Handling Default Data under a Case-based Reasoning Approach

Bruno Fernandes[1], Mauro Freitas[1], Cesar Analide[1], Henrique Vicente[2] and José Neves[1]
[1]*Department of Informatics, University of Minho, Braga, Portugal*
[2]*Department of Chemistry, University of Évora, Évora, Portugal*

Keywords: Case-based Reasoning, Intelligent Systems, Similarity Analysis.

Abstract: The knowledge acquired through past experiences is of the most importance when humans or machines try to find solutions for new problems based on past ones, which makes the core of any Case-based Reasoning approach to problem solving. On the other hand, existent CBR systems are neither complete nor adaptable to specific domains. Indeed, the effort to adapt either the reasoning process or the knowledge representation mechanism to a new problem is too high, i.e., it is extremely difficult to adapt the input to the computational framework in order to get a solution to a particular problem. This is the drawback that is addressed in this work.

1 INTRODUCTION

Case-Based Reasoning (CBR) provides the ability of solving new problems by reusing knowledge acquired from past experiences. In fact, it can be described as the process of solving new problems based on solutions of similar past ones. One well-known example is the mechanic that listening to the strange sound that the engine is making and the symptoms of the problem, uses his/her past experiences to make a first analysis of the problem, and uses a solution that has already been castoff on similar situations. This technique can be used in domains where there is a large amount of information that has been acquired through experience (Aamodt and Plaza, 1994). CBR is used, especially when similar cases have similar terms and solutions, even when they have different backgrounds, i.e., the knowledge acquired when solving some situation can be used as a first approach to solve new ones (Balke T. et al, 2009).

Nowadays, CBR is used in several areas and has a tremendous potential. There are examples of its use in The Law with respect to dispute resolution (Carneiro D. et al, 2009; Carneiro D. et al, 2010), in medicine (Kolodner J., 1992; Khan A. and Hoffman A., 2002), among others. A typical, and maybe the greatest, problem in the use of CBR is related to the availability of the data and the cost of obtaining it (Stahl A. and Gabel T., 2006).

The typical CBR cycle (Figure 1) clearly defines the vocabulary of the domain and the steps that should be followed to have a consistent model. The initial description of the problem becomes a new case that is used to *Retrieve* one or more cases from the repository. At this stage it is important to identify the characteristics of the new problem and retrieve cases with a higher degree of similarity to it. Then, a solution for the problem emerges when combining the new case with the retrieved case, on the *Reuse* phase. Here, the solution of the retrieved case is reused, tested and adapted, if possible, to the new case, thus creating the solved case that is suggested as a solution (Aamodt and Plaza, 1994). However, when adapting the solution it is essential to have feedback from the user since automatic adaptation in existing systems is almost impossible. Currently, the main adaptation techniques are the null adaptation (no adaptation is needed); replace the attributes of the solution and not the structure of the solution; use rules; and by analogy transfer the knowledge of the past problems to the new ones and with that generate the solution (Kolodner J., 1992). It is on the *Revise* stage that the suggested solution is tested and it is here where exists an interaction with the user letting him/her correct/adapt/change the suggested solution by creating the Test Repaired Case that sets the solution of the new problem. The test repaired case must be correctly tested to ensure that the solution is indeed correct. This is iterative since the solution must be tested and adapted while the result of applying that solution is unsatisfying.

Fernandes B., Freitas M., Analide C., Vicente H. and Neves J..
Handling Default Data under a Case-based Reasoning Approach.
DOI: 10.5220/0005184602940304
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 294-304
ISBN: 978-989-758-074-1

During the *Retain* (or learning) stage the case is learned and the knowledge base is updated with the new case. In this approach, the domain knowledge stands for itself, while the cases refer to past problems and their solutions (Kolodner J., 1992).

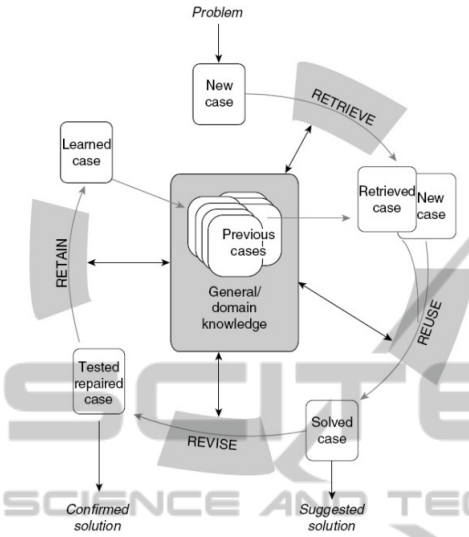

Figure 1: Typical CBR Cycle (Aamodt and Plaza, 1994).

This work brings the evidence that existent CBR systems are neither complete nor adaptable enough for all domains. In some cases, the user is required to follow the similarity method defined by the system, even if it does not fit user's needs. On present CBR system there is an obstacle related to the capability of dealing with unknown and incomplete information. This new approach will be completely generic and it will have a focus on that.

