Extracting Concept Maps With Clouds

Francisco Câmara Pereira, Ana Cristina Oliveira and Amílcar Cardoso

{camara, amilcar}@dei.uc.pt ana@student.dei.uc.pt
Centro de Informática e Sistemas da Universidade de Coimbra (CISUC)
Polo II – Pinhal de Marrocos 3030 Coimbra
Portugal

Key Words: Machine Learning, Concept Maps, Cognitive Modeling, Human-Computer Interaction

Abstract.

This paper presents Clouds, a program that aims to extract structural domain knowledge from the user. This extraction consists of the use of three different algorithms: one for choosing the concept to work with; and two, based on inductive learning, for suggesting new concepts and relations. In a first phase, the user must "teach" the computer with some important concepts from the domain that he wants to transmit. Then, gradually, in a simple dialogue, Clouds asks questions resulting from the learnt hypothesis.
Each concept is an element of a map, called concept map, which consists of a graph with concepts on nodes, and relations on arcs.
This work is an essay on the application of machine learning on knowledge extraction and is part of a wider project, named Dr. Divago, which will get from Clouds the help to build complete and coherent concept maps that it needs to work efficiently. Apart from this, we believe Clouds' scope can be extended to many different areas, namely natural language processing, human-computer dialogue and intelligent tutoring.

1. Introduction

Representing a domain of knowledge in a computer is usually a highly complex task, involving the knowledge representation itself, domain-specific particularities, problems in extracting knowledge from people, among others. This problem becomes bigger when there is dynamic introduction of knowledge from the user, who is not necessarily a computer expert. According to (Novak and Gowin, 1984), a formalism named "concept map" is extremely powerful for education and communication in humans. We think these simple nets may also be useful to computers, more specifically in tasks where structural domain knowledge is important.

The goal of the project presented in this paper is to be able to represent knowledge from a domain, by building its concept map. Our concept maps are graphs in which nodes are concepts and arcs are relations.
Although these maps can represent few more than structural knowledge of a domain, this seems sufficient for our purposes. Its inherent freedom (the user can assert any concept or relation in the map) brings us the expressiveness and generality we need for Dr. Divago (Pereira and Cardoso, 1999), the major framework in which Clouds is integrated.
The building of these maps results from a symbiosis between the computer and the user, each having a different role. Initially, the user "teaches" the program with some basic concepts of the domain. Afterwards, Clouds asks for the concepts it wants the user to talk about and, after a number of observations introduced, it starts improving its own performance, by asking more specific questions. It uses two inductive learning[1] algorithms to accomplish this, each of them focusing on two different points of view on the concepts: its categorization and its definition by context. This shall be clarified later in this paper.
In section 2, we present our main motivations about this work. Clouds is presented in more detail in the following sections: Representation (section 3), Selection of concepts (section 4), Learning (section 5) and Interacting with the user (section 6). Then, we will draw some conclusions from this work (section 7).

2. Motivation

Soon after the first steps of the development of Dr. Divago, a system that relies on Case Based Reasoning and Metaphor Theory (Veale and Keane, 1994) to search for creative solutions, we felt the need for the automatization of a difficult and exhausting task: the construction of concept networks. We then decided to build a module, to which we named Clouds, to help us on this task, whose goal is to build concept maps from observations of the user, gradually improving its performance. After a sufficient number of observations, it is expected to guide the user in the transmission of knowledge, by asking questions and proposing relations between concepts.

3. Representation and Ontological Issues

"According to the theory-based model (Murphy and Medin, 1985), concepts are embedded in theoretical understanding of the world. While a prototype representation of the concept *bird* would consist of a list of unconnected attributes, the theory-based representation would also represent theoretical knowledge about the relation of each attribute to others in a complex network of causal and explanatory links, represented

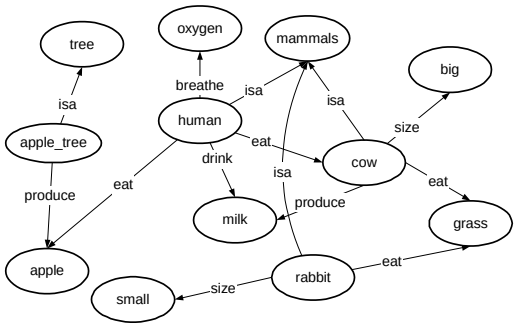
[1] We recommend (Muggleton, 1992) for an introductory course on Inductive Logic Programming.

in a structure frame or schema. *Birds* have *wings* in order to *fly*, which allows them to rest in *trees*, which they do to escape *predation*, and so forth. According to this view, objects are categorized in the class which best explains the pattern of attributes which they possess (Rips, 1999). " (Hampton, 1998).

