Automatic taxonomy building within an object formalism

Petko Valtchev
INRIA Rhône-Alpes
655 av. de l'Europe, 38330 Montbonnot Saint-Martin, France
Petko.Valtchev@inrialpes.fr

Abstract

Within an object knowledge representation formalism, class taxonomies are basic structures used to organize and query the knowledge base. It is therefore profitable to provide an object representation formalism with automatic means for taxonomy inference. Methods deriving classes from sets of unlabeled individuals have been developed within the data analysis and the machine learning fields but these methods use, in general, data descriptions which are less expressive than object models. A taxonomy building strategy based on object proximity estimations is presented in this paper which deals successfully with advanced features of object formalisms like inter-object relationships, multiple perspectives and generic typing. One of its basic tools is the *topological* object proximity model, which is universally applicable on object features and flexible, i.e. it includes a range of concrete dissimilarity functions. The output of the topological function are used by various clustering algorithms to build a hierarchy of object classes, characterized both intensionally and extensionally. The strategy has been implemented by the T-TREE module within the TROPES object knowledge model.

1 Introduction

The object paradigm is shared between fields like *programming languages*, *databases* and *knowledge-based systems*. In the following an *object* is considered exclusively as a representation means, and not as a program unit, which means that it is to be seen as a declarative unit (data structure) within the software model of a real-world domain. Furthermore, our study joins the class/instance approach to the object-based knowledge representation (OBKR). In this framework, objects represent identifiable entities, either real or abstract of the modeled domain, i.e. the individual knowledge about the domain. In contrast, an object *class* summarizes the knowledge about a group of entities and is thus related to a set of objects. Both objects and classes are described in terms of *attributes* which translate the domain entity features. Classes are organized into taxonomies with respect to a generality relationship.

The taxonomic structure allow the knowledge to be organized, stored, explored and completed in an efficient manner and thus represent the basic structure of an object knowledge base. Usually, a taxonomy is devised by the user of the knowledge representation system but it may happen that in particular context no taxonomic structure is specified. Even if only a domain expert can establish the "appropriate" taxonomy for a given object set, the representation system could assist the expert by suggesting various sketches of taxonomies. Thence the importance of the automatic taxonomy building-related topics within the OBKR field. The class inferring task has been studied within the frame of the statistical data analysis under the name of *automatic clustering* [1]. In the *numerical taxonomy*, sets of feature-described individuals are considered and inter-individual proximities are used to build classes, or *clusters*, which are both homogeneous (least intra-class distance) and distinguishable (greatest inter-class distance). The output of a basic clustering method is a set of extensionally described clusters. However, recent trends in the data analysis like *the symbolic object data analysis* [6] or *the formal concept analysis* [16], insist on the inference of an intensional description, usually called *concept*, for each cluster which contains the common features of the member individuals.

The *conceptual clustering* [12], [9] within the machine learning field represents the AI approach towards the automatic class inference. Unlike the conventional clustering methods, the *conceptual* ones put the emphasis on cluster intensional descriptions of various forms : *conjunctive* concepts [12, 2] within a logical formalisms, *probabilistic* concepts [9] or Bayesian clusters [5]. Descriptions are even integrated into the clustering process : the search for homogeneous groups is guided by a description quality criterion which, in general, does not merely limit to proximity. Moreover, the representation formalisms used allow individual descriptions of richer and diverging semantics. The early conceptual clusterers remained tied to attribute-value descriptions.

Lately, there has been some progress in the processing of more complex and structured data mainly within the concept formation paradigm [13, 11]. The integration of component objects in the clustering process relies strongly on the existence of a hierarchical structure of classes (probabilistic concept trees in the sub-sequent terminology). Interesting results on similarity-based clustering on FOL descriptions of complex individuals have been first reported in [2] and further extended in [7]. In [3], a description of a possible way to adapt the method on object-based representations may be found. The solution suggested emphasizes the relational structure of an object-based knowledge base but disregards the available class structures on different object types.

The author advocates for an alternative approach towards clustering of complex objects (i.e. related by attributes to other objects), based on a proximity measure that takes advantage of the existing class taxonomies. Its basic assumption is that two objects are as similar as their common class is specific (far from the taxonomy root class). This helps, when comparing complex objects, establishing a proximity measure on attributes whose values are objects themselves. Once the proximity has been computed for an object set, an arbitrary hierarchical clustering algorithm can be applied to discover possible homogeneous sub-sets. Eventually, each sub-set can be augmented to an ordinary object class by extending it with an intentional description inferred from the member object attributes.

