NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4173 - Assignment 1

Topic Name

Case Based Reasoning

Group:
Group Case_Based_Reasoning_Group 7

Authors:
Piri Babayev (pirib)
Danilas Miscenko (danilasm)
Aleksander Simmersholm (alekssim)

September 18, 2020

Abstract

In the first part of out method paper we introduced the Case-Based Reasoning's core concepts, going over its paradigm and functionalities. We introduced the core parts of what goes into a CBR-system, and furthered this explanation through definitions and specific constructs designed for CBR in mind.

In the second part we go deeper into the core phases and parts of a typical CBR system, and their respective functions and roles as components of the whole CBR system. We also go over examples in where CBR can be used, its versatility as a tool and which medium it operates on.

In the third part we take a closer look at the concept of similarity in CBR, defining its role and the multitude of ways it affects the process and assessments done in the system. We moreover go through how different methods attains this classification and what to be considered when similarity is involved between two cases.

In the last part, we examine a research paper written by Armin Stahl called "Learning Similarity Measures: A Formal View Based on a Generalized CBR model". This paper attempts to generalize the already established CBR model in a way that makes room for defining more modern use and approaches of CBR-systems. It does also go over its inherent effect from generalizing the model on similarity measures.

Table of Contents

List of Figures

List of Tables

1 Introduction

Case-Base Reasoning stands out in the crowd of Machine Learning approaches by opening up and presenting all its core concepts at once - it indeed presents a method that reasons its way to a solution by analyzing and using the cases of previous experiences. One could immediately draw parallels with our own, human way of solving problems, and not surprisingly, CBR indeed has its roots in Cognitive Science (Richter und Weber, 2013, p. 533). One of the most intuitive approaches to arriving at the solution is using relevant past knowledge and experience - if a snake suggests you eat an apple, you would recall that reptilians, and animals in general, do not speak human languages, and come to a conclusion that perhaps those mushrooms you had for dinner were of a different kind than you originally thought.

Before we put CBR under further scrutiny, a few aspects of interest have to be pointed out. CBR first and foremost is a **learning paradigm** - it presents and describes an overall approach to problem solving, and leaves us with the freedom to choose how to implement its components (Mitchell, 1997, p. 230). That leads us to our second aspect - CBR can be employed to tackle situations of very different nature, from reasoning about legal cases based on the previous experience with rulings (Mitchell, 1997, p. 240), or in an application to help people address lower back pain Mork [1].

A very high-level description of CBR would start the discussion by presenting the case base - this is the memory repository where "experiences" or more formally cases, are stored. Any time a new problem needs addressing, CBR tries to find a case with a similar problem, which would, hopefully, point to a solution fit to address the initial query. After doing so and, potentially receiving some feedback, the system would learn from this experience and it will be adapted in one form or another by updating the case base accordingly.

Having discussed CBR in action, some clarifications are in order. CBR is what one would refer to as a **lazy** learning method, meaning that it does all of its hard work, that is, generalize beyond training data, only when a query has been presented. Intuitively, a CBR system waits for a problem to be formulated and presented for it to be able to decide how to solve it. Another vital detail is the fact that CBR responds to a query by examining **similar** stored cases and ignores the ones that are very different. Furthermore, these instances have no restraints on their representation and are usually chosen to fit the problem domain - images, sounds or text, everything is on the table for the CBR. (Mitchell, 1997, p. 240)

This simple yet, in our minds, ample introduction to CBR should have excited and filled reader's imagination with exhilarating possibilities, but as one might have guessed, there are some difficulties one needs to address before hurling themselves blindly into the embrace of CBR. The aforementioned flexibility of CBR indeed is the source of many complications that arise when designing such a system - questions about the instance representation; query and case comparison; adaptation of the new cases etc. are central in the analysis of this approach.

