International Journal of Artificial Intelligence Tools. Special Issue of IEEE ITCAI-96 Best Papers.
6:(3&4), 1997. In Press.

IMPROVING PERFORMANCE OF CASE-BASED CLASSIFICATION USING CONTEXT-BASED RELEVANCE

IGOR JURISICA
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4
juris@cs.utoronto.ca
(1–416) 978-7589

JANICE GLASGOW
Department of Computing and Information Sciences
Queen's University
Kingston, Ontario K7L 3N6
janice@qucis.queensu.ca
(1–613) 545-6058

Received (1 May, 1997)
Revised (19 June, 1997)

Abstract

Classification involves associating instances with particular classes by maximizing intra-class similarities and minimizing inter-class similarities. Thus, the way similarity among instances is measured is crucial for the success of the system. In case-based reasoning, it is assumed that similar problems have similar solutions. The case-based approach to classification is founded on retrieving cases from the case base that are similar to a given problem, and associating the problem with the class containing the most similar cases.

Similarity-based retrieval tools can advantageously be used in building flexible retrieval and classification systems. Case-based classification uses previously classified instances to label unknown instances with proper classes. Classification accuracy is affected by the retrieval process – the more relevant the instances used for classification, the greater the accuracy.

The paper presents a novel approach to case-based classification. The algorithm is based on a notion of similarity assessment and was developed for supporting flexible retrieval of relevant information. Case similarity is assessed with respect to a given context that defines constraints for matching. Context relaxation and restriction is used for controlling the classification accuracy. The validity of the proposed approach is tested on real-world domains, and the system's performance, in terms of accuracy and scalability, is compared to that of other machine learning algorithms.

Keywords: Case-Based Reasoning; Classification; Context-Based Similarity; Relevance Assessment.

1 Introduction

Classification involves associating instances with classes: based on the object's description, a classification system determines whether a given object belongs to a specified class. In general, this process consists of first generating a set of categories and then classifying given objects into the created categories. For the purpose of this paper, it is assumed that the categories have been determined *a priori* from a prescribed domain theory [44].

Various techniques have been utilized for the classification task, for example neural networks [43], genetic algorithms [13], inductive and instance-based learning [1, 46, 42] and case-based reasoning [36, 4]. Individual approaches are compared to one another based on the method they deploy, the accuracy they achieve, and the complexity of the used algorithm. Most systems extract classification rules from training examples during a learning process. Then, they use these rules for the classification of unseen instances. Case-based systems, however, store whole cases during the learning process and assess the similarity between a given problem and a stored case in a case base, in order to determine an appropriate class for the problem.

Evaluating a classifier's performance is not a straightforward process. Individual systems are usually tested on different problem domains, and because of differences in domain complexities, obtained performance measures cannot be compared directly. While performance is highly domain dependent, it is possible to derive evaluation techniques that allow for a meaningful performance comparison of different algorithms [2, 32].

This paper presents a novel, case-based classification scheme, called $\mathcal{T}A$**3**, and evaluates it with respect to several real-world domains. The proposed algorithm is based on a notion of relevance assessment [20] and on a modified nearest-neighbor matching [47]. Its modifications include:

- Grouping attributes into categories of different priorities so that different preferences and constraints can be used for individual categories during query relaxation;
- Using an explicit context, a form of bias represented as a set of constraints, during similarity assessment; and
- Using an efficient query relaxation algorithm based on incremental context modifications.

$\mathcal{T}A$**3** (pronounced tah-tree)[1] is different from other case-based approaches since instead of using predefined retrieval strategies, case retrieval is custom-tailored and dynamically changed for a particular domain and specific application. This flexibility is accomplished by using context in similarity-based retrieval, deploying the *Telos* [34] representation language which treats objects and attributes uniformly. Flexibility is needed to support the following aspects of $\mathcal{T}A$**3** :

- Depending on the resources available the system allows for changing the classification accuracy.

[1] Name $\mathcal{T}A$**3** was created as a phonetic equivalent to the name of the highest mountain range in Slovakia, Tatry.