The feasibility of the taxonomy building task within an object model, has been proven in [8] where an application of a standard numerical taxonomy algorithm on numerically-described object is presented. Following in the same direction, we extend here the proximity-based approach towards objects with all kinds of attributes. In that, we use the *topological* model [14] which applies uniformly on object attributes, both single- and multi-valued (sets or lists, see [15]). The model's adaptation on object-valued attributes with multiple taxonomies on the underlying object set is discussed. The extended measure enables successful exploration of the available taxonomic knowledge for the purposes of building new taxonomies. Our approach towards clustering of objects has been implemented within the TROPES object model [10].

The paper starts by a presentation of the object paradigm in knowledge representation (section 2) illustrated by the example of the TROPES knowledge model, which is further used to characterize the taxonomy building task and to clarify the principles of our approach. The basics of the topological dissimilarity are described in section 3.1. The extension of the measure on multiple viewpoints is presented next (section 3.2). Finally, we describe a generic strategy for automatic clustering on complex objects with multiple viewpoints on object components (section 4).

2 Object model

Object-based knowledge representation (OBKR) systems are aimed at building faithful software models of real-world domains. They allow the available knowledge to be represented in both intuitive and efficient manner and its exploration to be highly automated.

A *structured object* is the basic descriptive unit which represents an identifiable domain element: entity, idea, phenomenon and thus carry its relevant characteristics. These features are modeled through valued object attributes or *fields*. Classes represent the generic knowledge, i.e. the knowledge pertaining to a set of domain elements, so a class is related to a set of objects, called its *instances*. Classes characterize their instances by specifying the instance attributes as well as restrictions on attributes values which summarize the features shared by all instances. The set of attributes and restrictions represent the class *intension* and the set of instances – its *extension*. Class and object descriptions are used by the model-related inference mechanisms, like classification or filtering, for querying, reorganizing the knowledge base as well as for integrating new pieces of knowledge.

In the following we present TROPES [10], an object model developed by our research team which admits complex objects and manages multiple viewpoints on them.

2.1 Concepts and primitive types

In TROPES, the objects of a knowledge base (KB) are partitioned into disjoint *concepts*. For example, in a KB modeling a research institute staff and their belongings, there will be concepts like `employee`, `house`, `vehicle`, etc. An object is said to be an *instance* of its concept. Thus, a research assistant, say Joe, will be represented as an object which is an instance of the `employee` concept (see Fig. 1). Concepts are provided with ontological prerogatives on their instances: they define their structure, i.e. the list of attributes, and their identity.

Each attribute is assigned a type, a primitive data type or a concept. Object attributes and their types are defined within the concept. For example, the instances of `employee` have fields like `name` and `town` which are strings, `salary` which is a float and `residence` whose value is an object in `house`. Object-valued attributes, also called *relational*, like `residence`, model relationships of different nature between domain individuals or concepts.

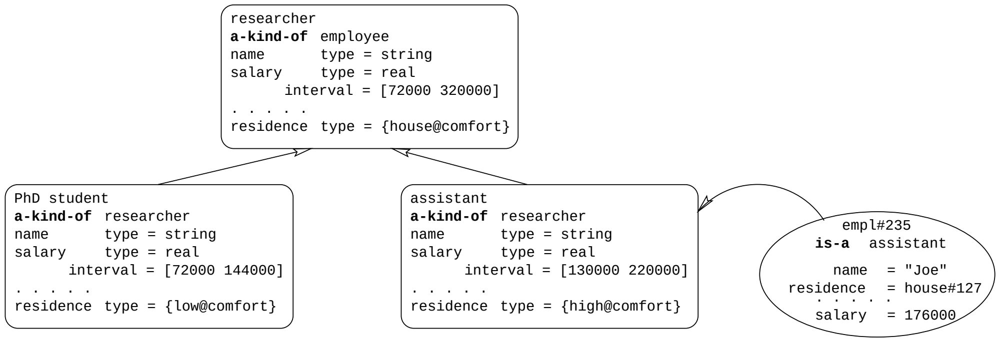

Figure 1: Example of three classes under the `functional` viewpoint on `employee` ordered with respect to specialization, with an instance, `Joe`, attached to its class, `assistant`.