As it will be explained later, existing CBR tools follow different approaches to problem solving, i.e., looking at the same patterns from different perspectives. However, they all look too complex and the effort to adapt them to a specific problem domain is comparable to develop a full new CBR. Also, an important feature that often is discarded is the ability to compare strings. In some problem domains, strings are important to describe a situation, the problem in itself or even an event. If the CBR is only prepared to work with numbers then this gap will prove to be fatal. Our CBR approach uses several of the most popular string similarity algorithms, namely the Dice Coefficient (Dice L., 1945), Jaro-Winkler (Winkler W., 1990) and the Levenshtein Distance (Levenshtein V., 1966) Strategies. Also, it will be possible to set the weight of a particular attribute along the attributes that make a case's argument. This new approach to CBR will potentiate a new perception of this methodology for problem solving, going in depth in aspects like the case´s *Quality-of-Information* (*QoI*) or the *Degree-of-Confidence* (*DoC*), a measure of one´s confidence that the value of a particular attribute is the expect one. Under this framework it will be possible to handle unknown, incomplete or even contradictory data. It will also induce a change on the typical CBR cycle set in Figure 1. This new cycle will have into consideration *a normalization* phase, with the attributes values of the case argument being set in the interval [0,1], therefore making easier their *DoCs* evaluation, as it will be seen later in the paper. It will also improve performance and reliability of the similarity strategy. The case base will be given in terms of triples that follow the pattern:

Case = {<*Raw-case*, *Normalized-case, Description-case*> | where *Raw-case* and *Normalized-case* stand for themselves, and *Description-case* is made on a set of strings or even in free text, which may be analysed with specific string similarity algorithms}

2 RELATED WORK

On the one hand, Case-Based Reasoning is either extremely simple but also extraordinarily efficient. CBR's applicability ranges across different areas like Medical Diagnosis, The Law, Assisted Student Learning, Diets, Income Tax Law, Drawing, Classification, Cooking Recipes, just to name a few.

Indeed, interesting results using CBR in education have been reported. It has been found that CBR could greatly improve Assisted Student Learning (Riesbeck and Shank, 1991; Tsinakos A., 2003). A domain independent education environment named SYIM_ver2 was developed using CBR techniques as the reasoning component, being able to monitor the student's progress/performance, recording his/her learning needs, or even answering to student's questions. The system uses a *Process of Identification of Similar Question* (PISQ) that is responsible to find similar cases. PISQ uses cases that are stored on an Educational Knowledge Base and provides answers to questions via a key word search feature. It is also employing two search methods: controlled vocabulary search and free text search among the cases of the knowledge base. This allowed the system to tackle some known pedagogical drawbacks that were present in SYIM_ver1 (Tsinakos A., 2003).

In medicine, CBR is mainly used for diagnosis and in diet recommendations (Kolodner J., 1992;

Khan A. and Hoffman A., 2002). Physicians use knowledge acquired from past experiences to treat new patients. Let's imagine a doctor that is treating a patient, which suffers from a specific disease, with specific symptoms. After treating the patient, the doctor assimilates this knowledge and creates a new "case". When a new patient appears, presenting the same symptoms, the doctor can "retrieve" the past "case" and treat this new patient in a much more effective way, saving time and money. Here the cases are stored in a healthcare knowledge base and contain information about the healthiness of a particular individual, as triples in the form <Problem, Solution, Justification> (Kolodner J., 1993). The system is only helping the Doctor in his/her analysis. The Doctor must validate if the case retrieved by the system reports to the disease that the patient is suffering. If this is the situation, then it is just a question of adapting the solution to the problem at hands.

The diet system helps in defining the patient diet when he is admitted to the hospital. Many have special needs in regard their medical conditions, nutrient requirements, calories restrictions or even cultural backgrounds. The system, using CBR, is able to provide a menu that fits the patient needs (Khan A. and Hoffman A., 2002). The expert must then validate if the menu is appropriate and tell the system if the solution was accepted or adapted, with more cases being added to the knowledge base.

Lawyers and judges are always using their past experiences to evaluate new cases. In other words, they are using CBR. Related to the legal subject there are many examples. One of them is related to online dispute resolution, since we now face an era of huge technology usage and so, traditional courts cannot deal with the amount of problems put to them. A solution is to use a CBR system that using past experiences is able to achieve a quicker solution to the real problem (Carneiro D. et al, 2009) (Carneiro D. et al, 2010). Another example was the development of HYPO, a model that improves legal reasoning. The name is a reference for both actual and hypothetical cases (Rissland E. and Ashley K., 1989). Each case includes information like the description of the problem, the arguments, the explanation, and the justification, among others. The input to HYPO is named as *Current Fact Situation* (cfs) and is provided by the user. The output is provided as a *case-analysis-record* with HYPO's analysis; a *claim-lattice* showing graphic relations between the new case and the cases retrieved by the system; *3-ply* arguments used and the responses of both plaintiff and defendant to them; and a *case citation summary* (Rissland E. and Ashley K., 1989).

The domain knowledge is kept in three different places, namely the Case Knowledge Base (CKB); the library of dimensions; and the normative standards (which are used to select the best cases). The CKB is the place where the cases are stored. Dimensions encodes legal knowledge for arguing about a claim from a certain point of view. A key aspect is that dimensions encapsulates the legal sense of what makes the situation better or worse from a case perception by the actors (Rissland E. and Ashley K., 1989).

HYPO has experienced very good results allowing new legal reasoning systems to be developed using HYPO as a base. It may be referred the TAX-HYPO that acts on the domain of tax law (Rissland E. and Skalak D., 1989), or the CABARET, also for income tax law domain (Skalak D. and Rissland E., 1992) and the IBP that makes predictions based on argumentation concepts (Brüninghaus and Ashley, 2003).

On the other hand, there are several CBR software tools developed and tested that are being applied in real situations, namely Art*Enterprise, developed by ART Inference Corporation (Watson I., 1996). It offers a variety of computational paradigms such as a procedural programming language, rules, cases, and so on. Present in almost every operating system, this tool contains a GUI builder that is able to connect to data in most of proprietary DBMS formats. It also allows the developer to access directly the CBR giving him/her more control of the system, which in most situations may turn into a drawback. The CBR in itself is also quite limited, once the cases are represented as flat values (attribute pairs). The similarity strategy is also unknown, which may represent another constraint since there is no information about the possibility of adapting it. These features are presented on Table 1.