Following these ideas, we understood that we needed a formalism that could transmit to some extent this "concept of concept". And we found it in Novak's (Novak and Gowin, 1984) concept maps. A concept map is a graph in which nodes are concepts and arcs relations (see figure 1). It doesn't follow any strong constraints, being its main concern to reflect natural association of ideas.

Every relation in a Clouds' concept map consists on Prolog predicates with 2 arguments, which means we only accept binary relations. We believe its extension to n-arity will be straightforward, and the only reason for delaying this improvement is to prevent ourselves from losing focus.

Since each concept is entirely defined by its connections to others, our knowledge base will consist on a large set of prolog binary predicates.



isa(rabbit,mammals).	eat(human, cow).
isa(cow, mammals).	eat(cow, grass).
isa(human, mammals).	eat(rabbit, grass)
isa(apple_tree, tree).	eat(human, apple).
produce(apple_tree, apple)	drink(human, milk).
produce(cow, milk).	size(rabbit, small).
breathe(human, oxygen).	size(cow, big).

Fig. 1. A small concept map and its representation in Clouds

As can be seen, every concept is defined with respect to others, so it is important to state a rather solid base of ground facts: the ontological base of Clouds. This set of primitive concepts is composed by a default isa-tree in which the first levels are the same as the ones used in the WebKB project (Craven et al, 1998), as shown below.

```
isa(entity, something).
isa(situation, something).
isa(state, situation).
isa(process, situation).
isa(temporal_entity, entity).
isa(spatial_entity, entity).
isa(information_entity, entity ).
```

Fig. 2. The first three levels (Prolog representation)

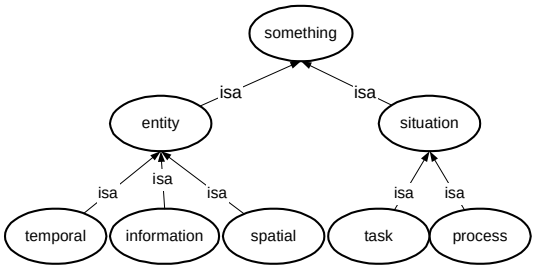


Fig. 3. The first three levels (graph representation)

Every time a new concept is presented to the program, it asks for its isa-relation to an existing concept in that tree, and therefore integrates it in the tree. As can be easily inferred from this, every concept in the map must be a descendant of a node from the default ontological base. This is the central assumption for one of the inductive learning algorithms used by Clouds, described below. The resulting categorization will allow the program to clusterize concepts/relations according to the observations given so far.

4. Selecting Concepts

One of the first problems that occur, when transmitting this kind of knowledge, is that of "what to say next?". It is interesting the ease how one tends to focus too much on a specific "area" of concepts, specially when trying to introduce large quantities of unorganized knowledge. To accomplish as much completeness as one can ask for, we need to "walk" through different points of view, in a more far-reaching way, sometimes returning to the same point (it is common to forget something that later, after visiting other concepts, is obvious). Choosing a proper sequence of concepts to think about will be the halfway for a correct concept map construction.

When reading Hostadter's (1996) "Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought", the notion of "scout" appeared gradually as fitting the place for solving this problem. In this book, "a scout namely looks at a potential action and tries to estimate its promise; the only kind of effect it

can have is to create one or more (…) scouts or effectors – to follow up on its findings".

Inspired on this notion of scouts, we developed the algorithm to find, for a given moment, for the set of concepts to work with. This algorithm is based on a competitive search for the most relevant, but not fully explained concepts of the map. We consider relevance of a node, in respect to another, as the number of separate paths between them. Furthermore, we consider "absolute relevance" of a node as the sum of its relative relevances. It is an estimation of this value that the scouts try to maximize. They jump from concept to concept until finding a promising position. Each jump consists of creating scouts in neighbor concepts, surviving (with major probabilities) only the ones that are conceptually more relevant (in relation to the original scout).

The search starts by seeding scouts in some central concepts, the ones that are the most referrer or referent. Using two kinds of memory, named "long term memory" (composed by the concepts the user explicitly refuses to talk about anymore – thus already *explained*) and "short term memory" (the *n* last selected ones), Clouds eliminates some undesirable choices. Here goes an overview of the algorithm:

1- Launch a small number of scouts(usually less than 4), starting from a random set of central concepts.
2 - If any concept is a leaf (i.e., it has no output relations) and it is not present in long or short-term memories, keep it and check whether there is any more alive scouts to search (if not, stop).
2 - Spread new scouts through their neighbourhood (if any of the current scouts is an ilegal leaf, spread backwards).
3 - Rank the new scouts according to relevance (relative to the previous generation).
4 - Choose randomly the surviving ones according to rank distribution.
5 - Go back to 2.