In TROPES, *primitive*, or non-objectified, values are associated with types. Some types are quite standard (e.g. Integers, Boolean), whereas others can be more elaborated (e.g. Date, Nucleic acid sequence). No built-in types exist in the model; instead all types are introduced via an interface, called *abstract data types* (ADT) which is independent from their implementation. The ADT for a type T is declared within a generic category. It includes a set of type management primitives: a value equality predicate ($=_T$), a typing predicate ($\in_T$), an order relationship ($\leq_T$), etc. This allows user-specified types to be easily integrated in object characterizations. Type operations are carried out by the METEO type system, [4], which is connected to TROPES.

2.2 Classes

In TROPES, classes describe sets of concept instances by defining a list of relevant attributes and constraints. Thus, given an object, its field values have to satisfy class constraints in order for the object to belong to the class. Constraints may be of different nature depending on the attribute type. For primitive fields, domain restrictions are usually provided. For example, on Fig. 1 the `salary` of a `researcher` should vary between FF `72000` and `320000`. For relational attributes, the possible values can be restricted to instances of a class under a particular viewpoint or to a set of classes under disjoint viewpoints. For example, `PhD students` may have `residences` which are in the `low comfort` class under the `comfort` viewpoint

Classes are organized into taxonomies, with respect to the *specialization* relationships. The specialization between a couple of classes, a *sub-class* and a *super-class*, is interpreted as follows: (*i*) the instances members of a sub-class are also members of the super-class nd (*ii*) the constraints defined by the sub-class are stronger than the constraints of the super-class. Thus, all `PhD students` are `researchers` too (see Fig. 2) and their salaries vary from `72000` and `144000` which is in the range of employee salaries (`[72000 320000]`).

2.3 Viewpoints and class taxonomies

Concept instances may only be seen under a *viewpoint*. A viewpoint restricts the knowledge manipulated at the same time by specifying a sub-set of visible concept attributes while the remaining attributes stay hidden. Thus, viewpoints represent a means for modeling different contexts by restricting object descriptions to relevant features. For example, employees may be seen both from the perspective of their status within the institute and their spear time activities which leads to the definition of `functional` and `hobby` viewpoints on `employee` respectively. Actually,n instance of `employee` may be in the `chess-player` class under `hobby` and in `PhD student` under `functional`. Clearly, the `salary` attribute is relevant to the `functional` viewpoint but not to `hobby`. Classes are associated with viewpoints in TROPES. Thus, each

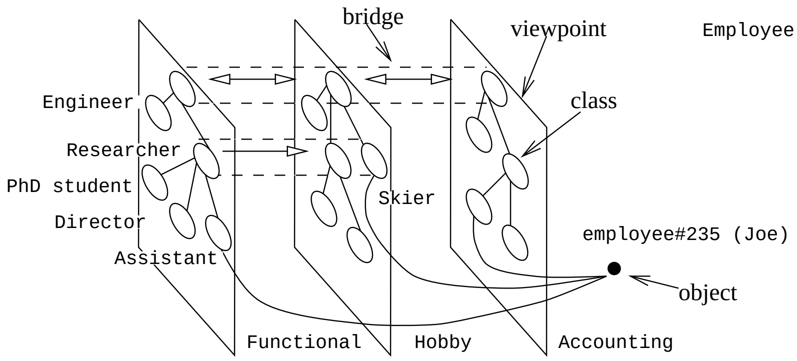

Figure 2: The `employee` concept with three viewpoints: `functional`, `hobby` and `accounting`. The object `Joe` is given with its classes of attachment under each of the viewpoints.

viewpoint may be provided with a class taxonomy. For example, under the `functional` viewpoint, instances of `employee` are partitioned into `engineers` and `researchers`. The researchers are further divided into `directors`, `assistants` and `PhD students`. On Fig. 2, three different viewpoints on the `employee` concept are drawn together with their taxonomies.

It is noteworthy that under each viewpoint, an object is attached to a single most specific class. For example, the employee `Joe` belongs to `assistant` under the `functional` viewpoint and to `skier` under the `hobby` viewpoint (see Fig. 2).

