Validation of CBL Principles*

Hans–Rainer Beick
Meme Media Laboratory
Hokkaido University
Kita 13, Nishi 8, Kita-ku
060-8628 Sapporo, Japan
beick@meme.hokudai.ac.jp

Klaus P. Jantke
Deutsches Forschungszentrum für
Künstliche Intelligenz GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany
jantke@dfki.de

Abstract

Case Based Reasoning (CBR) is deemed a rather promising technology of knowledge acquisition and related knowledge processing in recent computer science. Much of the CBR's attraction is based on the charming simplicity of its ideas. In particular, knowledge acquisition which is performed during CBR by incrementally collecting cases migth be understood as Case Based Learning (CBL).

The main goal of this paper consists in the application of validation scenarios to the investigation of principles of CBL. For this purpose, the domain of formal languages is used, where CBR is straightforwardly formalized to allow for a justified assessment of CBL principles. It is demonstrated that one of the key principles of improving case bases during CBR is not valid, seen from the viewpoint of a correct learning.

Two attempts of principle refinement are undertaken and validated subsequently. For the special domain, an improvement is gained.

Introduction

For the investigation, we need some background in CBR and CBL, in inductive inference of formal languages, and in complex systems validation.

In CBR, knowledge is represented in a more episodic form directly reflecting problem solving experience and, hence, being concrete and detailed, rather than abstract and declarative. This is deemed to model human knowledge acquisition and problem solving appropriately (cf. (Riesbeck & Schank 1989), (Kolodner 1993), e.g.).

In particular, knowledge acquisition which is performed during CBR by incrementally collecting cases migth be understood as Case-Based Learning (CBL), like in (Aha 1991) and (Aha, Kibler, & Albert 1991).

When a CBR system is in use for (mostly interactive) problem solving, it is frequently fed with new cases representing currently unresolved problems. These current problems shall be solved, according to the system's former experience stored in its case base. For this purpose, the system's case base is searched for formerly experienced cases that are similar to the current problem on hand. Those cases are taken and possibly adapted for problem solving.

CBR, in general, and CBL, in particular, are somehow inductive in spirit. Thus, it is worth to inspect some of the most alluring CBR and CBL principles for their power and limitations. But, we refrain from an in-depth discussion of validation issues and direct the interested reader to (Boehm 1984), (O'Keefe & O'Leary 1993), and (Jantke, Knauf, & Abel 1997), e.g.

The present investigations aim at validity assessments of CBL principles. A quite similar endeavour has been undertaken in (Dötsch & Jantke 1996).

This paper is a shorter, but more advanced version of (Beick & Jantke 1998). But there, a more detailed introduction into CBR, CBL, inductive learning of formal languages, and validation (of learning systems) is given. A special Prolog-tool for case-based language learning is described which is used here, too. In (Akaishi & Beick 1998), the first steps in using the IntelligentPad system (cf. (Tanaka 1989)) for the case-based learning are done.

The main goal of the present investigation consists in an experimental application and validation of some CBL principles. For this purpose, the domain of formal languages is used, where CBR is straightforwardly formalized to allow for a justified assessment of CBL principles. It is demonstrated that one of the key principles of improving case bases during CBR is *not valid*, seen from the viewpoint of correct learning.

According to this principle, a CBR system may selectively collect cases, thus adopting to its environment's needs. Whenever the system fails on certain cases, the corresponding information is used to enhance the system's area of competence by storing this particular new case.

The Validation of Learning Systems

The crux is that, given any learning problem and any learning system, it is generally undecidable whether or not the system on hand is able to solve the learning problem faced to.

*Copyright ©1999, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

For assessing the validity of complex systems, there have been proposed validation scenarios of several types. Complex systems are investigated on different levels of abstraction and approaches are classified by several features including a distinction of so-called black box and white box approaches.

A wide collection of interactive validation scenarios are intuitively rather similar to the so-called TURING test (cf. (Turing 1950)). Regardless of the legitimate criticism on TURING's ideology (cf. (Halpern 1987), e.g.), the perspective underlying the TURING test approach has led to a lot of validation approaches which invoke human experts for systems' interrogation.