- Based on a given task, the system allows for changing the retrieval objective in the following ways:

   (i). *recall-oriented retrieval* guarantees that all relevant cases are retrieved, although some irrelevant ones may also be included, and

   (ii). *precision-oriented retrieval* assures that only relevant cases are be retrieved, although some may be missed.

- The system allows for imprecise query specification, which enables the retrieval of cases even if the query is not specified completely. Moreover, similar cases are retrieved in addition to exact matches.

- Depending on the result of retrieval, the system allows for manual or automatic query modifications through context restriction or relaxation, as well as for query-by-example and query-by-reformulation.

These features are desirable for supporting a variety of different domains and tasks since the task requirements may change over time and individual users may have different preferences. In addition, the classification process must be efficient, i.e., the system must scale up as more cases are acquired.

The remainder of the paper is organized as follows: Section 2 describes the classification problem and presents various approaches to the problem. Section 3 introduces an approach to context-based relevance which is used to define similarity-based retrieval and case-based classification. Context manipulation is used for controlling the classification accuracy. Section 4 presents performance evaluation results and shows how controlling the relevance of retrieved cases can affect classification accuracy. Section 5 presents concluding remarks.

2 Classification

This section defines the classification task, discusses traditional approaches to the problem and presents a case-based approach to classification.

2.1 *Classification problem definition*

In a classification system, there is usually a fixed set of classes to which a given object can be assigned. Classification systems are generally trained first, i.e., already classified objects are presented, and the system induces knowledge from previous examples. For example, neural networks change connection weights, decision tree algorithms generate rules and case-based systems remember individual instances. The goal of such systems is to learn a correct classification for given examples. Obviously, the system must be able to generalize from these examples, so that novel examples are classified correctly with satisfactory accuracy.

Generally, objects in a classification system are represented as a collection of properties – attributes or features. In real-world domains, objects may or may not be represented properly and/or error-free: some attribute values might be missing or imprecise, or there may be irrelevant attributes present and relevant attributes missing. Such domains are called imperfect.

In conventional data analysis, objects are clustered into classes based on a distance or similarity measure [44]. The similarity between two objects is represented as a number – the value of a similarity function applied to symbolic descriptions of objects. Thus, the similarity measure is context free. Note that such methods fail to capture the properties of a cluster as a whole and are not derivable from properties of individual entities [44].

Learning algorithms can be divided into two approaches: supervised and unsupervised. Supervised learning algorithms use inductive learning. The learning program is given examples of the form (x_i, y_i) and it is supposed to learn a function f, such that $f(x_i) = y_i$ for all i. Typically, x_i is a composite description of some object, situation or event and y_i is a simpler description. In classification, x_i represents a problem and y_i corresponds to a solution or a class. Thus, supervised learning involves learning a function f that assigns each x_i into an appropriate class. The learning goal is to find a function f that captures general patterns in the training data, so that f can be applied to predict y_i values for a new, previously unseen, x_i values.

Unsupervised learning covers clustering and discovery. Here, the learning program is given a collection of x_i values and asked to look for regularities. How regularities are represented varies. Most of the programs search for some form of clusters of x_i values. Discovery programs look for more complex relationships among the x_i values.

Supervised learning algorithms for classification can be described as follows [42]:

> Given a sequence of training examples $\langle \bar{x}_1, c_1\rangle, \ldots, \langle \bar{x}_n, c_n\rangle$, where $\langle \bar{x}_i, c_i\rangle$ is a pair consisting of an object description $\bar{x}_i$ and a proper classification c_i, the classification system must learn the mapping $classify : X \rightarrow C$, where $X = \{\bar{x}_1, \bar{x}_2, ...\bar{x}_n\}$ is the space of objects descriptions and $C = \{c_1, c_2, ..., c_n\}$ is the space of possible classes.

Case-based classification systems classify instances based on their similarity to stored cases:

> Given a new problem (a partial case $\mathcal{C}$), the system retrieves a set of cases $SI = \{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ from a case base, Δ, and classifies the new problem based on the case solutions in SI.

