring some splitting points does not mean the values are lost; they will be used in other branches, for other leaf nodes. The minimal criteria are then translated into minimum or necessary spans for every attribute. There can be two types of spans – spans containing positive or negative infinity and finite spans. If the branch leading to the leaf compared an attribute only from one side, for example "A1 ≤ 5", the resulting span would be infinitely long and could be described as "$-\infty \leq A1 \leq 5$". A finite span results from the branch restricting the attribute from both ends. Having obtained the set of necessary spans for the leaf, a rule is created. It is a simple If/Then type rule. The condition part of the rule is a set of necessary spans as conditions for the leaf, separated by the keyword "and". The statement part of the rule references to the class chosen by the classification leaf. Heaving performed this for every leaf, we can now operate with the set of rules describing the classification tree, instead of the tree itself. At this point, it is possible to begin the ontology building by declaring a concept named "Attribute". This will be the most general concept of all attribute span concepts to follow. It is only a subconcept of the "thing" concept. It is also possible to create a general concept, which will encompass all rule classes obtained from leaves. In the example, the concept name "Risk" was chosen based on the purpose of the classification tree. The subclasses of these two main concepts were assigned by creating a list of unique attributes and classes from the classification tree. The tree used 9 attribute values ("G17", "GPA_UNITS", "IF_Units", "IgG", "Pg1", "Pg2", "Pg1/2", "Svars" and "Vecums") to determine two risk classes ("IVR" and "Vesels"); the resulting hierarchy is shown in Fig. 1.

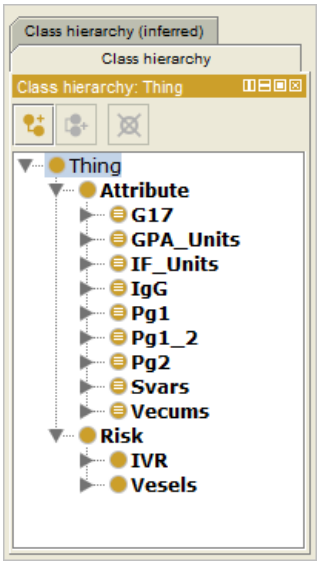

Fig. 1. Base concept ontology.

Before the creation of concepts describing the necessary spans of every rule, the internal hierarchy of the spans is found. The basis for the hierarchy of spans is a simple geometric one. If one span contains another, it is to be of a higher concept than the one it contains. Algorithmically, this is realised by comparing every span to every other span and determining if it is contained in the other. It is contained if all its values are inside the second span. For every S_i in all spans and for every S_j in all spans, if S_i is not the same as S_j and S_j contains S_i, then S_i is a subspan of S_j. However, we are interested only in the abridged hierarchy, so the redundant subspans can be removed. This is done by again poling every span S_i in all spans. If S_i has a subspan S_k and another subspan S_l, and S_l or any one of its sub-spans also contains S_k, S_i may remove S_k as a direct subspan. By doing this for every span, only direct subspan relationships remain, and the minimum direct hierarchy is obtained.

The last step for the spans is to create named concepts for

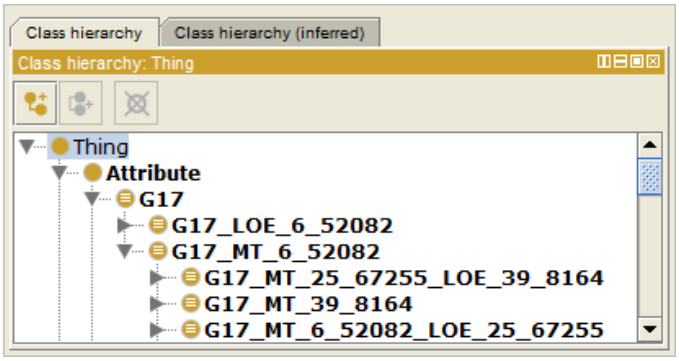

Fig. 2. Generated concept names.

them in the ontology. Names are generated from the span values and directions. For example, the concept name "G17_LOE_6_52082" indicates that the concept belongs to the attribute "G17" and has a value of "less or equal" (LOE) to "6.52082". The letter combination MT indicates the value is "more than". Concepts containing both abbreviations indicate a finite span, which has an indicated beginning and end. Some of the generated names are shown in Fig. 2.

Apart from the name and its place in the hierarchy of attributes, every attribute concept is also given an equality expression. The most abstract concept "G17" has the equality expression "*has_G17 some double*". This means that any individual in the ontology that has a data relationship of the name "has_G17" with some value of the "double" kind can be classified as being an individual of the abstract concept "G17". This is equivalent to defining a span from negative to positive infinity. All other value spans are contained within it. The concept "G17_MT_25_67255_LOE_39_8164" has the following equality expression "*has_G17 some double[> "25.67255"^^double , <= "39.8164"^^double]*". This has the meaning that any individual with the specified data property of a value between these given values will automatically be classified as an individual of this concept.