CasePower which is a tool developed using CBR. It was developed by Inductive Solutions Inc. (Watson I., 1996) and uses the spreadsheet environment of Excel. The confines of the Excel functionalities are echoed on the CBR framework, making it more suitable for numerical applications. The retrieval strategy of CasePower is the nearest neighbour, which uses a weighted sum of given features. In addition to this strategy the system attaches an index to each case in advance, improving the system performance in the retrieval process. The adaptation phase in this system may be made through formulas and macros from Excel but the CBR system in itself cannot be adapted as explained

in Table 1.

CBR2 denotes a family of products from Inference Corporation, and may be the most successful CBR. This family is composed by CBR Express which stands for a development environment, featuring a customer call tracking module; the CasePoint which is a search engine for cases developed by CBR Express; the Generator that allows the creation of cases-bases through MS Word or ASCII files; and finally the Tester which is a tool capable of providing metrics for developers. In this tool the cases structure comprises a title, a case description, a set of weighted questions (attribute pairs), and a set of actions. Such as CasePower, CBR2 uses the nearest neighbour strategy to retrieve the cases initially. Cases can also be stored in almost every proprietary database format. However, the main and most valuable feature of CBR2 is the ability to deal with free-form text. It also ignores words like *and*, *or*, *I*, *there*, just to name a few. It also uses synonyms and the representation of words as trigrams, which makes the system tolerant to spelling mistakes and typing errors.

The EasyReasoner that is a module within Eclipse, being extensively regulated since it is available as a C library, so there is no development interface and is only suitable for experienced C programmers. This software developed by Hayley Enterprises (Watson I., 1996) is very similar to ART and also ignores noise words and use trigrams to deal with spelling mistakes. Once more, these tools are not likely to be adapted because it is an API, and the similarity strategy is not even known.

The ESTEEM software tool, which was developed by Esteem Software Inc., has its own inference engine, a feature that allows the developer to create rules, providing a control over the induction process. Moreover, cases may have a hierarchy that can narrow the searches, which is very useful when accessing multiple bases and nested cases. The matching strategy is also the nearest neighbour. The advantages of this tool are the adaptability of the system and the usage of two different similarity strategies (nearest neighbour and induction). A disadvantage is the fact that it only runs on Windows (Watson I. D., 1996).

jCaBaRe is an API that allows the usage of Case-Based Reasoning features. It was developed using Java as a programming language, which gives this tool the ability to run in almost every operating system. As an API, jCaBaRe has the possibility to be adapted and extended providing a solution for a huge variety of problems. The developer can establish the cases attributes, the weight of each attribute, the retrieval of cases, and so on. One of its main limitations is the fact that it still requires a lot of work, by the developer, to achieve an working system. Several features must be developed from scratch.

The Kate software tool that was produced by Acknosoft (Althof, K-D., 1995) and is composed by a set of tools such as Kate-Induction, Kate-CBR, Kate-Editor, and Kate-Runtime. The Kate-Induction tool is based on induction and allows the developer to create an object representation of cases, which can be imported from databases and spreadsheets. The induction algorithm can deal with missing information. In these cases, Kate is able to use the background knowledge. The Kate-CBR is the tool responsible for case comparison and it uses the nearest neighbour strategy. This tool also enables the customization of the similarity assessments. Kate-Editor is a set of libraries integrated with ToolBook, allowing the developer to customize the interface. Finally, Kate-Runtime is a set of interface utilities connected to ToolBook that allows the creation of an user application.

Another CBR system is the ReMind, a software tool created by Cognitive Systems Inc. (Watson I., 1996). The retrieval strategies are based on templates, nearest neighbour, induction and knowledge-guided induction. The templates retrieval is based on simple SQL-like queries, the nearest neighbour focus on weights placed on cases features and the inductive process can be made either automatically by ReMind, or the user can specify a model to guide de induction algorithm. These models will help the system to relate features providing more quality to the retrieval process. The ReMind is available as a C library. It is also one of the most flexible CBR tools, however it is not an appropriate tool for free text handling attributes. The major limitations of this system are the fact that cases are hidden and cannot be exported, and the excessive time spent on the case retrieval process. The nearest neighbour algorithm is too slow, and besides the fact that inductive process is fast, the construction of clusters trees is also slow (Watson I., 1996).

The characteristics of all these tools are compiled on Table 1, where the implemented features are represented by "+" and the features not implemented by "-". This table provides an overview of the main features existing in each tool.

Table 1: Characteristics of existent CBR Software Tools.

	CBR Software Tools Characteristics					
CBRs/Features	Type	Similarity Strategy	Database Platforms	Cross Platform	Free Text Handling	Adaptability
ART*Enterprise	User Application	Unknown	Several	+	-	-
CasePower	User Application	Nearest Neighbour	Unknown	+	-	-
CBR Express	User Application / API	Nearest Neighbour	Several	+	+	-
Easy Reasoner	API	Unknown	Unknown	+	+	-
ESTEEM	User Application / API	Nearest Neighbour / Induction	Several	-	-	+
jCaBaRe	API	To be developed	To be developed	+	To be developed	+
KATE	User Application	Nearest Neighbour / Induction	Several	-	-	-
ReMind	User Application / API	Nearest Neighbour / Induction	Unknown	+	-	+