The main task of such a system is to identify the similarity of a problem to cases in a case base. Since the stored cases are used for classifying the new case, the more similar they are to the problem, the more accurate the classification can be. Similar cases are ordered based on their closeness to the problem description. Then, if not all retrieved cases belong to the same class, the class for the problem is determined using this partial order, preferring closer matches.

2.2 *Overview of some existing approaches to classification*

Traditional approaches to classification involve generating a set of rules, based on induction from training examples. The requirement for such algorithms is that created rules correctly classify known, i.e., training, examples and perform well on novel (i.e., test) examples [37]. Next we describe some relevant classification systems, as well as systems that use case-based reasoning or similarity-based retrieval. Classification systems are relevant due to the fact that we apply case-based reasoning to a classification problem and similarity-based retrieval algorithms are relevant because this type of retrieval is the basis of our approach.

ID3 [38] is a classification system based on a decision tree algorithm. Using the induction from training examples, ID3 generates a decision tree – a classification rule that examines the values of some attributes of an object in order to assign it into a proper class.

AC^3 [44] is a classification algorithm based on attribute-based conjunctive conceptual clustering – a way of grouping objects into conceptually simple classes. Using this method, a set of objects forms a class only if it can be described by a conjunctive concept involving relations on object attributes.

AUTOCLASS II [9] is an induction algorithm used to discover classes from databases based on Bayesian statistical techniques. The system determines the number of classes, their probabilistic descriptions and the probability that each object is a member of a given class. This allows for making each and every attribute potentially significant as well as assigning objects to different classes. The system also allows for identifying hierarchies of attributes, selecting common attributes and distinguishing attributes between classes.

IB1 [2, 5] is a nearest-neighbor, instance-based learning system used for classification and for control tasks. Based on a problem description, IB1 retrieves the k-nearest neighbors and uses them to supply a solution to a given problem. Since all attributes are used during retrieval, the system's performance decreases quickly with the number of irrelevant attributes. Such a performance degradation of IB1 can be solved either by data pre-processing that would remove all irrelevant attributes or by the use of an intelligent, selective partial matching algorithm. The latter approach is similar to m_of_n concepts in machine learning [35], i.e., the system considers only $m < n$ attributes during retrieval. There are various methods used to decide m and Ortega [35] presents an evaluation of system performance for various m. In addition to deciding the size of m, the approach described in this paper allows for specifying which attributes to exclude and for posing additional constraints on attribute values, if desired.

CB1 [18] is a PAC (Probably Approximately Correct) learning, case-based classification algorithm designed for general purpose learning. Even though it is relatively inefficient, considering the sample complexity, it is an interesting step towards a computational learning theory for CBR systems.

PROTOS [36] is a a case-based system used for heuristic classification in the clinical audiology domain. The system uses exemplar-based learning [30, 6, 7, 3], a method especially appropriate for domains lacking a strong domain theory. In this system, classification is combined with explanation. Explanation is used for justifying case classification and for determining similarity between two cases.

An incremental retrieval system, I-CBR, is presented in [11]. For this case-based reasoning paradigm, the user specifies a case skeleton and attempts to retrieve all cases similar to it. I-CBR partitions case attributes into two groups: the ones that are known immediately and the ones that need some extra effort to specify. During the retrieval process, the user specifies known attributes and the system returns a pool of possible case candidates. From the set of unknown attributes the most discriminating one is selected and the user is queried for its value. Thus, the initial pool of retrieved cases is processed to decrease its size to eliminate less-similar cases which increases classification accuracy.

The most appealing feature of the similarity-based retrieval approach proposed in this paper is that it allows for retrieving relevant information even without a complete and precise query or a perfect match. Such retrieval is an essential part of a case-based classification system. After retrieving relevant cases, i.e., cases that are similar to a given problem, the case-based classification system can adapt previous solutions and use them to predict the class for a current case.

3 The $\mathcal{TA}$3 Case-Based Classification System

In this section we present our proposed case-based classification theory, using flexible relevance assessment [20, 24] – a fundamental component of the $\mathcal{TA}$**3** case-based reasoning system. We define relevance in terms of context and similarity, and we also show how context modifications can be used to dynamically control the accuracy of the classification process.