We sketch the common idea underlying the approaches of this family, briefly: The human validators choose target phenomena. When a particular target phenomenon has been selected, test cases are generated to probe the system. Experimentations with the system yield certain results. Those results are subject to the experts' evaluation. Individually evaluated experimentation results provide some insight into the system's behaviour and, thus, into the issue of its validity. However, they are somehow local by nature. The ultimate validity assessments are synthesized upon the totality of interactive validation results.

First systematic investigations towards the validation of learning systems can be found in (Grieser, Jantke, & Lange 1997), e.g.

The present paper is intended to tailor these general ideas towards the validation of CBL principles.

Validation Scenarios

The main stages of interactive systems validation according to the so-called TURING test approach are

- test case generation,
- experimentation by feeding in test cases,
- evaluation of experimentation results,
- synthesis of validity assessments.

A particular *validation task* is determined by a target problem, by a candidate system, and by a certain understanding what it means to solve the problem appropriately.

Here, we do not deal with *verification tasks* which are particularly characterized by the substantial advantages of, first, a formalized problem specification and, second, formalized requirements determining criteria of success in problem solving. Based on these preferable, but rather idealistic assumptions, verification can be usually performed deductively. In contrast, validation is facing induction problems.

User expectations and needs of application domains are rarely explicitly available. It is a key assumption of validation that human beings might sufficiently well substitute for the formal knowledge which is either not existent or not available, at least.

Thus, validation scenarios are characterized by the division of labour between humans and automated tools for each of these four phases, respectively.

The Concrete Validation Scenario

The objects wich are to validated are case-based learning principles. For validating particular CBL principles, they are implemented in a case-based manner. In this paper, the learning of formal languages is used as the special domain for the implementation.

A more detailed introduction into the used CBR and CBL notions and notation is given in the next section. A test case consists of a given case-based description of a formal language and of a syllabus for teaching.

The result of the experiment is also a case-based decription of a language, the actual hypothesis of the learning process. In the evaluation of the experiment, both language descriptions are compaired. For that, the same CBR semantics is used for both descriptions.

Normally, a large set of test cases is necessary for a validation. In this paper, only one test is described, for a better understanding. It is the extract of a large number of tests and leads to a validity assessment recommended.

CBR and CBL

In CBR, knowledge is represented in the form of particular cases with a suitable similarity measure rather than any generalized form. The key idea is that such episodic knowledge comes along during a CBR system is in use. Therefore, the CBR paradigm is deemed a key for alleviating the truly serious bottleneck of knowledge acquisition. According to this perspective, a CBR system in use is usually changing over time. Learning takes place.

The Basic Notions of CBR and CBL

The following notions of CBR are used:

- $\mathcal{P}$ is a nonempty set of **problems**.
- $\mathcal{S}$ is a nonempty set of **solutions**.
- $\mathcal{SM}$ is a nonempty and partially ordered set of **similarities**. A **similarity measure** is a function $\sigma : \mathcal{P} \times \mathcal{P} \rightarrow \mathcal{SM}$. For $s = \sigma(P, Q)$, it is also said that P has the similarity s compared with Q.
- In addition, a mapping **notSimilar** on $\mathcal{P} \times \mathcal{P}$ is introduced. (For instance, if there is a minimum min in $\mathcal{SM}$, we define that $notSimilar(P, Q)$ means $\sigma(P, Q) = min$.)
- A **case base** is a finite sequence of pairs $[problem, solution]$.

On this basis, the relation "*problem P is most similar to problem P'*" is well defined. For any given problem and case base, a case which is most similar to the problem should be found. (The similarity of a case to a problem P is defined by the similarity of the case problem to P.) The solution of this case is the experience which should be used for problem solving.

In general, for a given problem several most similar cases could exist in the case base. Then, one case must be selected by a **selection strategy**. This strategy can use further knowledge modelled in such notions like **priority of a solution** or **relevance of a case**.

On the other hand, there could be no similar case in the case base. Then, a **default** solution is taken.

In particular, knowledge acquisition which is performed during CBR by incrementally collecting cases migth be understood as **Case Based Learning** (CBL), like in (Aha 1991) and (Aha, Kibler, & Albert 1991), for instance.

The overall CBL approach is exemplified on the following principle: **Given any CBR system, apply it. Whenever it works successfully, do not change it. Whenever it fails on some input case, add this experience to the case base. Don't change anything else.**

According to this principle, a CBR system may selectively collect cases, thus adopting to its environment's needs. Whenever the system fails on certain cases, the corresponding information is used to enhance the system's area of competence by storing this particular case appropriately.