2.4 The taxonomy building task

TROPES represents a powerful environment for expressing and comparing taxonomic knowledge. In fact, while in other models object can only be created within a particular class, in TROPES, objects are instances of concepts and can thus exist independently from classes: an object belongs to different classes under each concept viewpoint. It is thus possible to create a new viewpoint on a concept and then to design a class taxonomy within it.

employee\field	salary	fam-members	town	residence
emp#235	180,000	5	Paris	house#127
emp#174	150,000	3	Lyon	house#56
emp#84	250,000	4	Grenoble	house#14
emp#149	200,000	1	Paris	house#92
emp#61	190,000	4	Grenoble	house#39

Figure 3: Sample set of objects in `employee`.

Imagine, for example, that the institute staff is proposed a new housing insurance product and the possible interests have to be accounted for in the KB. In other words, the objects in `employee` will have to be grouped with respect the (expected) preferences for different insurance formulae.

The gradually cumulated knowledge concerning the new insurance product will have to be gathered into a particular viewpoint on `employee`. Thus, a first task will be to create the new viewpoint, call it `insurance`, with all the relevant fields: `salary` which specifies the amount of annual income of the employee, `fam-members`, i.e. the size of the employee's family, `town` and `residence`. On Fig. 3, a set of instances of `employee` is given in a object $\times$ attributes like table. Moreover, the values of the `residence` attribute, are viewed under two different viewpoints which model two different aspects to take into account when the employee potential interest is evaluated. The first one, `comfort`, concerns the living quality of the residence while the second viewpoint, `security`, reflects the possible risks for the house. On Fig. 4, the taxonomies of both viewpoints are given, each of the residences from Fig. 3 is linked to its classes of attachment.

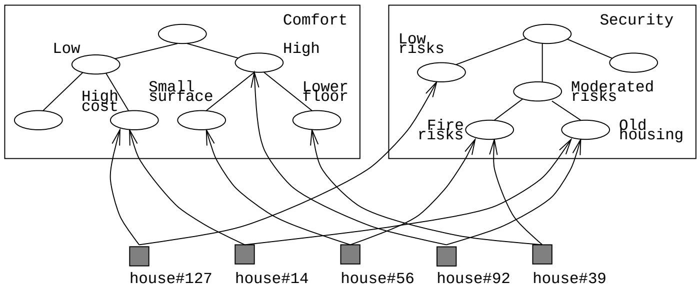

Figure 4: The instances of `house` associated to employees on Fig. 3 are given with their classes within the taxonomies of the `comfort` and `security` viewpoints.

A new taxonomy under `insurance` is necessary in order to answer queries about preferences of the existing employees and for predicting the preferences of a new employee. The taxonomy design task can be assisted by an automatic module. The module should however be provided with powerful tools for processing all the available knowledge about the objects. It may implement, for instance, an automatic clustering algorithm based on inter-object proximities (see [8]). For that reason, a proximity measure is needed to deal with different object description components: attribute values belonging to an ADT or to a concept, inter-object relations, taxonomies on related objects. For example, the proximity of a couple of employees under `insurance` viewpoint, includes the proximity of the respective residences which is, in turn, a combination of their proximities within each of the contexts described by `comfort` and `security`.

In [14], we proposed a dissimilarity model, called *topological*, to deal with abstract data types in TROPES. The topological model may be easily adapted to complex objects with single viewpoints on relational attributes as it is shown in the following section.

3 Topological dissimilarity

A proximity model defined on a set of objects Ω described by features $\{a_1, .., a_n\}$ includes two functions: a field level function δ_i for each a_i and a global function d. While δ_i are computed on filed values, d aggregates the values of each δ_i into a single global value. Furthermore, δ_i usually depends on the respective field type, and d – on the nature of the data and the problem to be solved. Thus, for a generic clustering module, it is only sensible to give a class of functions d.

3.1 Basic model

The *topological dissimilarity model* focuses on the way the δ_i are computed [14]. It is based on the most faithful exploration of an object attribute semantics in the comparison of the attribute values. Thus attributes are considered with their types, which are assumed to indicate a value domain and a hierarchy of meaningful sub-domains. Within the domain, the proximities between members reflects their mutual position in the hierarchy. Thus, the *field level topological measure* δ^t on a domain is based on the shortest path distance in the graph of the hierarchy.