Although it has been acknowledged that context plays a central role in determining similarity, previous works on similarity measures for case-based reasoning generally assume that context is implicit in the case representation, or is acquired through machine learning techniques [8]. We define context as a parameter of a similarity relation and demonstrate the monotonicity of case retrieval. We show how this property can be used for controlling the classification accuracy.

Definition 3.1 (A case) *A* case*, $\mathcal{C}$, is a representation of a real world experience or entity in a particular representation scheme. A case will be represented as a finite set of attribute/value pairs (descriptors):*

$$\mathcal{C} = \{\langle a_0 : V_0 \rangle, \langle a_1 : V_1 \rangle, \ldots, \langle a_n : V_n \rangle\},$$

where $A_i = \langle a_i : V_i \rangle$ is an attribute/value pair.

Using the information about the usefulness of individual attributes and information about their properties, attributes are grouped into two or more Telos-style categories [34]. This enhancement of case representations will be used during the retrieval process to increase the accuracy of classification and flexibility of retrieval, and to improve system performance. In classification tasks, each case has at least two categories: problem description and class. The problem description characterizes the problem and the class gives a solution to a given problem.

We shall explain the functionality of the proposed system using an example from a simulation of a servo system that involves a servo amplifier, a motor, a lead screw/nut and a sliding carriage. A servo system is an automatically operating device for the regulation of a system's variable(s) actuated by the difference between the actual and a desired value of such a variable. The servo domain covers a non-linear phenomenon – predicting the rise time of a servo-mechanism in terms of two gain settings and two choices of mechanical linkages. The task in a given servo domain is to compute the rise time of a specific servo-mechanism. Thus, the problem is described by the servo characteristics that include *motor, screw, pgain* and *vgain*. The solution, i.e., the class, is a rise time or the time required for the system to respond to a step change in a position set point (more details are presented in Section 4.2). Here, the case is a collection of five attribute/value pairs, as illustrated in Table 1.

Table 1: A servo domain case example.

MOTOR:	E
SCREW:	D
PGAIN:	4
VGAIN:	5
RISE_TIME:	0.28125095

Attributes in a case are grouped into three categories: *motor* and *screw* form the first category; *pgain* and *vgain* comprise the second category; and the last category is the class *rise time*. Attribute membership in a particular category is defined either using domain knowledge or using a knowledge discovery algorithm [12, 23, 28]. This grouping allows for ascribing different constraints to different groups of attributes, and the process of retrieving relevant cases can be described as a constraint satisfaction process [45, 14, 31]. Categories allow for improved system performance, as will be shown later.

An explicitly specified context allows for less strict matching than equivalence. The goal is to retrieve not only exact matches, but partial matches (similar cases) as well. In short, context is a parameter of a relevance relation which maps a case base onto a set of relevant (in terms of context) cases.

Definition 3.2 (A context) *A* context, Ω, *is defined as a finite set of attributes with associated constraints on the attribute values:*

$$\Omega = \{\langle a_0 : CV_0 \rangle, \ldots, \langle a_k : CV_k \rangle\},$$

where a_i is an attribute name and constraint CV_i specifies the set of "allowable" values for attribute a_i. A context specifies the important (or relevant) attributes and how "close" an attribute value must be in order to satisfy the context. A constraint set CV_k must be a subset of the domain D_k for an attribute a_k: $CV_k \subseteq D_k$.

A possible context in a servo domain is defined in Table 2. A context allows for controlling what can and what cannot be considered as a partial match. As a result, context controls the classification accuracy in case-based classification, and classification will not degrade with irrelevant attributes present. In addition, by modifying the context, the system may flexibly change the tradeoff between accuracy and speed [24].

Table 2: A servo domain context example.

MOTOR:	$\{$D, E$\}$
SCREW:	$\{$C, D, E$\}$
PGAIN:	$\{4, 5\}$
VGAIN:	$\{5\}$
RISE_TIME:	$\{x\}$, where x is a real number

For the purpose of performance evaluation, measured in terms of precision and recall, we define relevance using context and similarity. There is little commonality between the definitions of relevance used in different systems [17]. The most common ground is reached when relevance is defined as something useful. Our definition of relevance is based on the assumption that the more similar (with respect to a given context) the problem description is to a case in the case base, the higher the accuracy for case-based classification, and thus the more relevant the case.