Case–Based Language Description

For the case-based description of a formal language, the following appointments are made:
• Problems are words over a fixed alphabet.
• The solutions are 1 for '*belonging to the language*' and 0 for the opposite case. Here, solutions are also called **classes**, seen from the view point of classification.

After fixing a similarity, a selection strategy, and a default solution, every case base describes a formal languages: A word belongs to thelanguage if and only if the solution 1 is found by the CBR process for it.

The Concrete CBR Basis

At the beginning, the alphabet, the similarity, the selection strategy, and the default class is fixed.

For the language, the alphabet $\Sigma = \{\mathsf{a,b}\}$ is used, with words *abab*, *aa* and *b*, for instance.

The similarity is a binary relation. $\sigma(v, w)$ equals 1, exactly if v is a subword of w. 1 stands for *similar* and 0 for *non-similar*. So, the word *aba* is similar to *aaba* and to itself, but not to *abba*. Here, $notSimilar(P, Q)$ means $\sigma(P, Q) = 0$.

For CBR, the classification should be done with the *selection strategy* **'selection per sequence'** in connection with the default 0:
• Given any query q, it searches for the **first case** $[w, c]$ where w is similar to q. If such a case is found, c determines how to classify q.
• Otherwise, the **default class** 0 is taken.

The following case base is used as the description of the object which is to be learnt in the process of the validation : [bab,0], [aa,1].

For using the principle of the overall approach, it is clear what correct classification means. But it is necessary to define how a new case is to be added. This definition depends on the selection strategy. A '*careful adding for selection per sequence*' is used:

• If the use of one specific case causes an *incorrect* classification, the new case is inserted directly before this troublesome case.
• If the default is causing an incorrect classification, the new case is added at the end of the case base.

For instance, the following classifications with their reasons are got:

a : 0 (default-value)
bab : 1 (Case 1 is the *first (and only)* similar one.)
babaa : 1 (Case 1 is the *first* similar case.)

The teaching process starts with the empty case base as the actual hypothesis. The teaching syllabus consists of the first 1000 words over Σ, connected with the class defined by the given language. For illustration, we display the first and the last cases of this teaching sequence: [a,0], [b,0], [aa,1], [ab,0], [ba,0], ..., [bbbbaabbb,1], [bbbbabaaa,0], [bbbbabaab,0].

In general, the result of the learning process also depends on the used sequence in teaching and the size of the given information about the object which is to be learnt. But here, the influence of the sequence is not so important and the sequence is long enough for getting the wanted effects.

The Validation of the Overall Principle

The experiment described in the last section leads to the following case base as the final hypothesis of the learning process:

$$\begin{gathered}[aabab, 0], [babaa, 0],\\ [aabbab, 0], [babbaa, 0],\\ [aabbbab, 0], [babbbaa, 0],\\ [aabbbbab, 0], [babbbbaa, 0],\\ [aabbbbbab, 0], [babbbbbaa, 0],\\ [aa, 1].\end{gathered}$$

At first glance, the database is larger than the pre setting standard. This may be more or less normal because one cannot expect that learning results in optimized solutions. However, this is only a somehow superficial comment. The problem is considerably more involved.

The case [bab,0] will never be added to the case base by the overall principle because it is always correctly classified by the default.

That means, for instance, that all cases of the form '$[aab(b)^n ab, 0]$' have to be added to the case base during learning. As this is an infinite number of cases, any current case base never gives a correct description of the pre set language.

The detected phenomenon is a non-pathological one. When choosing a case base at random, it frequently occurs. Thus, already very simple have yield a first insight of some generality.

To some extent, this exhibits the *invalidity* of the CBL principle under inspection.

CBL without Defaults?

The analysis of the considered learning example leads to the following observation:

Cases can exist in the case base which are not necessary for the classification of their own problem because of the default classification. But, they are necessary for the correct classification of other problems. Such cases will never be added to the case base if the overall approach is used.

Learning without any classification default could be a way out. We consider the same language to be learnt, the same learning syllabus, and the overall learning method, too. The only difference is that the classification procedure, used by the learning method, works without a default. If there is no similar case, the output is like *noSolution* and hence, the case is added to the case base. Then, the following hypothesis case base is created:

$[aabab, 0], \ldots, [babbbbbaa, 0]$,
$[aa, 1], [a, 0], [b, 0]$.