3 KNOWLEDGE REPRESENTATION AND REASONING

Many approaches for knowledge representations and reasoning have been proposed using the Logic Programming (LP) paradigm, namely in the area of Model Theory (Kakas A. et al, 1998; Gelfond M. and Lifschitz V., 1988; Pereira L. and Anh H., 2009), and Proof Theory (Neves J., 1984; Neves J. et al, 2007). We follow the proof theoretical approach and an extension to the LP language, to knowledge representations and reasoning. An Extended Logic Program (ELP) is a finite set of clauses in the form:

$$p \leftarrow p_1, \cdots, p_n, not\ q_1, \cdots, not\ q_m \quad (1)$$

$$?\,(p_1, \cdots, p_n, not\ q_1, \cdots, not\ q_m)\ \ (n, m \geq 0) \quad (2)$$

where *?* is a domain atom denoting falsity, the p_i, q_j, and p are classical ground literals, i.e., either positive atoms or atoms preceded by the classical negation sign $\neg$ (Neves J., 1984). Under this representation formalism, every program is associated with a set of abducibles (Kakas A. et al, 1998; Pereira L. and Anh H., 2009) given here in the form of exceptions to the extensions of the predicates that make the program. Once again, LP emerged as an attractive formalism for knowledge representations and reasoning tasks, introducing an efficient search mechanism for problem solving.

Due to the growing need to offer user support in decision making processes some studies have been presented (Halpern J., 2005; Kovalerchuck B. and Resconi G., 2010) related to the qualitative models and qualitative reasoning in Database Theory and in Artificial Intelligence research. With respect to the problem of knowledge representation and reasoning in Logic Programming (LP), a measure of the *Quality-of-Information* (*QoI*) of such programs has been object of some work with promising results (Lucas P., 2003; Machado J. et al, 2010). The *QoI* with respect to the extension of a predicate *i* will be given by a truth-value in the interval [0,1], i.e., if the information is *known* (positive) or *false* (negative) the *QoI* for the extension of $predicate_i$ is 1. For situations where the information is unknown, the *QoI* is given by:

$$QoI_i = \lim_{N \to \infty} \frac{1}{N} = 0 \qquad (N \gg 0) \quad (3)$$

where *N* denotes the cardinality of the set of terms or clauses of the extension of $predicate_i$ that stand for the incompleteness under consideration. For situations where the extension of $predicate_i$ is unknown but can be taken from a set of values, the *QoI* is given by:

$$QoI_i = {}^{1}/_{Card} \quad (4)$$

where *Card* denotes the cardinality of the *abducibles* set for *i*, if the *abducibles* set is disjoint. If the *abducibles* set is not disjoint, the *QoI* is given by:

$$QoI_i = \frac{1}{C_1^{Card} + \cdots + C_{Card}^{Card}} \quad (5)$$

where C_{Card}^{Card} is a card-combination subset, with *Card* elements. The next element of the model to be considered is the relative importance that a predicate assigns to each of its attributes under observation, i.e., w_i^k, which stands for the relevance of attribute *k* in the extension of $predicate_i$. It is also assumed that the weights of all the attribute predicates are normalized, i.e.:

$$\sum_{1 \le k \le n} w_i^k = 1, \forall_i \quad (6)$$

where $\forall$ denotes the universal quantifier. It is now possible to define a predicate's scoring function $V_i(x)$ so that, for a value $x = (x_1, \cdots, x_n)$, defined in terms of the attributes of $predicate_i$, one may have:

$$V_i(x) = \sum_{1 \le k \le n} w_i^k \times QoI_i\,(x)/n \quad (7)$$

It is now possible to engender the universe of discourse, according to the information given in the logic programs that endorse the information about the problem under consideration, according to productions of the type:

$$predicate_i(x_1, \cdots, x_n) :: QoI \quad (8)$$

and evaluate the *Degree of Confidence* given by $DoC = V_i(x_1, \cdots, x_n)/n$, which denotes one's confidence in a particular term of the extension of $predicate_i$. To be more general, let us suppose that the Universe of Discourse is described by the extension of the predicates:

$$a_1(\cdots), a_2(\cdots), \cdots, a_n(\cdots) \ \ where \ \ (n \ge 0) \quad (9)$$

The solution we present does not only affect the case structure but also how similarity is calculated. So, assuming we have a clause that is mapped into a case, that clause has as argument all the attributes that make the case. The argument values may be of the type unknown or members of a set, may be in the scope of a given interval or may qualify a particular observation. Let us consider the following clause where the first argument value may fit into the interval [35,65] with a domain of [0,100], the value of the second argument is unknown, which is represented by the symbol ⊥, with a domain that ranges in the interval [0,20], and the third argument stands for itself, with a domain that ranges in the interval [0,4]. Let us consider that the case data is given by the extension of predicate f_1, given in the form:

$$f_1 : x_1, x_2, x_3 \rightarrow \{True, False\} \quad (10)$$

One may have:

$$\begin{cases} \neg f_1(x_1, x_2, x_3) \longleftarrow not\ f_1(x_1, x_2, x_3) \\ f_1(\underbrace{[35, 65], \quad \perp, \quad 1}_{attribute`s\ values\ for\ x_1, x_2, x_3}) :: 0.85 \\ \underbrace{[0, 100][0, 20][0, 4]}_{attribute`s\ domains\ for\ x_1, x_2, x_3} \end{cases} \quad (11)$$

Once the clauses or terms of the extension of the predicate are set, the next step is to transform all the arguments, of each clause, into continuous intervals. In this step, it is important to consider the domain of the arguments. As the second argument is unknown, its interval will cover all the possibilities of the domain. The third argument speaks for itself. The 0.85 value taken from the interval [0,1] denotes the *QoI* of the term $f_1([35, 65], \perp, 1)$. Therefore, one may have:

$$\begin{cases} \neg f_1(x_1, x_2, x_3) \longleftarrow not\ f_1(x_1, x_2, x_3) \\ f_1(\underbrace{[35, 65], [0, 20], [1,1]}_{attribute`s\ values\ ranges for\ x_1, x_2, x_3}) :: 0.85 \\ \underbrace{[0, 100]\ [0, 20]\ [0, 4]}_{attribute`s\ domains\ for\ x_1, x_2, x_3} \end{cases} \quad (12)$$