Formally, let be given a domain D, e.g. an object concept or an integer type, and C a set of sub-domains, object classes or integer intervals, hierarchically organized. Let also e_1 and e_2 be a couple of elements in D and let C^* denote the set of sub-domains which include both e_1 and e_2. Then the topological distance between e_1 and e_2 is computed as the smallest number of (contiguous) intermediate sub-domains:

$$\delta^t(e_1, e_2) = \min_{c \in C^*}[dist(e_1, c) + dist(e_2, c)]$$

where $dist(e, c)$ is the number of intermediate sub-domains between an element e and a sub-domain c, inclusive c. The normalized function is:

$$\overline{\delta}^t(e_1, e_2) = \frac{\delta^t(e_1, e_2)}{\max_{o_1, o_2 \in C} \delta^t(o_1, o_2)}$$

The above definition corresponds, in case of a discrete structure on D, to the shortest path distance within a circuit-free graph. As far as the formal properties of δ^t are concerned, it is a valid dissimilarity index and, in case D is a TROPES concept, – a tree distance.

The model allows all object attributes to be dealt with in a homogeneous manner. Thus it represents a universal means for comparing objects with respect to their field values. Despite this universal applicability, the behavior of the topological measure on usual data types, like nominal, ordinal or continuous, is quite intuitive and its instantiations on those data types are well known functions (see [14]).

The way $\overline{\delta}^t$ behaves on concepts is illustrated on the taxonomy of the `comfort` viewpoint on `house` (see Fig. 4. On Fig. 5 the values of $\overline{\delta}$ are given.

	house#56	house#14	house#92	house#39
house#127	1	1 / 3	5 / 6	1
house#56		1	1 / 2	2 / 3
house#14			5 / 6	1
house#92				1 / 2

Figure 5: The values of $\overline{\delta}^t$ on the `comfort` viewpoint.

The *object level function* d^t is defined on TROPES concept Cpt viewed under viewpoint Vp with n visible attributes $\{a_1, .., a_n\}$. For a couple of individuals, say e and e', given with their corresponding field values $e.a_i$ and $e'.a_i$ $(i = 1 \ldots n)$:

$$d^t{}_{Cpt}(e, e') = Aggr_{i=1}^{n}\ \overline{\delta}_i^t(e.a_i, e'.a_i) \tag{1}$$

where $\overline{\delta}_i^t$ is the topological dissimilarity on the i-th attribute of Cpt and $Aggr$ is a generic function. The exact form of $Aggr$ may be chosen between typical function like Euclidean distance, weighted City block, etc..

The topological distance d^t processes successfully the relational structure of the knowledge. In fact, when comparing two objects, both the local information, i.e. the primitive field values, and the relational information, i.e. the object-valued attributes, are taken into account.

It is noteworthy that, unlike d^t, when applied on a concept δ^t ignores the inner structure of objects and considers them as if they were atomic values. In this respect δ^t is an approximation of the effective dissimilarity measure d_i computed on the viewpoint Vp'. In addition, δ^t allows the relational structure to be explored reasonably, that is at a depth one. In other terms, objects which are linked by a chain of more than one attributes to a given object will no more be processed. For example, if the house should be represented with each of its rooms, and those rooms be instances of the `room` concept, when computing d^t on `employee`, the respective objects will be ignored. In fact, δ^t will not consider the inner structure of house objects and thus their rooms will remain hidden. This may seem to be a rather rough approximation, however, δ^t is a directly, and universally, available metrics, whereas d_i may not be known for some a_i.

3.2 Multi-viewpoint dissimilarity

Consider an object-valued attribute a_i defined in concept Cpt whose domain is a concept Cpt'. Let Vp be a viewpoint on Cpt and let a_i be visible under Vp. Furthermore, let $Vp'_1, .., Vp'_k$ be the viewpoints on Cpt' relevant to Vp. In the example of the previous section, Vp is `insurance`, Cpt is `employee`, a_i is `residence` and Cpt' with $Vp'_1, .., Vp'_k$ are respectively `house` concept with the `comfort` and `security` viewpoints ($k = 2$).

To measure the proximity of a e_1 and e_2 under Vp with respect to their field values for a_i, say $o_1 = e_1.a_i$ and $o_2 = e_2.a_i$ viewed under $\{Vp'_1, .., Vp'_k\}$, it is necessary to: (i) compute δ_i upon the entire set $\{Vp'_1, .., Vp'_k\}$, (ii) process each viewpoint independently but combine the different values in a single one, (iii) explore the taxonomy on each Vp'_j in a similar way as in the single viewpoint case. In sum, $\delta(o_1, o_2)$ has to be a (weighted) combination of $\overline{\delta}_j^t(o_1, o_2)$ on each Vp'_j:

$$\delta_m^t(o_1, o_2) = Aggr'\;{}_{j=1}^{k}\overline{\delta}_j^t(o_1, o_2) \tag{2}$$

$Aggr'$ is again a generic function. For its exact form, we suggest a normalized linear combination, so that $\overline{\delta}^t{}_m = \delta^t{}_m$. On Fig.6.a, the values of $\delta^t{}_m$ on the `residence` under the `insurance` viewpoint are given where $Aggr'$ is the mean. On Fig.6.b, the respective values of d^t on `insurance` viewpoint are given whereby $Aggr$ is a weighted City block with weights $1, 1, 1$ and $0, 5$ for `salary`, `fam-members`, `residence` and `town`respectively.