It is obvious that there is no improvement. The learning method creates two default cases by itself. That shows that default management and case-based learning are closely connected.

CBL in Two Steps

In the moment of getting a test case, there is no other, obvious possibility for deciding: *Adding the new case if and only if the classification works incorrectly.* But, if there is a bigger **block** in the case base, block denotes a maximal subsequence with a uniform solution and without gaps, we can try to substitute cases by '*better and fewer*' ones.

This leads us to a '**two step learning**'. In the first step, the learning process is done, by constructing a extented case base with normal and generalized cases as a hypothesis (cf. (Bergmann & Wilke 1996), for a closely related approach taking generalized cases into account). In the second step, the extented case base is reduced to a (classical) one.

For the generalization of cases, further assumptions for the similarity are necessary which are fulfilled by the subword relation:

- The similarity predicate $<$ should be transitive.
- Let $Pred(x, y) = \{ z \mid z < x \text{ and } z < y \}$ the set of all common predecessors of x and y, related to $<$. This set should be finite for all x and y (and of course generally computable).

Generalization and Evaluation of Cases

An **extented case base** consists of (classical) cases and **generalized cases**. The classification is always done with and without these generalized cases, in a uniform way. For all problems contained in the (classical part of the) case base, both classifications have to have the same solution. In every block, the generalized cases stand at the top and the plain cases at the end. A generalized case has the form $[word, class, evaluation]$. The following conditions are always fulfilled:

- The *class* is this class used in the whole block.
- Normally, the **evaluation** is the nonempty set of all words of cases of the block so that *word* is similar to them. The generalized case represents all these cases. If a represented case causes a classification then the representing generalized case does so, as well, because of the transitivity of the similarity. But, an empty evaluation means that the generalized case is *deleted for ever* and not used for the classifications.

A (plain, i.e. conventional) case is added if the classification without extented cases works incorrectly, in the same way like before.

If a new case $[w, c]$ is added to the extented case base, the following is also done:

- If the case creates a new block at the top or at the end, then nothing else is changed.
- If the case divides a block into three new blocks, the extented cases of the former block, which are now in the first of the three blocks, are copied to the third block.
- If the case is embedded in a block, every minimal member of the following sets causes a new generalized case at the top of the block if there is still no one in the block:

$Pred(w, x) \cap \{ z \mid [z, c, \{\}] \text{ not in the block} \}$
$\cap \{ z \mid [z, c, \{\}] \text{ correct \& necessary in the block} \}$
$\cap \{ z \mid [z, c] \text{ not case in the block} \}$

for every other case x of the block. Correctness in the block means that all case words standing behind the block have the same classification both with and without this generalized case. Necessity means that the cases before the block don't cause the classification of the word.

In addition, the evaluation of all the extented (generalized) cases of the block is completed if necessary.

For every information [w,c] in the learning process, the classification with and without generalized cases is compared. If there is a difference, the generalized case that caused the difference is deleted by substitution of the evaluation by the empty set. And then, this comparison is repeated until both results are equal.

For the example learning task, we get the following generalized cases:

$[bbbbb, 0, \{\}], [bbbbaa, 0, \{\}], \cdots, [aab, 0, \{\}]$,
$[bab, 0, \{aabab, babaa, aabbab, \ldots, babbbbbaa\}]$,
$[baa, 0, \{\}], [bb, 0, \{\}], [ba, 0, \{\}], [ab, 0, \{\}], [b, 0, \{\}]$.

One can easily recognize that the generalized case $[bab, 0, \{\ldots\}]$ represents all additional cases.

So far, the result is semantically quite satisfying. But it is syntactically odd, because the form of the hypothesis, which includes generalized cases, do not meet the requirements of the originally assumed specification language. Some reduction step is necessary.

Reduction of the Extented Case Base

The reduction could be done in the following way: A threshold value for the size of the evaluation is fixed. say 8, for instance. The following is done for all the generalized cases behind the threshold value, where the procedure is starting with the largest elements following the ordering in the case base:
Transform the chosen generalized case into a classical one, remove all represented cases, and delete them in the other evaluations.

This procedure yields description of the object to be learnt from which we started originally. This fact is a first hint to the validity of the learning method presented above. at least in the domain of language learning.

In the future, a variety of related experiments are necessary. We did present only one of the many possibilities of combining CBL, generalization, evaluation. and reduction appropriately.