Fig. 4. The algorithm for choosing the next concepts to work with.

5. Learning

In Clouds, the role of the two inductive learning algorithms is to improve its performance in the guidance of the user. More than getting what a concept is, it is important to understand what makes a given concept be related to another. This, we believe, is the key to improve gradually the capabilities of Clouds, allowing it to predict the existence of new concepts and relations.

As can overcome from the discussion about Knowledge Representation in Clouds, concepts are not explicitly declared, being implicitly described through their relations, so the task of learning concepts turns into learning its relations. The next two sections will describe these algorithms.

5.1. Category divide and conquer

The first algorithm used in the program aims at finding, for each relation, for pairs of categories that it typically links (e.g. “trees *typically* produce fruit”). In order to do it, Clouds relies on the ontological isa-tree to find generalizations and specializations. Generalizing corresponds to get categories up in the tree, and specializing the opposite. The algorithm is very simple: each time it receives an observation, it detects the categories of concepts involved. In doing so, Clouds extracts the ontological path from the most general concept “something” to the specific concept of the observation.

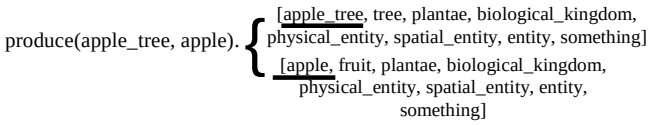


Fig. 5. Ontological paths of both arguments of the relation “produce(apple_tree, apple)”.

The leftmost intersection of both ontological paths of a relation (one for argument) with the ones of other relations (from already user-asserted observations) yields a generalization.

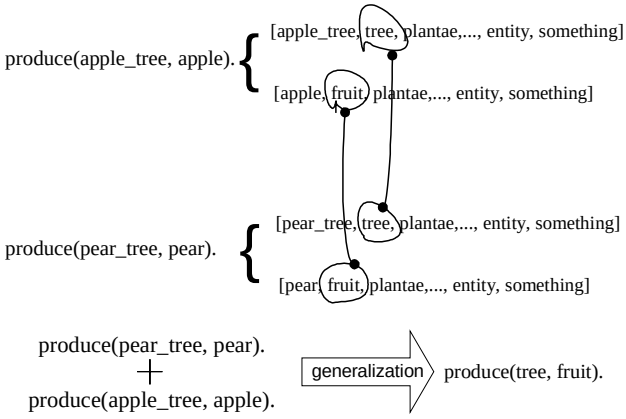


Fig. 6. Intersecting the paths of “produce(pear_tree, pear)” and “produce(apple_tree, apple)”, and selecting the leftmost elements (tree and fruit), yields a generalization “produce(tree, fruit)”.

The specialization occurs when a negative example is given. In this case, Clouds searches down in the tree for the most general specializations that “avoid” this new observation. The result of this, conversely to generalization, is to split the space into new hypothesis. This divide and conquer results in a number of binary predicates that represent the pairs of categories of the arguments that cover the positive examples and avoid the negative ones. These predicates are helpful mainly in the first phase of knowledge introduction, by asking the user questions like “what is the tree that

produces bananas?" after the observations, for example of "monkeys eat bananas" and "banana isa fruit".

5.2. Inductive Logic Programming

Inspired on studies from Inductive Logic Programming (Muggleton and Feng, 1990; Quinlan, 1990), we implemented a relation explanation generator that concentrates on the context of each argument. The relations each argument has with other concepts, down to a predefined maximum depth uniquely define this context.

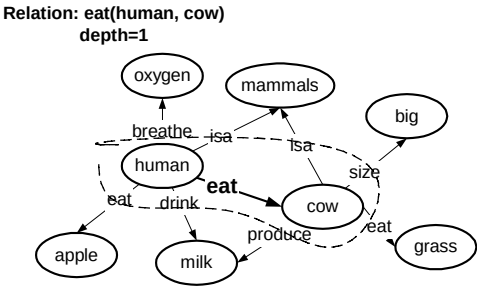


Fig. 7. The relation "eat(human, cow)" and its contexts of depth=1 (relations immediately linked to its arguments).