3.1 *Similarity assessment*

In the proposed theory, similarity is considered as a relation between cases with respect to a specified context. It is defined to supplement equivalence by allowing for partial matches. Two cases are considered similar if they satisfy a given context.

Definition 3.3 (Satisfiability) *A case $\mathcal{C}$* satisfies *a context Ω, denoted $sat(\mathcal{C}, \Omega)$, if and only if for all pairs $\langle a_i : CV_i\rangle \in \Omega$, there exists a pair $\langle a_i : V_i\rangle \in \mathcal{C}$ such that V_i is in CV_i:*

$$sat(\mathcal{C}, \Omega) \ iff \ \forall a_i \langle a_i : CV_i\rangle \in \Omega \ \rightarrow \ \exists V_i \langle a_i : V_i\rangle \in \mathcal{C} \ \wedge \ V_i \in CV_i.$$

We say that the context Ω is satisfiable *if and only if there exists at least one case in a case base that satisfies it:*

$$\Omega \ is \ satisfiable \ iff \ \exists \mathcal{C} \in \Delta \ : \ sat(\mathcal{C}, \Omega).$$

Using the servo domain example, the *motor* attribute is satisfied in the given context since the value E is an element of the constraint defined in the context *{D, E}*. Other attributes are processed similarly.

Since the context affects the class into which the input problem is assigned, context specification is vital for the correct answer, i.e., for accurate classification. In general, the context can be specified using the following scenarios:

(i). The user has enough domain knowledge to specify the context (query-by-example can be used as an initial context specification).

(ii). The user specifies the task (s)he wants to solve and the system selects an appropriate context (task-based retrieval).

(iii). First, a knowledge discovery algorithm is used to locate relevant attributes and attribute values. Then, a user forms a context for retrieval using this information. This approach is suitable for novice users.

(iv). The user submits an ad hoc query, reviews the resulting solution, and then iteratively modifies and resubmits the query (retrieval by reformulation). This approach is best suited for repository browsing.

Using the context, the input problem and cases in a case base are interpreted and their similarity is assessed.

Definition 3.4 (Similarity) *A case $\mathcal{C}_1$ is* similar *to a case $\mathcal{C}_2$ with respect to a given context Ω, denoted: $\mathcal{C}_1 \sim_\Omega \mathcal{C}_2$, if and only if both $\mathcal{C}_1$ and $\mathcal{C}_2$ satisfy context Ω:*

$$\mathcal{C}_1 \sim_\Omega \mathcal{C}_2 \;\leftrightarrow\; sat(\mathcal{C}_1, \Omega) \wedge sat(\mathcal{C}_2, \Omega).$$

Table 3: A servo domain case example.

MOTOR:	C
SCREW:	C
PGAIN:	4
VGAIN:	5
RISE_TIME:	0.32456011

Using the case and the context defined in Tables 1 and 2, and the case defined in Table 3, we see that the two cases are not similar with respect to the given context because the *motor* attribute value of the second case does not satisfy the constraint, specified in the context for that attribute. If the *motor* value is changed from C to D (or the context is changed to *motor: {C, D, E}*) then the two cases would be similar with respect to the given context, since they would both then satisfy the context.

For the purpose of case retrieval, the similarity relation maps a context Ω and a case base Δ onto the set of cases SI in the case base that satisfy the context.

Definition 3.5 (Retrieval function) *Given a case base Δ and a context Ω, the function retrieve returns a set of similar cases $SI \subseteq \Delta$, such that all cases in SI are similar to each other with respect to Ω, i.e., if they satisfy the given context. The retrieval function is complete in the sense that it returns all relevant cases and only relevant cases.*

Input:	Ω, Δ
Output:	*SI*
Description:	retrieve $(\Omega, \Delta) = SI$, *where*
	$\forall \mathcal{C} \in \Delta, \mathcal{C} \in SI \;\; iff \;\; sat(\mathcal{C}, \Omega)$

The complexity of this task is characterized by the number of required comparisons which is, in the worst case, proportional to the size of the context and the size of a given case base: i.e., $\mid \Omega \mid \times \mid \Delta \mid$. Case base organization may improve the efficiency of performing the task by limiting the search space of the case base.