To keep this paper brief and focused, we will first dissect CBR into basic components, explain how they all fit together and which processes make it tick. Further, we will embark on the examination of the concept of similarity and how it can be improved, as well as presenting some current applications for it.

[1] Which, in fact, is coordinated by NTNU, and we will leave the reader to decide whether our decision to include it here was tacky or not.

2 Foundations

To delve deeper into depths of thrilling opportunities CBR offers, one needs to get better acquainted with it first. As it can be daunting to tackle a system of such proportions we will begin by going through one cycle of a CBR routine to get a better intuition of the process and understand how each element fits within the system. To do so, we will be reusing CBR-cycle model by Agnar Aamodt and Enric Plaza [fig 1] Aamodt A. (2001).

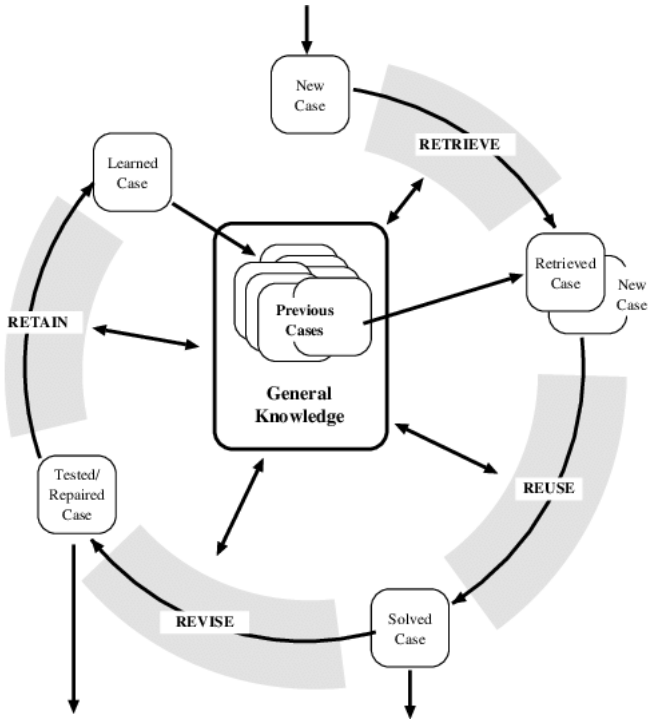

Figure 1: CBR Cycle (Image courtesy of Agnar Aamodt and Enric Plaza)

The main focus here will be put on the four processes shown on the diagram: Retrieve, Reuse, Revise and Retain.

Our CBR cycle starts with **Retrieval**, that is, when a new problem, also known as query, is introduced to the system. The posed problem is, of course, the task that the system is expected to solve. In some cases, the system is expected to find an example that is the most analogous to the posed query rather than come up with a solution, e.g. recommend a similar book or suggest the correct spelling. In either case, in order to do so, our CBR goes to the Case Base and looks up a previous case (or cases) that are the most **similar** to the problem at hand. After retrieving a case from our general knowledge base we try to apply it to the task at hand, e.g. **Reuse** it. Having done that, the system hopefully has arrived at a potential solution, which means it is time to evaluate that proposal, say, by a domain expert. This step, unsurprisingly, is called **Revise** and the main goal here is to evaluate the usefulness of the proposed solution, and additionally, prepare it for the next step. The main objective of **Retain** is to update the base case with the new experience by **adapting** the new solution and make sure that it can be reused in the next cycle.

As an example, we will take the situation that is favored by many authors of the field (and we regret to inform the reader that we were unable to resist the urge to reuse it). Let us consider the following: a rather expensive sports car rolls into a service centre in bad need of repair - one of its headlights refuses to function properly. To do so, our technicians rush to the best expert in the field, that is, our CBR system. First, the problem needs to be introduced - perhaps, by filling up a standardized form or questionnaire that could help the system narrow down the search. Now, it is all up to the expert - it needs to analyze the input and look up the cases that could help the

one at hand. It could look up for the cars of the same brand and model, or similar problems - the significance of each attribute heavily depends on the problem that needs to be solved. As such, color of the car would hardly be of importance, while a problem with back lights could be quite helpful. After receiving the feedback from our technicians about the success of the solved case our system could use the newly gained knowledge to generalize from "back light not working" and "headlight not working" to "signalling lights not working"[2] and as such adapt the new solution for future use.