This method enables Clouds to understand each relation in terms of what usually characterizes its context, opening the way for applying deduction and abduction.
The result of this algorithm has the form of Prolog expressions, as in the example shown in fig. 8

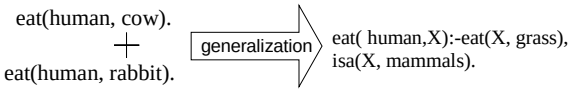


Fig. 8. Joining the contexts of "eat(human, cow)" and "eat(human, rabbit)", leads to a generalization.

Generalization may happen in two ways: universal quantification and dropping of a term.
Universal quantification occurs when, in a given clause, one atom representing a concept becomes a variable representing a set of concepts.

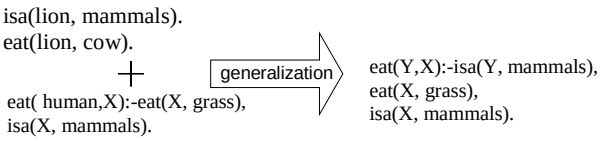


Fig. 9. . Universal quantification of the first argument of eat/2.

The dropping of a term occurs whenever a new observation reflects an over-specialization of the clause (e.g. after saying that "Lion eat ostrich" and that an "ostrich is a bird that eats grass", the term "isa(X, mammals)" in figure 8 will be dropped

Specialization is a more complicated procedure in this algorithm. To avoid loosing information or covering negative examples, a deeper examination of the clauses in question must be done. The search for specificities that can safely differentiate a positive context from a negative one are not always clear. The method we apply here is probably not the most elegant, but surely the most reliable: when a negative example is given, Clouds selects the faulty clauses (the clauses that cover the example) and finds all predicates already asserted as positives and that are also covered by the same clauses. Now, the generation of the new hypotheses is made by adding to the previous faulty clauses terms that are satisfied by the positive examples, but not by the negative ones. Therefore, the output of this sometimes corresponds to a big number of new clauses that cluster the space of concepts. This is, in the end, what we intend Clouds to do.

6. Interacting with the User

As stated in the beginning of this paper, the primary goal of Clouds is to guide the user in the creation of concept maps. How can it do it? First of all, the method for choosing the concept to explore is itself, as we could see in experiments, a helpful mechanism. There are several reasons for this: Clouds chooses the concepts that are, at a given moment, the ones that seem the most relevant (according to the conceptual relevance heuristics). And it will get any of these concepts back again each time long and short time memory allows it. This means that some concepts are periodically re-inspected, each time probably with new data that can bring the user to other points of view on the concept.

The second mechanism used by Clouds is to apply the results of the first inductive learning algorithm, in which, as said above, there is a definition of each relation in terms of the categories of each of its arguments. This information becomes important in suggesting relations in the first phase of map construction. After selecting the concept to work with, Clouds checks if there is any relation whose categories match with the concept's (Fig. 10).

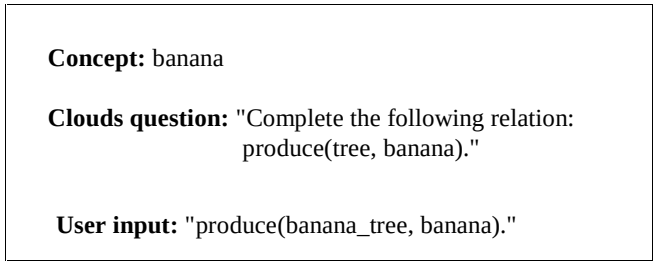


Fig. 10. After matching the concept "banana" with the generalization of fig. 6, Clouds asks for which is/are the descendants of the concept "tree" that turn the relation into a positive example.

The reason why this is very important in the beginning of the process concerns to the fact that, in the early stages of map construction, Clouds hasn't got yet sufficient information to behave effectively in the other inductive learning algorithm.
Finally, the ILP algorithm becomes important to "strengthen" the map in that it searches for relations between pairs of already existing concepts, either with deduction or abduction.
At this moment, deduction is used only when a concept is investigated, but it surely could be applied at any time in the program execution, depending only on the existence of clauses that cover examples that were not already asserted by the user. In spite of this, Clouds presents its deductions only when asking new observations to the user, in a list of "suspected relations". At this point, the user can assert one of them or even say it is false (triggering the specialization procedures).

In the figure 11, we can see an excerpt of a session in the domain of Biology:

```
-->Complete the relation
        have(hippopotamus, teeth): y.

I already know that:
isa(hippopotamus,ruminantia)
have(hippopotamus, teeth)

I suspect that:
property(hippopotamus,friendly)
Can you give more observations related to the concept
hippopotamus ?
-->property(hippopotamus, big).
-->Is it true that eat(hippopotamus,water_plants)? y.
-->Is it true that have(hippopotamus,fin)? n.
-->Is it true that have(hippopotamus,gill)? n.
-->Is it true that live_in(hippopotamus,water)? y.
-->Is it true that isa(big,behaviour)? n.
-->live_in(hippopotamus, land).
-->Is it true that isa(hippopotamus,reptilia)? n.
-->property(hippopotamus, big).
I already know that!
-->property(hippopotamus, heavy).
-->no.
```

Fig. 11. An excerpt of session 8 of the experiment in Biology domain.