Now, one is in position to calculate the *Degree of Confidence* for each attribute that makes the term arguments (e.g. for attribute one it denotes one's confidence that the attribute under consideration fits into the interval [35,65]). Next, we set the boundaries of the arguments intervals to be fitted in the interval [0,1] according to the normalization procedure given in the form $(Y - Y_{min})/(Y_{max} - Y_{min})$, where the Y_s stand for themselves.

$$\begin{cases} \neg f_1(x_1, x_2, x_3) \longleftarrow not\ f_1(x_1, x_2, x_3) \\ x_1 = \left[\frac{35 - 0}{100 - 0}, \frac{65 - 0}{100 - 0}\right], \\ x_2 = \left[\frac{0 - 0}{20 - 0}, \frac{20 - 0}{20 - 0}\right], \quad x_3 = \left[\frac{1 - 0}{4 - 0}, \frac{1 - 0}{4 - 0}\right] \\ f_1\,(\underbrace{[0.35, 0.65], [0, 1], [0.25,0.25]}_{attribute`s\ values\ ranges\ for\ x_1, x_2, x_3}) :: 0.85 \\ \underbrace{[0, 1] \quad [0, 1] \quad [0, 1]}_{attribute`s\ domains\ for\ x_1, x_2, x_3} \end{cases} \quad (13)$$

The *Degree of Confidence* is evaluated using the equation $DoC = \sqrt{1 - \Delta l^2}$, as it is illustrated in Figure 2. Here Δl stands for the length of the arguments intervals, once normalized.

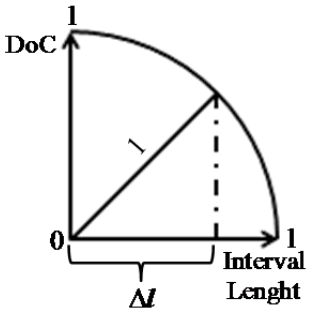

Figure 2: Evaluation of the Degree of Confidence.

$$\begin{cases} \neg f_1(x_1, x_2, x_3) \leftarrow not\ f_1(x_1, x_2, x_3) \\ f_{1_{doc}} \underbrace{(0.95, \quad 0, \quad 1)}_{attribute`s\ confidence\ values\ for\ x_1,x_2,x_3} :: 0.85 \\ \underbrace{[0.35, 0.65][0, 1]\ [0.25, 0.25]}_{attribute`s\ values\ ranges\ for\ x_1,x_2,x_3} \\ \underbrace{[0, 1] \quad [0, 1] \quad [0, 1]}_{attribute`s\ domains\ for\ x_1,x_2,x_3} \end{cases} \quad (14)$$

It is now possible to represent the case base in a graphic form, showing each case in the Cartesian plane in terms of its *QoI* and *DoC* (Figure 3).

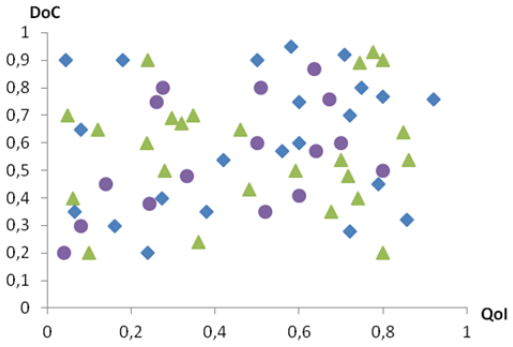

Figure 3: Case base represented in the cartesian plane.

To select possible solutions to the case problem under consideration, we look at Figure 3 where the best cluster is selected according to a similarity measure given in terms of the *modulus* of the arithmetic difference between the arguments of the cluster case with respect to their counterparts on the case problem. The new case is represented as the square inside the circle (Figure 4).

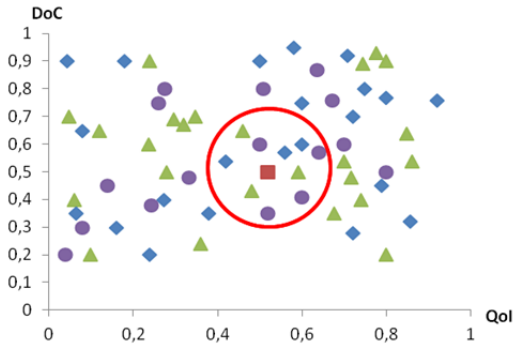

Figure 4: Clustering the case base.

4 A CASE STUDY

As a case study, consider the scenario where a company provides software product support to customers. When customers have problems in a given product, they open a new ticket with the description and the symptoms of the problem. Different products may share the same components so, the solution for a problem in a specific product can be found in a ticket regarding another product. In this case study, the severity of a problem was defined as being scaled as *Very Low*, *Low*, *Medium*, *High* and *Very High*. In a numeric scale this corresponds to the intervals [1,2], [3,4], [5,6], [7,8] and [9,10], respectively. There are five possible operating systems: HP-UX 11.31, Solaris 9, Solaris 10, Windows XP and Windows 7. As for the database provider, it may be one of the following: Oracle10g Release 1, Oracle10g Release 2, Oracle11g Release 1 and Oracle11g Release 2. The company has a total of twenty three products.