The last step of this ontology-building algorithm is to add the classification concepts contained in the "Then" part of the generated rules. The generated name for every class begins with the number of the rule, from which it was originally obtained. In our case, it is "R01" to "R25", since there were 25 rules obtained from 25 classification-tree leaves. This is done

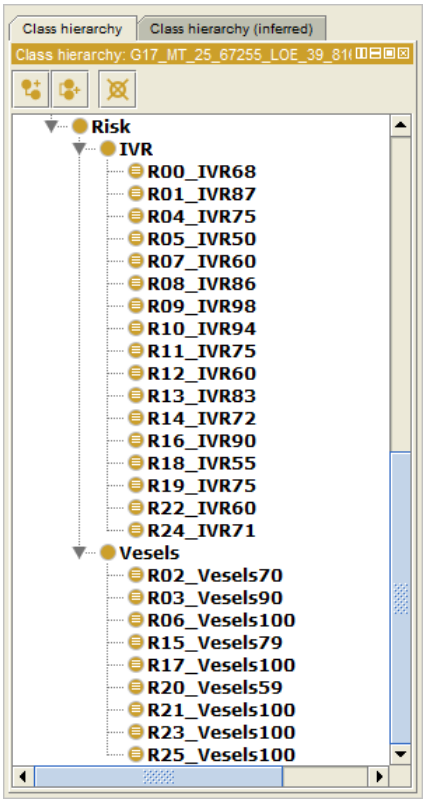

Fig. 3. Generated rule concept names.

to mitigate confusion during ontology reasoning, since there may be many ways to obtain the same classification. After the number of the rule, the name of the assigned class is added. Since many leaves in this particular tree were ambiguous and contained examples of both classes, we added the percentage of the most common class of the leaf to the name of the concept. The generated names of this example can be seen in Fig. 3.

Each of these concepts represents the expression part of the rule it was generated from, and they are also given an equality expression. The equality expression is a simple union of the generated span concepts. For example, the concept "R06_Vesels100" has the following equality expression: "*G17_LOE_6_52082 and IgG_MT_81_03753 and Pg1_MT_38_9425 and Pg2_MT_14_68767 and Svars_MT_69_5*". This means that if an individual was classified to belong to all these concepts at the same time, it would also be an individual of the concept "R06_Vesels100".

IV. Results

The concepts found and defined by the algorithm are fully usable for the classification of data. During the translation from classification tree to the set of rules, the tree data were lost. Tree data are used for finding solutions faster. By only comparing values provided by tree nodes and choosing a single next node, the classification algorithm has to traverse only one branch. This ability is lost in the ontology. However, no loss of information occurred during the process of translating from the classification rules to ontology. This means that the obtained ontology should have the same abilities to classify data as a ruleset. Fig. 4 shows one

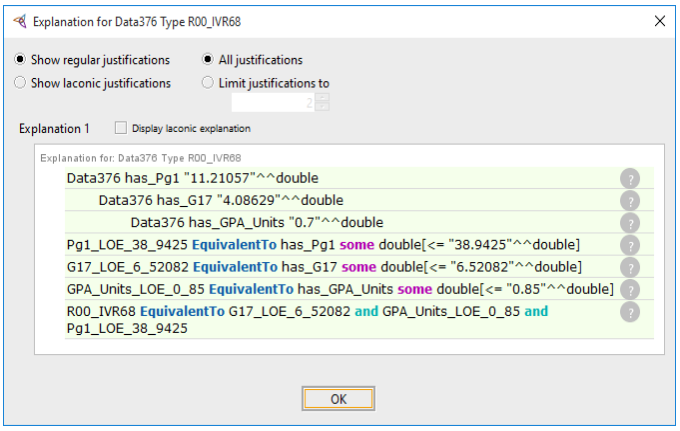

Fig. 4. Reasoning result.

example, where a data point, which was added to the ontology as a new individual and described using the data property names obtained during translation, was correctly classified as belonging to the concept "R00_IVR68" based on the original rule:

```
IF has_G17 some double[<= 6.52082] and
has_Pg1 some double[<= 38.9425] and
has_GPA_Units some double[<= 0.85] THEN
R00_IVR68
```

There were some errors for an earlier version of the method, in which rule names were not uniquely identified by the rule number. When there are multiple leaves in a classification tree with the same outcome (the same class or the same class and class percentage) and they are given the same concept name, they become a description of the same concept. Because of conflicting descriptions of the same concepts in the equality expression, an individual can become defined as having an attribute value both below and above a splitting point. By defining rules as distinct and separate concepts with different names, value crashes were avoided.