After selecting the concept (*hippopotamus)* and matching it with the relation *have*(*mammals*, *teeth*), Clouds starts by asking to complete the relation. Then, it shows its knowledge and suspicions. The user asserts a new fact (*property*(*hippopotamus*, *big*)), after which an abduction mechanism (Kakas et al, 1998) asks about the truth of the relation *eat*(*hippopotamus*, *water_plants*). Its positive answer generates some more questions. When a repeated assertion (*property*(*hippopotamus*, *big*)) is given, the program shows a warning. To pass to the next concept, the user says "no", and a new cycle begins.

7. Conclusions

Clouds was initially designed to be part of our Dr. Divago project, in which we are starting to apply it, and is now also part of another framework, along with TextStorm, where these two modules cooperate to extract knowledge directly from text. In both frameworks, we have noted that, even in the worst case, in which it suggests only wrong relations, it is useful because it guides us on the selection of the concepts to work with (after some dozens of concepts, we tend to loose focus without Clouds). In the better cases, it surprised us with some interesting conclusions (e.g. in the Biology domain, it considered the "eat" relation transitive, therefore if a *eats* b and b *eats* c, then a *eats* c – this is not quite true, obviously, but it showed us an idea of the incomplete way we defined that relation) and fertile suggestions.

In which respects to applications of Clouds, it seems an interesting tool to be included in any Natural Language conversation program because it emulates at some extent the

conceptual apprehension mechanism. Needless to say, it would be necessary to improve some of its parts in order to accomplish this goal effectively (n-arity relations; higher deepness of context in rule generation), but the whole architecture seems to us sufficiently solid to perform such task.

In a world where knowledge organization and management seems to become more and more important, programs that propose to help the user on organizing his own concepts also tend to be useful. There is already several of this kind (in the World Wide Web, for example), and Clouds can possibly join the group. More than the program itself, the framework seems valuable. The idea of using these learning algorithms in apprehending users' models may be useful, especially in situations in which it is important to have a structure of his knowledge (e.g. text interpretation, WWW search, etc.).

Being part of a bigger AI project (the above-referred Dr. Divago), Clouds is a major milestone that served as an important point of reflection. Individually, it is becoming an autonomous project aimed at knowledge extraction and user modeling.

References

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K. and Slattery, S.: Learning to Extract Symbolic Knowledge from the World Wide Web. Technical report (1998) CMU-CS-98-122.

Hampton, J.: Concepts: The elements from which prepositional thought is constructed, thus providing a means of understanding the world (1998) http://mitpress.mit.edu/MITECS/work/hampton_r.html.

Hofstadter, D.: Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. (1996) Basic Books; ISBN: 0465024750

Kakas, A.; Michael, A. and Mourlas, C; Abductive Constraint Logic Programming. Technical Report, University of Cyprus, 1998.

Muggleton, S. and Feng, C.: Efficient Induction of Logic Programs. In Proceedings of the First Conference on Algorithmic Learning Theory, Tokyo. (1990) Ohmsa Publishers. Reprinted by Ohmsa Springer Verlag.

Muggleton, S.: Inductive Logic programming. (1992) Academic Press.

Murphy, G. L. and Medin, D. L.: The role of theories in conceptual coherence. Psychological Review 92. (1985)

Novak, J. D. and Gowin, D. B.: Learning How To Learn, New York: Cambridge University Press. (1984)

Pereira F. C. and Cardoso, A.: Wondering in a Multi-Domain Environment with Dr. Divago. Proceedings of the 8th Cognitive Science of Natural Language Processing (CSNLP-8). Ireland. August (1999)

Quinlan, J. R.: Learning Logical Definitions from Relations, Machine Learning 5. (1990)

Rips, L. J.: Similarity, typicality and categorization. In Vosniadou, S. and Ortony, A. (Eds.), Similarity and Analogical Reasoning. (1989) Cambridge: Cambridge University Press.

Veale, T. and Keane, M. T.: Metaphor and Memory: Symbolic and Connectionist. Issues in Metaphor Comprehension, in the Proceedings of the European Conference on Artificial Intelligence Workshop on Neural and Symbolic Integration, Amsterdam. (1994)