Figure 5 shows the main attributes of a new ticket that was opened by a customer. The severity of the problem was defined as being *Very High*. The database provider was not specified and this problem, described by comment "Description3", is affecting product with id 12. One's goal is to find the solution, to this new ticket, in the knowledge base. Having that in mind, let us consider a relational database that is given in terms of the three relations presented in Figure 6. That information is related with past problems that affected several products of the company. Some incomplete data, which is represented by the ⊥ symbol, is presented on the extensions of some relations or predicates that are being considered. For instance, for problem 2 (Problem Report table) the operating system is unknown. For problem 1, the Severity is *Low/Medium* and so its value ranges in the interval

[3,6] (union of intervals [3,4] and [5,6]), in a set defined between one and ten. Each problem has its own description.

Severity	System_id	DbProvider_id	Product_id	Description
Very High	3	Unknown	12	Description3

Figure 5: New ticket's characteristics and description.

The case repository is given in terms of the set {<*Raw-case*, *Normalized-Case*, *Description-case*>}, as it was stated above.

Let us consider the relations given in Figure 6, in terms of the *ticket* predicate, given in the form:

$$ticket: S_{everity}, Sy_{stem}, DB_{Provider}, Pr_{oduct} \rightarrow \{0,1\}$$

where 0 (zero) and 1 (one) denote, respectively, the truth-values *false* and *true*. Then, it would be possible to give the extensions of the predicates as:

{

$\neg ticket\ (S, Sy, DB, Pr) \leftarrow not\ ticket(S, Sy, DB, Pr)$

$ticket_1(\underbrace{[3,6],\ 3,\ \perp,\ 11}_{attribute`s\ values}) :: 1$

$\underbrace{[1,10][1,5][1,4][1,23]}_{attribute`s\ domains}$

$ticket_2(\underbrace{[7,8],\ \perp,\ 2,\ 12}_{attribute`s\ values}) :: 1$

$\underbrace{[1,10]\ [1,5]\ [1,4][1,23]}_{attribute`s\ domains}$

⇓ ***1st interaction: transition to continuous intervals***

$ticket_1(\underbrace{[3,6],[3,3],[1,4],[11,11]}_{attribute`s\ values}) :: 1$

$\underbrace{[1,10]\ [1,5]\ [1,4]\ [1,23]}_{attribute`s\ domains}$

$ticket_2(\underbrace{[7,8],[1,5],[2,2],[12,12]}_{attribute`s\ values}) :: 1$

$\underbrace{[1,10][1,5]\ [1,4]\ [1,23]}_{attribute`s\ domains}$

⇓ ***2nd interaction: normalization*** $\frac{Y - Y_{min}}{Y_{max} - Y_{min}}$

$ticket_1([0.222,0.556],[0.5,0.5],[0,1],[0.45,0.45])$

$ticket_2([0.667,0.778],[0,1],[0.33,0.33],[0.5,0.5])$

⇓ ***DoC calculation:*** $DoC = \sqrt{1 - \Delta l^2}$

$ticket_{1_{DoC}}(\underbrace{0.943,\ 1,\ 0,\ 1}_{attribute`s\ confidence\ values}) :: 1$

$\underbrace{[3,6]\ [3,3]\ [1,4]\ [11,11]}_{attribute`s\ values}$

$\underbrace{[1,10][1,5]\ [1,4]\ \ [1,23]}_{attribute`s\ domains}$

$ticket_{2_{DoC}}(\underbrace{0.994,\ 0,\ 1,\ 1}_{attribute`s\ confidence\ values}) :: 1$

$\underbrace{[7,8]\ [1,5][2,2]\ [12,12]}_{attribute`s\ values}$

$\underbrace{[1,10][1,5][1,4]\ [1,23]}_{attribute`s\ domains}$

}

The description of the case will be necessary when analysing the similarities between the description of the tickets present in the knowledge base and new problems that arrive to the system.

Now, by adapting the new case (Figure 5) to this approach we would have:

{

$\neg ticket\ (S, Sy, DB, Pr) \leftarrow not\ ticket(S, Sy, DB, Pr)$

$ticket_{new}(\underbrace{[9,10],\ 3,\ \perp,\ 12}_{attribute`s\ values}) :: 1$

$\underbrace{[1,10][1,5][1,4][1,23]}_{attribute`s\ domains}$

⇓ ***1st interaction: transition to continuous intervals***

Severity

Problem_id	Severity
1	Low/Medium
2	High

Comments

Problem_id	Product_id	Description
1	11	Description1
2	12	Description2

Platforms

Problem_id	System_id	DbProvider_id
1	3	⊥
2	⊥	2

Problem Report

Problem_id	Severity	System_id	DbProvider_id	Product_id	Description
1	[3,6]	3	⊥	11	Description1
2	[7,8]	⊥	2	12	Description2

Figure 6: Extension of the Relational Database model.

$$ticket_{new}(\underbrace{[9,10],[3,3],[1,4],[12,12]}_{attribute`s\ values}) :: 1$$

$$\underbrace{[1,10]\ \ [1,5]\ \ [1,4]\ \ [1,23]}_{attribute`s\ domains}$$

$$\Downarrow \textit{2nd interaction: normalization } \frac{Y - Y_{min}}{Y_{max} - Y_{min}}$$

$$ticket_{new}([0.889,1],[0.5,0.5],[0,1],[0.5,0.5])$$

$$\Downarrow \textit{DoC calculation: } DoC = \sqrt{1 - \Delta l^2}$$

$$ticket_{new_{DoC}}(\underbrace{0.994,\ \ 1,\ \ 0,\ \ 1}_{attribute`s\ confidence\ values}) :: 1$$

$$\underbrace{[9,10]\ [3,3]\ [1,4]\ [12,12]}_{attribute`s\ values}$$

$$\underbrace{[1,10]\ [1,5]\ [1,4]\ \ [1,23]}_{attribute`s\ domains}$$

}

Having all clauses normalized, the system is able to retrieve cases with higher similarity value. First, the input case is compared with every retrieved case from the cluster. The result is achieved by getting the average between the attributes, except the *Description*. It will be assumed that every attribute has equal weight. In clauses where each attribute has a weight, the average must be weighted.