The developed approach automatically creates a concept hierarchy for every attribute value span used in the original classification tree. Fig. 5 shows the visualisation of these spans and their hierarchy.

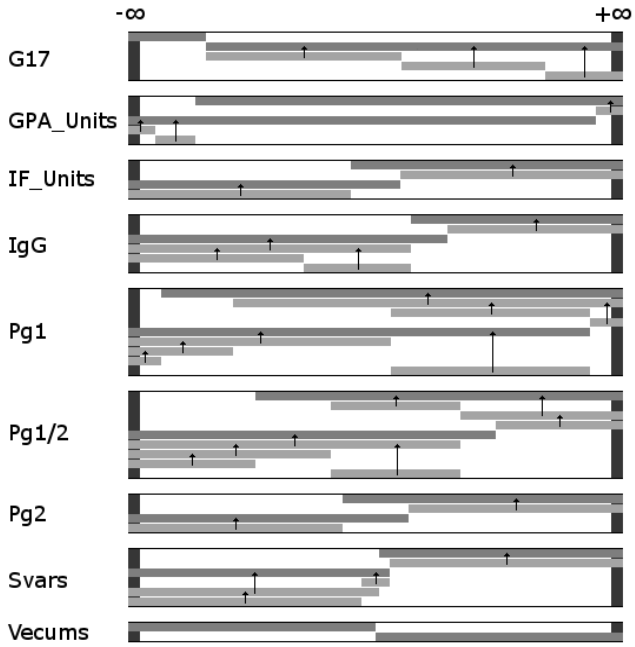

Fig. 5. Visualisation of the value spans.

Every value span is visualised by a grey line. The sizes of the lines are normalised and therefore are only usable for visualisation purposes. The black arrows indicate asserted sub concept relationships. The slightly darker lines indicate top-level spans. Those spans are not sub concepts of other span concepts. The darkest edges left and right indicate infinities. If a line extends all the way left and crosses the dark area, it has no defined endpoint and extends infinitely long.

As we can see from the figure, some attributes are far more complex than others. Attribute "Vecums" has only one splitting point and splits the value space into two distinct regions. An ontology engineer might choose to name these two regions "High" and "Low", making them understandable to humans, while maintaining their classification ability. However, if one looks at the value spans described by most other attributes, mapping every span to a human language might be a far more complex task. One might call the top-level concepts of attribute Pg2 "High" and "Low", while ignoring the overlapping part in the middle. One could do the same for attribute G17 and call the sub concepts of larger span "Low_High", "Mid_High" and "Very_High". The number of intervals and complexity of the ontology depend on the complexity of the dataset, from which the classification tree was created. A simpler dataset with less contradictory data and a clearer relation between data and the classes will result in a simple hierarchy of value spans.

V. Conclusion

This paper has presented a novel approach to ontology building from classification trees created by the C4.5 algorithm. The main novelty of this approach is the creation of concepts, which reflect unique and important data value intervals or spans for every attribute used in the classification process. These unique intervals have been found during the creation of nodes in the C4.5 algorithm; however, they are not readily available without extraction tools. By analysing every leaf and finding the most necessary endpoint values for every interval, and by sorting the found intervals into a hierarchy, an ontology has been built. The built ontology if fully capable of using the data from the classification tree and is able to classify new data point individuals, as long as all necessary data properties are provided. However, during testing it has been found that the reasoning capabilities of the protégé software are very time-consuming. Having more than a dozen individuals describing data point, slows down reasoning to a crawl. Using this tool the ontology does have the capability to classify data, but it is not adequately usable for a large dataset. It can still be usable for a small number of individuals or using a different reasoning tool.

The above ontology building approach might also have some drawbacks. The number of intervals found by the C4.5 algorithms can be large and not intuitively understandable to a human user. The reasonability of the found value intervals can only be evaluated by a domain expert. Maybe the complexity of the span hierarchy is only a perceived one, maybe to an expert the value spans make sense and he will be able to give them appropriate names. One possible solution, which could make these concepts more understandable, is to add familiar concepts as super-concepts to the ontology. Using concepts like "High" and "Low", and manually inserting them into the hierarchy, as the expert sees fit, would group complex concepts beneath simpler ones. Clearly high or low values can be subordinated as such, while others can simply be conceptualised as "ambiguous", which is also a reasonable human concept. As a result, the interval concepts will maintain their complexity and functionality, while being subconcepts to familiar concepts and therefore understandable to humans.