So far we have skimmed over the entire process by painting the big picture with broad strokes. This was done deliberately, in the hopes that the reader can, now, organize the mess from Introduction chapter into something more coherent. It will also, in turn, allow us to go into more detailed discussion on how one designs such a system, and what one should keep in mind while working on CBR.

In the Introduction section, as well as by providing the example above, we gave a glimpse of how CBR can be used and hinted on the versatility of the tool. How exactly can CBR be adapted to solve these contrasting problems? Let's start by examining the building blocks of CBR, the cases, which are also sometimes referred to as instances. The first question to pose is how does one represent them? The good, but ambiguous answer is, it depends, and it does so mainly on the problem domain. A CBR system that focuses on classification of images would definitely benefit from a representation that includes pictures, an e-commerce recommend system would fare much better if it has some attribute-value type, for example an object-oriented representation. Other types include text, speech, sensor data and conversational representations, or indeed any combination of them. It is important to note that one involves domain experts when designing a CBR system, to better tailor it to the needs of the end user. (Richter und Weber, 2013, p. 108)

The design of the case representation needs to be done carefully since this decision affects almost all other components of our system. While discussing the Retrieval step, we have mentioned that the system would fetch cases most **similar** to the problem at hand. The choice to embolden the term was very much deliberate since the concept of similarity takes up one of the central discussions in CBR. As the next chapter will cover Similarity with utmost deliberation, we will withhold from discussing it further in this chapter, simply noting that this notion is the inductive bias in CBR - when a new case is classified, it will be most similar to the ones that this classification was derived from. (Mitchell, 1997, p. 234)

At the Retain step we have also mentioned another important aspect of a CBR system, the **adaptation** of the newly gained experience to the base case. What we have failed to note was that adaptation can happen at the Reuse stage as well - it is not often that the case base has a solution that matches the problem exactly. Either way, adaptation is a process of adapting a given case to suit the current needs of the process. In our example the older solution was indeed useful, but still had to be adapted for the usage, and consequently for the storage along with the newly gained knowledge.

The approach to adaptation differs greatly depending on the case representation, but also on the know-how - in our example, the CBR system was able to merge two solutions into one, more general one. Something similar happens at the retrieval step when the system has to compare the cases with the new one - for example, if the car had a problem with the engine, then the system would decide to look for the issues with similar engines, rather than same brand/model. So, the components of the case base, similarity measures and adaptation have to communicate with each other at the various stages of the cycle, and therefore are in need of a medium to be able to do so.

This medium is called a Vocabulary and it has the big responsibility of regulating what the system can reason about. If there is no concept of "engine" (however this concept is defined, encoded and used), our components of the system will have a hard time trying to find a solution for something that does not exist (in its understanding). In a nutshell, the Vocabulary is the domain of all

[2]Here we note that author's understanding of cars is rather limited, and as such, suspension of disbelief in readers is highly encouraged.

possible discussions within the CBR system. A given concept can have several interpretations in the Vocabulary that can be used used for different tasks. (Richter und Weber, 2013, p. 35)

Given that we are discussing CBR in the context of Machine Learning, what does exactly constitute as **learning** in a CBR system? An imminent answer comes when one conveniently breaks the discussed cycle into two main components, each with their own main task - Reasoning and Learning. Retrieve and Reuse belong to the former, while the latter one incorporates Revise and Retain. Intuitively, the second half contributes to learning by integrating new experiences and increasing the knowledge base. After (quite) a few iterations CBR will have a larger collection of cases that it can utilize to perform better.