Taking δ^t in $\delta^t{}_m$ is not the only possible solution. In fact, other functions are imaginable, computed on field values: either locally, that is on each viewpoint, or globally, i.e. on a subset of concept attributes, visible under at least one viewpoint. The advantages

House	house#56	house#14	house#92	house#39
house#127	1	0,67	0,92	1
house#56		0,9	0,65	0,53
house#14			0,62	0,9
house#92				0,65

a)

Employee	emp#174	emp#84	emp#149	emp#61
emp#235	2,3	2,61	2,11	1,85
emp#174		2,65	2,15	1,68
emp#84			1,86	2
emp#149				2

b)

Figure 6: The values of ${\delta^t}_m$ for the `house` set given on Fig. 4 (a) and the respective values of d^t on `employee` set of Fig. 3 (b).

of solution we proposed are multi-fold. First, it is simpler to compute since it requires neither attribute duplication manage nor two level weight assignment (viewpoint and attribute). Then, $\overline{\delta}^t_m$ seems to be of higher intelligibility than any measure based on attribute values. Last, but not least, ${\overline{\delta}^t}_m$ is always available on a set of viewpoints provided with taxonomies.

4 Clustering strategy

Enhanced with the topological dissimilarity model, an automatic clustering module may successfully process complex object descriptions. Nevertheless, the module should establish a dialogue with the user (expert, taxonomist) in order to acquire the necessary parameters of the classification process: concept, viewpoint, attributes, classification method, etc. The choice of both an appropriate method and a concrete proximity function is a non-trivial task. It depends on the aims of the clustering and on the nature of the data processed. Thus, not only the topological dissimilarity includes a family of effective functions, but also the user is granted the possibility to replace it with another measure. In sum, when processing a set of objects, the user has the choice between: (i) apply d^t directly since it processes uniformly object fields, (ii) choose a more elaborate effective functions δ^t for some attributes and fine-tune d^t by feeding-up attribute weights or (iii) simply replace d^t with a more appropriate measure.

Next, the application of the topological dissimilarity on an object set requires the existence of a taxonomy for each viewpoint of each object-valued attribute. This means that clustering objects of concept Cpt may involve clustering on concepts related to Cpt. Thus, the taxonomy building module carries out a postfix search loop in the graph of the knowledge base. The graph is virtually composed by concepts as vertices and relational attributes as edges.

The above described strategy has been implemented in T-TREE the taxonomy building module of TROPES. T-TREE includes the implementation of some classical clustering methods like *single linkage*. The available proximity functions represent widely used members of the Minkowski metrics family like Euclidean distance or City block.

For a concept Cpt with instances $\{o_i\}_{i=1,m}$ seen under viewpoint Vp with visible attributes $\{a_i\}_{i=1,n}$, the overall loop of T-TREE may be described as follows.

CLASSIFY(Vp)

choice of a classification method M
choice of proximity measure d
choice of set of attribute weights $\{\omega_i\}_{i=1,n}$
for each $\{a_i\}$
  if the domain of $\{a_i\}$ a concept Cpt' with k viewpoints then
    choice of set of viewpoint weights $\{\lambda_i\}_{i=1,k}$
    for each viewpoint Vp'_l the concept Cpt' concerned
      if no taxonomy exists on Vp'_l then
        CLASSIFY *(Vp'_l)*
M($\{o_j\}_{j=1,m}$, $\{a_i\}_{i=1,n}$, $\{\omega_i\}_{i=1,n}$, D)

Fig.7 shows the taxonomy which is obtained with a single linkage method and the topological dissimilarity specified on the right of Fig.6.b.