Descriptions will be compared using String Similarity Algorithms, as stated before, in order to get a similarity measure between them. According to this, one first has to compare the four attributes that make the normalized case, obtaining the following similarity values between ticket 3 (three) and the other two:

{

$$ticket_{1_{DoC}}(\underbrace{0.943,\ \ 1,\ \ 0,\ \ 1}_{attribute`s\ confidence\ values}) :: 1$$

$$ticket_{2_{DoC}}(\underbrace{0.994,\ \ 0,\ \ 1,\ \ 1}_{attribute`s\ confidence\ values}) :: 1$$

$$ticket_{new_{DoC}}(\underbrace{0.994,\ 1,\ \ 0,\ \ 1}_{attribute`s\ confidence\ values}) :: 1$$

$$ticket_{DoC_{new\to1}} = \frac{|0.994 - 0.943| + |1 - 1| + |0 - 0| + |1 - 1|}{4}$$

$$ticket_{DoC_{new\to2}} = \frac{|0.994 - 0.994| + |1 - 0| + |0 - 1| + |1 - 1|}{4}$$

$$ticket_{DoC_{new\to1}} = 0.013$$

$$ticket_{DoC_{new\to2}} = 0.5$$

}

where "|" and "|" is our notation for *modulus*, i.e. it stands for the absolute value of a given number. It was assumed that every attribute has equal weight.

It is then necessary to compare the description of the new ticket with the descriptions of ticket one and two (for this case study, the strategy used was the Dice Coefficient):

{

$$ticket_{SIM_{new\to1}} = 0.327$$

$$ticket_{SIM_{new\to2}} = 0.814$$

}

With these values we are able to get the final similarity measure:

{

$$ticket_{new\to1} = \frac{0.013 + 0.327}{2} = 0.17$$

$$ticket_{new\to2} = \frac{0.5 + 0.814}{2} = 0.657$$

}

According to these results, our approach determines that ticket two is the most similar case, in our knowledge base, when receiving ticket three as input, and so it will be the case that will be retrieved by the system.

5 CONCLUSIONS AND FUTURE WORK

The comparison between free text parameters is an expensive and not so reliable task. However, there are many case studies that demand a huge reliability on those comparisons. For example, if we want to establish a relation between errors of an application reported by users, there is no way to quantify each one of the parameters. That was not however the only proposed goal. We also developed a solution that is able to handle default, unknown and incomplete data using concepts like the *DoC* and the *QoI*. This DoC denotes one's confidence in a

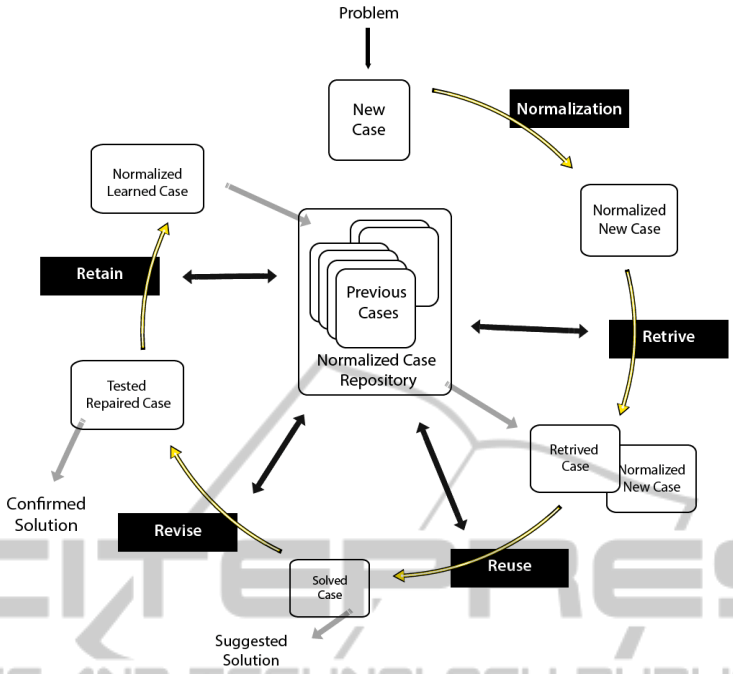

Figure 7: Adapted CBR cycle taking into consideration the normalization phase.

particular term of the extension of a predicate. It allows one to normalize all cases that live within the knowledge base and it improves the performance of the similarity analysis that is made when retrieving cases. In fact, this concept of calculating the *DoC* can be represented in an adapted CBR cycle, as shown in
Figure 7. The main difference between this new cycle and the well-known typical CBR cycle is the Normalization phase and the fact that all cases within the knowledge base are normalized. The Normalization phase is where the new case is normalized and where its *DoC* is calculated. In fact, when learning the case, the system is learning also the normalized case, allowing the cycle to retrieve cases that fulfil this structure. The Retrieve phase takes into account the *DoC* of each case when analysing similarities between cases. This analysis handles generic data without caring about the data type of the attributes. It is also able to analyse free text attributes using several String Similarities Algorithms which fulfil a gap that is present in almost all CBR Software Tools. This approach proved to be useful in other real situations such as schizophrenia diagnosis (Cardoso L. et al, 2013) or asphalt pavement modeling (Neves J. et al, 2012). These two cases revealed a demand that was not forgotten in this work, which is the possibility of setting the weights of each attribute on the fly. This feature gives the user the possibility to narrow the search for similar cases at runtime.