References

[1] A. Nicola, M. Missikoff, R. Navigli, "A software engineering approach to ontology building. Information Systems," vol. 34, issue 2, April 2009, pp. 258–275. http://dx.doi.org/10.1016/j.is.2008.07.002

[2] H. Gorskis, J. Čižovs, "Ontology Building Using Data Mining Techniques," *Information Technology and Management Science*, vol. 15, 2012, pp.183–188. http://dx.doi.org/10.2478/v10313-012-0024-5

[3] I. Polaka, A. Kirshners, H. Gorskis, M. Leja, "The use and modification of decision tree classification algorithm for gastric cancer risk

stratification," *Expert Systems with Applications*. Submitted for publication, 2015.

[4] F. Kharbat, H. Ghalayini, "New Algorithm for Building Ontology from Existing Rules: A Case Study," In: *Information Management and Engineering*. ICIME '09, 2009, pp. 12–16. http://dx.doi.org/10.1109/icime.2009.16

[5] A. Elsayed, S.R. El-beltagy, M. Rafea, O. Hegazy, "Applying data mining for ontology building," In: *Proc. of the 42nd Annual Conf. on Statistics, Computer Science and Operations Research*, 2007.

[6] F. Saïs, R. Thomopoulos, "Ontology-aware prediction from rules: A reconciliation-based approach," *Knowledge-Based Systems*, vol. 67, Sept. 2014, pp. 117–130. http://dx.doi.org/10.1016/j.knosys.2014.05.023

[7] D. Kang, M. Kim, "Propositionalized attribute taxonomies from data for data-driven construction of concise classifiers," *Expert Systems with Applications*, vol. 38, issue 10, 15 September 2011, pp. 12739–12746. http://dx.doi.org/10.1016/j.eswa.2011.04.062

[8] R. Thomopoulos, S. Destercke, B. Charnomordic, L. Johnson, J. Abécassis, "An iterative approach to build relevant ontology-aware data-driven models," *Information Sciences*, vol. 221, 1 Feb. 2013, pp. 452–472. http://dx.doi.org/10.1016/j.ins.2012.09.015

[9] H. Wimmer, R. Rada, "Good versus bad knowledge: Ontology guided evolutionary algorithms," *Expert Systems with Applications*, vol. 42, issue 21, 30 Nov. 2015, pp. 8039–8051. http://dx.doi.org/10.1016/j.eswa.2015.04.064

[10] Y. Kassahun, R. Perrone, E. De Momi, E. Berghöfer, L. Tassi, M.P. Canevini, R. Spreafico, G. Ferrigno, F. Kirchner, "Automatic classification of epilepsy types using ontology-based and genetics-based machine learning" *Artificial Intelligence in Medicine*, vol. 61, issue 2, June 2014, pp. 79–88. http://dx.doi.org/10.1016/j.artmed.2014.03.001

[11] Y. Kuo, A. Lonie, L. Sonenberg, K. Paizis, "Domain ontology driven data mining: a medical case study," *Proc. of the 2007 int.workshop on Domain driven data mining*, 2007, pp. 11–17. http://dx.doi.org/10.1145/1288552.1288554

[12] J. Hastings, D. Magka, C. Batchelor, L. Duan, R. Stevens, M. Ennis, C. Steinbeck, "Structure-based classification and ontology in chemistry," *Journal of Cheminformatics*, 2012. http://dx.doi.org/10.1186/1758-2946-4-8

Henrihs Gorskis is a third-year Doctoral Student majoring in Information Technology at Riga Technical University (RTU). He received his *Mg. sc. ing.* degree in 2013. His research interests include data mining, ontology engineering, evolutionary computing and programming. In 2013 he participated in the AICT conference and also provided a paper on ontology engineering. He is especially fond of the Java programming language and uses it for both work and personal application development.
E-mail: henrihs.gorskis@rtu.lv

Arkady Borisov received his Doctoral degree in Technical Cybernetics from Riga Polytechnic Institute in 1970 and *Dr. habil. sc. comp.* degree in Technical Cybernetics from Taganrog State Radio Engineering University in 1986. He is a Professor of Computer Science at the Faculty of Computer Science and Information Technology, Riga Technical University (Latvia). His research interests include fuzzy sets, fuzzy logic and computational intelligence. He has 235 publications in the field. He has supervised a number of national research grants and participated in the European research project ECLIPS. He is a member of IFSA European Fuzzy System Working Group, Russian Fuzzy System and Soft Computing Association, honorary member of the Scientific Board, member of the Scientific Advisory Board of the Fuzzy Initiative Nordrhein-Westfalen (Dortmund, Germany).
E-mail: arkadijs.borisovs@cs.rtu.lv