Using the presented theory, we define a case-based classification algorithm, CBC, as part of the $\mathcal{T}A$**3** case-based classification system as follows [26]:

Definition 3.6 (Case-based classification) *The input to a classification problem comprises a case base Δ, a context Ω and a problem $\mathcal{C}_p$ (also called an input case, i.e., a case without a solution or appropriate class). An output is a class c_k for a problem C_p, defined as follows:*

$$class(\mathcal{C}_p, c_k) \leftrightarrow (\exists \mathcal{C}_r \in \Delta \, : \mathcal{C}_p \sim_\Omega \mathcal{C}_r \, \wedge \, \langle \mathcal{C}_r, c_k \rangle),$$

where $class(\mathcal{C}_p, c_k)$ denotes c_k is the class for case $\mathcal{C}_p$.

Using the definitions of similarity (Definition 3.4) and CBC (Definition 3.6), it can be shown that by manipulating the context it is possible to control the quantity and quality of retrieved cases. Thus, context serves as a form of bias. Since CBC is based on the idea that similar past cases can be used to determine the class for a given problem, context manipulation allows for changing the classification for the problem. It should be noted that the context cannot be arbitrarily chosen. For a correct answer, it must include attributes and constraints that would allow for organizing cases into clusters based on classification, i.e., it must include attributes predictive of a correct class. During the classification process, the following three situations may occur:

(i). No relevant cases are retrieved – the system cannot classify a given problem.

(ii). All relevant cases belong to the same class – the input problem is assigned to the class suggested by retrieved cases.

(iii). Retrieved cases belong to multiple classes – the system assesses the relevance of individual cases to produce a partial order and assigns the input problem into the class of the most relevant case. If this is not possible, due to equally relevant cases belonging to different classes, the majority class is selected. If this step fails as well, i.e., there are at least two groups of equally relevant cases belonging to different classes, then the system returns all possible classifications with an “Undecided” answer.[2]

3.2 *Context relaxation and restriction.*

Explicitly defined context allows for controlling the closeness of retrieved cases. If too many or too few relevant cases are retrieved using the initial context, then the system automatically transforms the context or a user may modify it manually. There are two possible transformations: *relaxation* – to allow for retrieving more cases and *restriction*

[2]Clearly, in some domains it might be possible for a case to belong to different classes. However, some applications (e.g., character recognition) require strict recognition of the class.

– to allow for retrieving fewer cases. These context transformations are a foundation for supporting iterative retrieval and browsing. Based on the introduced theory, this section discusses specific examples of relaxation and restriction transformations, with an emphasis on automatic methods.

Definition 3.7 (Relaxation) *A context Ω_1 is a* relaxation *of a context Ω_2, denoted $\Omega_1 \succ \Omega_2$, if and only if the set of attributes for Ω_1 is a subset of the set of attributes for Ω_2 and for all attributes in Ω_1, the set of constraints in Ω_2 is a subset of the constraints in Ω_1. As well, contexts Ω_1 and Ω_2 are not equal.*

$$\Omega_1 \succ \Omega_2 \ iff \ \forall \langle a_i : CV_i \rangle \in \Omega_1, \ \exists \langle a_i : CV_j \rangle \in \Omega_2 \ : \ CV_i \supseteq CV_j \ \wedge \ \Omega_1 \neq \Omega_2.$$

Definition 3.8 (Restriction) *A context Ω_1 is a* restriction *of a context Ω_2, denoted $\Omega_1 \prec \Omega_2$, if and only if Ω_2 is a relaxation of Ω_1.*

$$\Omega_1 \prec \Omega_2 \ iff \ \Omega_2 \succ \Omega_1.$$

As pointed out by Gaasterland in [14], the relaxation technique can advantageously be used for returning answers to a specific query as well as for returning related answers. Without such automatic relaxation, users would need to submit alternative queries. The restriction technique works analogously, but is used mainly for controlling the amount of returned information, keeping it to manageable levels for the user. Since the search for relaxed or restricted contexts can be infinite, there must be a mechanism for controlling it, either by user intervention via user preferences or by other means.