The performance of the system, on the other hand, depends not only on the amount of knowledge it has but also on how it can utilize it. There are many aspects as to how this can be achieved by the system - the Vocabulary needs to enable the overall reasoning, case representation to describe the elements of the system, adaptation to properly evolve the knowledge base, similarity to compare and select the most suitable cases, etc. We have spent this chapter scrutinizing CBR in hopes that it provides a better understanding to the reader, but cannot shake off the feeling that we have left more unanswered questions than we started with. To atone ourselves, the next chapter we will focus on the Similarity concept and explore better the precautions one should take when coming into closer contact with it.

3 The Concept of Similarity in CBR

Similarity is not a concept that has a clear definition. We may say that two items are similar if they share some attributes, but their similarity is also heavily dependent on the context they're evaluated in. To give a familiar example; two cars are brought into a repair shop. Are they similar in any way? One of the cars is a cheap Volkswagen Golf, while the other one is a Ferrari. The Golf is yellow, while the Ferrari is red. Out of these attributes, one might say that they are not similar in any way, but if they share a problem, say both have malfunctioning headlights, then they become contextually similar and might require the same solution to fix the headlights. If a different vehicle, say a scooter, is brought in, then the two cars become more similar, because they are both vehicles with four wheels.

The point of this example is to show that similarity is hard to define without relying on context. The usecase of the CRB system will therefore be a huge factor in defining how similarity between cases is calculated. A good method that is commonly used is to use some algorithm to cluster the cases together based on their attributes, then when a new case arrives, the k-Nearest Neighbour algorithm is used to find the nearest neighbours to the new case. The Idea is that if the case is similar to some other case that has a solution, then it is likely that that solution will work for the case in question. This method is defined as such:

> "For some fixed x, each y that satisfies R(x, y, z) for all z is called nearest neighbour of x.
>
> Notation: NN(x, y)" (Richter und Weber, 2013, p. 118).

In this definition, x is the current case, y and z are cases in the case base that are not x, and R(x, y, z) is an expression defined as "*x is more similar to y that x is to z*". The set of k nearest neighbours to x is then returned in the order of similarity to x, so that a solution out of that set may be picked out. The similarity of cases is calculated based on some principles and similarity measures, which will be addressed later on in this chapter.

One of the principles used for calculating similarities is the Local-Global principle. This principle is usually adequate enough for most applications, which is why it's the first one we learn about on this course. The principle says: "There are elementary (local or atomic) description elements (attributes). Each object or concept A is (globally) described by some construction operator C from the local elements:

$$A = C(A_i | i \in I)$$

where I denotes some index set of the atomic elements A_i (attributes)" (Richter und Weber, 2013, p. 96). If all of the attributes are just "flat values", then the constructor produces flat vectors with the attributes of the object. The constructor can also produce complex descriptions of the object if the attributes are sets or lists of some sort which contain many values within themselves.

By using this principle we can define and compare cases. Neat! But what if some of the attributes are more important than others in deciding whether the cases are similar or not? Referring to the example provided in the beginning of this chapter, the two cars that were brought in to the shop were either similar or different depending on the context we used to look at them through. Also, the importance of attributes might change depending on the task which has to be preformed. An example of this would be the two cars at the shop. The color of each car may me irrelevant if the owners of the cars want to fix the headlights, but may be more relevant if they instead want to change the color of the car.

We can assign weights to each attribute to indicate how important each attribute is to the similarity measure. Say we have some function to check for similarity between two objects by comparing each atribute of one object to it's counterpart from the other object:

$$sim(a, b) = \sum_{i=1}^{i=n} sim_i(a_i, b_i)$$

By assigning a weight to each attribute, we can get a similarity of the objects based on the important attributes:

$$sim(a, b) = \sum_{i=1}^{i=n} \omega_i sim_i(a_i, b_i)$$

It is allso important to note that the sum of all weights $(\omega_1, \omega_2 ... \omega_n)$ is equal to 1. This method is very effective at assigning importance to individual attributes, but sometimes attributes may exhibit importance when linked together with one or more other attributes. The textbook on CBR gives a good example of this: Say a bank is checking your monthly income and monthly spending to determine if you're eligible for a loan [tab 1].