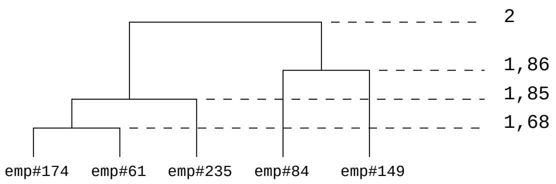

Figure 7: A taxonomy on the sample of `employee` obtained by an agglomerative hierarchical clustering algorithm of single-linkage type.

The discovered classes are provided with intensional descriptions. The exact way of computing those descriptions is beyond the scope of this paper. Once completely characterized the taxonomy is submitted to the user's validation. In case the taxonomy does not fit user's expectations but only minor changes are necessary, they may be carried out manually. Otherwise, the classification process is to be restarted with new parameters. A validated taxonomy is immediately integrated into the knowledge base.

5 Conclusion

We presented a possible way to carry automatic classification directly within an advanced object knowledge representation model. In particular, the above described strategy fits to environments of multiple taxonomies on the same object set. The knowledge encoded in those taxonomies is successfully explored for classification purposes by the particular model of object metric, the *topological* dissimilarity. The *topological* dissimilarity, previously defined and extended here to multiple viewpoints, considers objects with respect to their localization within each of the available taxonomies.

The topological dissimilarity has been integrated into T-TREE the taxonomy building module of the TROPES knowledge model. The generic automatic classification strategy implemented by the module allows the constitution of several taxonomies on different, inter-related concepts in the knowledge base. It is thus an example of how previously built taxonomies may be used to infer new taxonomies.

References

[1] M. Anderberg. *Cluster analysis for applications.* Academic Press, 1973.

[2] G. Bisson. Conceptual clustering in a first order logic representation. In *Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria*, pages 458–462, 1992.

[3] G. Bisson. Why and how to define a similarity measure for object-based representation systems. In N.J.I. Mars, editor, *Towards Very Large Knowledge Bases*, pages 236–246, Amsterdam, 1995. IOS Press.

[4] C. Capponi. *Identification et exploitation des types dans un modèle de connaissances à objets.* Thèse d'informatique, Université Joseph Fourier, Grenoble (FR), 1995.

[5] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *Proceedings of the 5th Internatinal Conference on Machine Learning, Ann Arbor, MI*, pages 54–56, 1988.

[6] E. Diday. The symbolic approach in clustering and related methods of data analysis: tha basic choices. In H. Bock, editor, *Proceedings of the First IFCS Conference*, 1987.

[7] W. Emde and D Wettschereck. Relational instance based learning. In L. Saitta, editor, *Machine Learning - Proceedings 13th International Conference on Machine Learning*, pages 122–130. Morgan Kaufmann Publishers, 1996.

[8] J. Euzenat. Brief overview of t-tree: the tropes taxonomy building tool. In *Proceedings of the 4th ASIS SIG/CR classification research workshop, Columbus (OH US)*, pages 69–87, 1993.

[9] D.H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] INRIA Rhône-Alpes, Grenoble (FR). *Tropes 1.0 reference manual*, 1995.

[11] A. Ketterlin, P Gancarski, and J.J. Korczak. Hierarchical clustering of composite objects with variable number of components. In D. H. Fisher and P. Lenz, editors, *Proceedings of the 5th International Workshop on Artificial Intelligence and Statistics, Fort Lauerdale, FL, USA*, 1995.

[12] R. Michalski and R. Stepp. *Machine learning: an Artificial Intelligence approach*, volume I, chapter Learning from observation: conceptual clustering, pages 331–363. Tioga publishing company, Palo Alto (CA US), 1983.

[13] K. Thompson and P. Langley. *Knowledge and experience in unsupervised learning*, chapter Concept formation in structured domains, pages 127–161. Morgan Kaufman, San Mateo (CA US), 1991.

[14] P. Valtchev and J. Euzenat. Classification of concepts through products of concepts and abstract data types. In Y. Lechevallier E. Diday and Dr O. Opitz, editors, *Ordinal and symbolic data analysis*, pages 3–12, Heildelberg (DE), 1996. Springer Verlag.

[15] P. Valtchev and J. Euzenat. Dissimilarity measure for collections of objects and values. In P. Coen X. Liu and M. Berthold, editors, *Proceedings of the 2nd Symposium on Intelligent Data Analysis.*, volume 1280 of *Lecture Notes in Computer Science*, pages 259–272, 1997.

[16] R. Wille. Restructuring the lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, pages 445–470, Dordrecht-Boston, 1982. Reidel.