Conclusions

The ideas of CBR and CBL are rather intuitive and alluring. This makes it considerably difficult to separate the chaff from the wheat. Systematic systems validation may help to get a better understanding of the power and the limitations of principles in CBR, in general, and in CBL in particular.

The reader may consult (Dötsch & Jantke 1996), for instance, where large series of experiments are reported which support the believe that some of the charming ideas of CBL do not work as desired.

Besides the present paper's intended contribution to case-based reasoning, the validation of CBL principles is mainly understood as a case in complex interactive systems validation. A considerably small amount of learning experiments has pointed to substantial flaws of a sample principle focused on in the present paper. This is illustrating that the present approach works. Further investigations will go into more details and extend our first results.

Some simple CBL learning principles are not valid because they do ignore important information. Like in tennis, you lose the game if you lose the Big Points. But in learning, it is not decidable which points are big and which are not. Here, a first step is done in the correction of lost Big Points, by processing its consequences. In contrast, in tennis the game might be over. Thus, there is some hope - in learning, at least.

References

Aha. D. W.; Kibler. D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Aha. D. W. 1991. Case-based learning algorithms. In Bareiss, R., ed., *DARPA Workshop on Case Based Reasoning, May 8 - 10, 1991, Washington DC, USA*, 147–157. Morgan Kaufmann.

Akaishi, M., and Beick, H.-R. 1998. Case based learning using the IntelligentPad system. In Wittig, W. S., and Grieser, G., eds., *LIT-98, Proc. 6. Leipziger Informatik-Tage, Leipzig, October 1-2, 1997*, 27–33. Forschungsinstitut für InformationsTechnologien Leipzig e.V.

Beick, H.-R., and Jantke, K. P. 1998. Tools for validation of CBL principles. In Grieser, G.; Beick, H.-R.; and Jantke, K. P., eds., *Aspects of Intelligent Systems Validation, January 1998, Ilmenau, Germany, Meme Media Laboratory, Hokkaido University, Series of Meme Media Management, Report MEME-MMM 98 2*, 53–71.

Bergmann, R., and Wilke, W. 1996. On the role of abstraction in case-based reasoning. volume 1168 of *Lecture Notes in Artificial Intelligence*, 28–43. Springer-Verlag.

Boehm. B. W. 1984. Verifying and validating software requirements and design specifications. *IEEE Trans. Software* 1(1):75 88.

Dötsch, V., and Jantke, K. P. 1996. Solving stabilization problems in case-based knowledge acquisition. In Compton, P.; Mizoguchi, R.; Motoda, H.; and Menzies, T., eds., *Pacific Knowledge Acquisition Workshop, Oktober 23 25, 1996, Sydney, Australia*. 150-169. University of New South Wales, Department of Artificial Intelligence.

Grieser, G.; Jantke, K. P.; and Lange, S. 1997. A formal framework for validation of learning systems. In Wittig, W. S., and Grieser, G., eds., *LIT 97, Proc. 5. Leipziger Informatik-Tage, Leipzig, 25./26. September 1997*, 111 116. Forschungsinstitut für InformationsTechnologien Leipzig e.V.

Halpern, M. 1987. Turing's test and the ideology of artificial intelligence. *Artificial Intelligence Review* 1:79 93.

Jantke, K. P.; Knauf, R.; and Abel, T. 1997. The TURING test approach to validation. In Terano, T., ed., *15th International Joint Conference on Artificial Intelligence, IJCAI-97, Workshop W32, Validation, Verification & Refinement of AI Systems & Subsystems, August 1997, Nagoya, Japan*, 35 45.

Kolodner, J. L. 1993. *Case-Based Reasoning*. Morgan Kaufmann.

O'Keefe, R. M., and O'Leary, D. E. 1993. Expert system verification and validation: A survey and tutorial. *Artificial Intelligence Review* 7:3-42.

Riesbeck, C. K., and Schank, R. C. 1989. *Inside Case-Based Reasoning*. Lawrence Erlbaum Assoc.

Tanaka, Y. 1989. A toolkit system for the synthesis and the management of active media objects. In *1st Intern. Conference on Deductive and Object-Oriented Databases, Kyoto*, 269–277.

Turing, A. M. 1950. Computing machinery and intelligence. *Mind* LIX(236):433 460.