Query	**Case 1**	**Case 2**	**Case 3**
Income	1000	2000	6000
Spending	1500	5000	4500
Eligible?	No	No	Yes

Table 1: Bank loan example from the book (Richter und Weber, 2013, p. 141)

We as humans can see that because some of the people have higher spending than income, they will be unable to pay back a loan, and that is why they are not eligible for one, but to implement that into our function would be tricky. We can however introduce a "virtual variable" which is not a part of the data set. This variable would be called "Relation I/S" and would be Income divided by Spending. By assigning the weights of the two original variables to our new variable we can more correctly determine the similarity between each case. Also, by only needing to check one variable instead of the two, we have made our similarity check easier to compute. Now all the system needs to check is whether Relation I/S $\geqslant 1$, since it's $\geqslant 1$ when Income $\geqslant$ Spending.

So how does one actually go about calculating similarity? There are many types of similarity measures that one can use, and they're all valid options. The main two deciding factors are the CBR system itself and the data you have in your case base. Some similarity measures depend on how the data is structured, and the choice also depends on what you want the system to do, and how you want it to do it.

The most simplest similarity measure is to simply count the similarities. The Hamming measure is a measure hailing from coding, which just checks all of the attributes and returns how many of the attributes have the same value. This measure works best if all attributes are boolean values. This measure also fails to capture the importance of each variable, but versions of this method have been derived which assigns weights to each attribute. The weighted method is predictably called the Weighted Hamming Measure. A measure that handles real values has also been invented. It's called the Tversky measure and is just a more generalised version of the Hamming measure.

Metric similarity measures are used for cases with real value attributes and are usually distance based. By distance, we usually mean that each case is defined as a vector, and the distance between vectors defines how similar they are. The distances might be euclidean, or otherwise. "For the real space, a useful type of property often used is the invariance property" (Richter und Weber, 2013, p. 130). This means that since the values are treated as distances between vectors, they are unchanging if both vectors get transformed.

Transformational similarities are most useful for attributes that are symbolic. This measure often uses the Levenshtein distance, which looks at a string of symbols and determines how many transformation operations need to be done in order to transform the symbol string in case 1 into the symbol string in case 2. The less operations required, the "shorter the distance" and the more similar the symbol strings are. The operations in question are *Insertion*, *Deletion* and *Modification*.

4 Current Applications of CBR

To start of Section 4, we would like to explain our choice in our paper. Case-based reasoning entails algorithms and methods from a broad range of topics in ML, having the ability to use supervised learning algorithms in its similarity measures, or k-means clustering. We looked at possible papers and found it problematic to nail down one specific paper or application because most similarity topics are covered in multiple topics of computer science. We were initially going to write about melodic approximation, but had a hard time defining a use case for this in specifically CBR as the paper based itself on dynamic programming instead. We choose therefore to base the topic of Similarity in CBR on the research done by Armin Stahl, which is available on Blackboard. We will go over caveats he addresses in his research with the classical CBR-model and his version of the Generalized CBR model (Stahl (2005)).

The inherent problem with similarity measures comes from the fact that, even though it is a crucial component of CBR systems, clear methodologies for defining these measures does not yet exist. Most similarity measures derive from the context in which the application is to propose a solution for. Multiple approaches exist as seen in our previous section, but most can be boiled down to general distance metrics solutions. Of course, this is an oversimplification as more complex similarity measures can be considered with domain knowledge, or machine learning approaches, but to keep the justification frank, there exists a need for clearer goal-oriented approaches to similarity measures.