As future work it will be important to add more string similarity strategies. Furthermore, it may be also important to implement an independent CBR system to automatically choose which strategy is the most reliable to the current case study. Still related to the similarity strategies, our approach allows the user to define the weights of cases attributes on the fly, letting the user to choose the strategy he prefers.

ACKNOWLEDGMENT

This work is funded by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within projects PEst-OE/EEI/UI0752/2014 and PEst-OE/QUI/UI0619/2012.

REFERENCES

Aamodt A. & Plaza E., 1994. "*Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*", AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

Althoff K-D, Auriol E., Barletta R. & Manago M., 1995, "*A Review of Industrial Case-Based Reasoning Tools*", AI Intelligence.

Balke T., Novais P., Andrade F. & Eymann T., 2009. "*From Real-World Regulations to Concrete Norms for Software Agents – A Case-Based Reasoning Approach*", ICAIL.

Brüninghaus S. & Ashley K., 2003. "*Combining Case-Based and Model-Based Reasoning for Predicting the Outcome of Legal Cases*", Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR-03), pp. 65-79.

Cardoso L., Martins F., Magalhães R., Martins N., Oliveira T., Abelha A., Machado J. & Neves J., 2013. "*Schizophrenia Detection through an Artificial Neural Network based System*".

Carneiro D., Novais P., Andrade F., Zeleznikow J. & Neves J., 2009. "*The Legal Precedent in Online Dispute Resolution*", Jurix.

Carneiro D., Novais P., Andrade F., Zeleznikow J. & Neves J., 2010. "*Using Case Based Reasoning to Support Alternative Dispute Resolution*", Series Advances in Intelligent and Soft Computing.

Dice L., 1945. "*Measures of the Amount of Ecologic Association Between Species*", Ecology 26, pp. 297-302.

Gelfond M. & Lifschitz V., 1988. "*The stable model semantics for logic programming*", in Logic Programming – Proceedings of the Fifth International Conference and Symposium, pp. 1070-1080.

Halpern J., 2005. "*Reasoning about uncertainty*", Massachusetts: MIT Press, 2005.

Kakas A., Kowalski R. & Toni F., 1998. "*The role of abduction in logic programming*", in Handbook of Logic in Artificial Intelligence and Logic Programming, Vol. 5, pp. 235-324.

Khan A. & Hoffmann A., 2002. "*Building a case-based diet recommendation system*", Artificial Intelligence in Medicine 27, pp. 155-179.

Kolodner J., 1992. "*An Introduction to Case-Based Reasoning*", Artificial Intelligence Review 6, pp. 3-34.

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

Kovalerchuck B. & Resconi G., 2010. "*Agent-based uncertainty logic network*", in Proceedings of the IEEE International Conference on Fuzzy Systems – FUZZ-IEEE, pp. 596-603.

Levenshtein V., 1966. "Binary codes capable of correcting deletions, insertions, and reversals", Soviet Physics Doklady, pp. 707-710.

Lucas P., 2003. "*Quality checking of medical guidelines through logical abduction*", in Proceedings of AI-2003, London: Springer, pp. 309-321.

Machado J., Abelha A., Novais P. & Neves J., 2010. "*Quality of service in healthcare units*", International Journal of Computer Aided Engineering and Technology, Vol. 2, pp. 436-449.

Neves J., 1984. "*A logic interpreter to handle time and negation in logic data bases*", in Proceedings of the 1984 annual conference of the ACM on the fifth generation challenge, pp. 50-54.

Neves J., Machado J., Analide C., Abelha A. & Brito L., 2007. "*The halt condition in genetic programming*", in Progress in Artificial Intelligence - Lecture Notes in Computer Science, Vol. 4874, pp. 160-169.

Neves J., Ribeiro J., Pereira P., Alves V., Machado J., Abelha A., Novais P., Analide C., Santos M. & Fernández-Delgado M., 2012. "*Evolutionary intelligence in asphalt pavement modeling and quality-of-information*", Progress in Artificial Intelligence, Vol. 1, pp. 119-135.

Pereira L. & Anh H., 2009. "*Evolution prospection*", in New Advances in Intelligent Decision Technologies – Results of the First KES International Symposium IDT, pp. 51-64.

Riesbeck C. & Schank R., 1991. "*From Training to Teaching: Techniques for Case-Based ITS*", Intelligent Tutoring Systems: Evolution in Design, Lawrence Erlbaum Associates, Hillsdale, pp. 55-78.

Rissland E. & Ashley K., 1989. "*HYPO: A Precedent-Based Legal Reasoner*", Recent Advances in Computer Science and Law.

Rissland E. & Skalak D., 1989. "*Case-Based Reasoning in a Rule-Governed Domain*", In Proceedings of the Fifth IEEE Conference on Artificial Intelligence Applications.

Skalak D. & Rissland E., 1992. "Arguments and cases: An inevitable intertwining", Artificial Intelligence and Law, pp. 3-44.

Stahl A. & Gabel T., 2006. "*Optimizing Similarity Assessment in Case-Based Reasoning*", 21st National Conference on Artificial Intelligence.

Tsinakos A., 2003. "*Asynchronous distance education: Teaching using Case Based Reasoning*", Turkish Online Journal of Distance Education – TOJDE.

Watson I. & Marir F., 1994. "*Case-Based Reasoning: A Review*", The Knowledge Engineering Review, Vol. 9.

Watson I., 1996. "*Case-Based Reasoning Tools: An Overview*".

Winkler W., 1990. "*String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage*", Proceedings of the Section on Survey Research Methods, pp. 354-359.