In the research paper, Sthal propose a more formal generalization derived from the classic CBR model that is mentioned in previous sections. Unfortunately, for the sake of generalization, the model might seem a tad bit complex next to what is already mentioned, but we will try our best to keep each method cycle frank and to the point.

He proposes a figure (which we will define later) as a generalization, making it easier to orient clearer definitions of modern methods. He commends the classical figure for its clarity but minds us that the classical approach creates limitation on more recent research as we have come farther in research than at the time the classical model was proposed. This makes it harder to create clear definitions on methods used in CBR-systems today.

The classical CBR model accounts for a typical problem-solving situation: It is given a problem, then proposes the solution(s). This creates the structure of cases that consists of a problem part, and a corresponding solution part that successfully solved the problem in the past. It falls under the typical paradigm of CBR-systems where "Similar problems have similar solutions", given the assumption that the system already possesses some case knowledge in the specific domain. This can be naïvely summarized as: a function that indexes a list of solutions that might apply to the current query problem based on the similarity of past problems, resulting in an action of useful information retrieval. However, nowadays slightly different approaches are used where this classical approach, which consist of a problem part and a solution, might not work.

Another limitation of the classic CBR cycle is that it does not take into consideration issues and topics of recent research. Some of these are

- **Dialog strategies**, which is the topic on how to obtain efficient and accurate queries as input to the system to better its efficiency in providing more precise solutions.
- **Explanation**, where the CBR system explains the underlying reasons of its proposed solutions.
- **Feedback**, where CBR systems incorporates a feedback-loop to better its understanding on cases and refining solutions to problems.

The classical CBR cycle consists of the four basic steps: Retrieve, Reuse, Revise and Retain, so we will first try to organize our new function cycle under the already established classical CBR cycle to make it easier to compare the CBR formalization to the classical CBR-model [tab 2].

Retrieval	• Situation-Characterization, Query. • Case, Case Characterization, Case Lesson, Case Space. • Similarity Measure. • Retrieval Function.
Reuse	• Adaptation Function. • Output Function, Output Space .
Revise	• Output Processing Function. • Feedback Function.
Retain	• Learning Functions

Table 2: Function cycle categorized by their equivalent phases in classical CBR phases

So finally, we arrive at the Formal Generalization of the Classical CBR Cycle. It presents a more formal way for analysing CBR functionality in more detail.

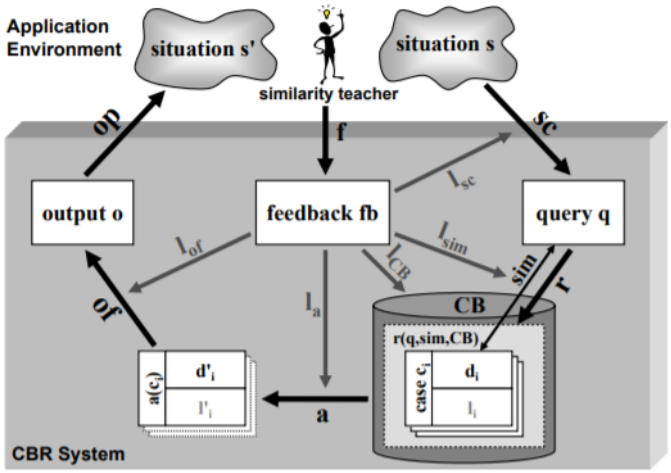

Figure 2: The Generalized CBR Cycle by Armin Sthal

The CBR system is to provide the necessary information by generating a corresponding output solution (o).

Situation-Characterization. Query. At the first step, a situation/problem (s) is characterized into a query (q) by comparing it to existing situations in the situation space. (sc)

Case, Case Characterization, Case Lesson, Case Space. In the second step, the query (q) is compared to the cases (c) that exists in the case base (CB) which might contain useful information for the solving the situation which the query is derived from. It is important to note that cases consist of case lessons and case characterizations. A case might also only consist of characterizations to be used as utility for solutions, as in, they might inherent some sort of information that might be of value to future case estimations.

Similarity Measure. While still in case base, similarity measures are used to estimate the query to a case. We denote this as sim(q,c).

Retrieval Function. The retrieval function then returns a set of case-bases for the given query according to their similarity values. The retrieval function denoted (r) and can be done in a multitude of ways. Either, by simply retrieving the list of cases ordered by their similarity values, or some other arbitrary way. This function is denoted r(q,sim,CB).

Output Function, Output Space The output function, denoted (of), has the ability to transform the output (o) in many ways, from a respective dialogue for the original input situation (s), to further filter and select from the cases left after the adaptation process. This depends on the application's purpose and if there is a further need to elaborate the finished result. One problem from the classic CBR-system this component might solve is the ability to explain the underlying reasoning for the output solution.

Output Processing Function. The output process function, denoted (op), is an informal process which typically modifies, or creates a situation (s) into the situation space (S). It can have the job of updating the situation accordingly with the new case it just proposed. Of course, if the current situation is still left without a sufficient solution it might just update the current situation case new initial situation to run through the cycle again.

Feedback Function. The CBR system receives feedback from the feedback function about its performance and the usefulness of its proposed solution, applying learning strategies to itself and improving its function by updating its knowledge containers.

Learning Functions. The learning functions allows the system to take note from feedback and appropriating itself, modifying case bases, similarity measures and adaptation. It describes the general purpose of learning functions, where the system might improve itself on some matter.

5 Conclusions and Further Work

Case-Based reasoning is a learning paradigm with an active research community that are even to this day still making recent advancements in the field, bettering the overall methodology and efficiency of CBR to further its goals as a solution driven machine learning process. New and exciting discoveries are made every day with interesting ramifications for the rest of the Artificial Intelligence field. CBR is by itself also a paradigm that can take a very flexible form in it's use cases due to it being so contextually dependent.

This method paper has illuminated us, and hopefully you too, on just how broad the topic of CBR truly can be, not only in how powerful and versatile it can be with the help of learning process, but also on how it stands as a tool for creating machine learning systems that are based on cognitive architectures. The flexibility of this type of system is a direct result of the system having so many diverse methods of computing, as well as not being closed off for implementation of other methods that might prove more successful in certain applications and domains.

While writing this paper we gained a lot of insight on one of the core concepts of CBR, namely similarity and its effect on the CBR-cycle, both in the traditional approach, and the more generalized approach. The topic of similarity in itself is extremely broad and far-reaching, offering a multitude of methods for calculating similarity between different case-structures.

Finally we have assessed the possibility of a different conceptualization of the CBR methodology, making room for more recent approaches in the CBR-field and abstracting ourselves from the traditional approach. Where it would no longer suffice for more effective approaches to common problems in the field. With a more generalized approach to the CBR-model, it's easier to make clear definition to specific approaches when it comes to the concept of similarity.

Further work might include exploring other concepts of CBR, such as Adaptation and Evaluation.

Bibliography

[Aamodt A. 2001] Aamodt A., Plaza E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. In: *Ai Communications* (2001), Nr. 7, S. 39–59

[Mitchell 1997] Mitchell, Tom M.: *Machine Learning*. McGraw-Hill, 1997

[Mork] Mork, Paul J.: *selfBACK*. – URL `http://www.selfback.eu/`

[Richter und Weber 2013] Richter, Michael M. ; Weber, Rosina: *Case-Based Reasoning, A Textbook*. Springer-Verlag, 2013

[Stahl 2005] Stahl, Armin: Learning Similarity Measures: A Formal View Based on a Generalized CBR Model. In: *German Research Center for Artificial Intelligence DFKI GmbH* (2005), S. 507–521. – URL `https://link.springer.com/chapter/10.1007/11536406_39`