Context relaxation and restriction are used during retrieval to control the quantity and quality, or closeness, of cases considered relevant. Thus, when modifying the context, the system may return an approximate answer quickly or may spend more resources getting an accurate answer. An approximate answer can be iteratively improved, so that the change between an approximate and an accurate answer is continuous.

Theorem 3.1 *If Ω_1 is a relaxation of Ω_2 then all cases that satisfy Ω_2 also satisfy Ω_1:*

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_2) \wedge \Omega_1 \succ \Omega_2) \rightarrow sat(\mathcal{C}, \Omega_1).^3$$

Corollary 3.1 *If Ω_1 is a restriction of Ω_2 then the set of cases that satisfy Ω_1 is a subset of the set of cases that satisfy Ω_2.*

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_1) \wedge \Omega_1 \prec \Omega_2 \ \rightarrow \ sat(\mathcal{C}, \Omega_2)).$$

[3] The proofs of Theorem 3.1 and Corollary 3.1 are available in Appendix.

Considering Definition 3.7, there are two possible implementations of context relaxation – *reduction* and *generalization*. *Reduction* removes constraints by reducing the number of attributes required to match: given m_of_n matching, the required number of attributes is reduced from m to p, where $0 < p < m \leq n$. *Generalization* is a context transformation that relaxes the context by enlarging the set of allowable values for an attribute. This technique has been used in [10, 15] for relaxation of database queries.

Consider the example presented in Table 3. Applying reduction to the first category, attributes *motor* and *screw*, changes the requirement that both attributes must satisfy the constraint to the requirement that if either of the two attributes satisfies the requirement then the whole category would satisfy it.

Considering the example presented in Table 2, the two cases would match if the constraint on the *motor* attribute is relaxed to include value C, i.e., *motor: {C, D, E}*.

Analogically, considering Definition 3.8 there are two possible implementations of context restriction – *expansion* and *specialization*. *Expansion* is a context transformation that strengthens constraints by enlarging the number of attributes required to match: given m_of_n matching, the required number of attributes is increased from m to p, where $0 \leq m < p \leq n$. Thus, expansion is the inverse operation to reduction. *Specialization* strengthens constraints by removing values from a constraint set for an attribute. This may lead to a decreased number of cases that satisfy the resulting context.

3.3 *Monotonicity of context-based similarity*

Monotonicity is an important property of a flexible retrieval system and can advantageously be used in case-based classification – as more constraints are put in the query, fewer cases satisfy it. It should be noted that the resulting set of retrieved cases is equal to or subset of the original set of cases. We will show that the retrieval function defined using context-based similarity is monotonic with an inverse monotonicity – more constraints imply less retrieved cases and vice versa.

Theorem 3.2 (Monotonicity) *The retrieval function based on context-based similarity assessment is monotonic. More precisely, given a case base* Δ *and contexts* Ω_1, Ω_2*:*

$$(\Omega_1 \succ \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \supseteq retrieve(\Omega_2, \Delta).$$

$$(\Omega_1 \prec \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \subseteq retrieve(\Omega_2, \Delta).$$ [4]

A graphical illustration for Theorem 3.2 is presented in Figure 1. Assume a graphical matching function, where a case satisfies a given context if and only if the whole context overlaps with a case. Both example **A** and example **B** use the same rectangles **a**, **b** and **c**. In example **A**, all three rectangles are similar with respect to context Ω_1. However, in the example **B**, when the context Ω_2 is more restrictive (graphically larger, i.e., $\Omega_1 \succ \Omega_2$), only rectangles **a** and **c** are similar – rectangle **b** does not qualify since it matches only part of a given context.

[4] The proof of Theorem 3.2 is available in Appendix.

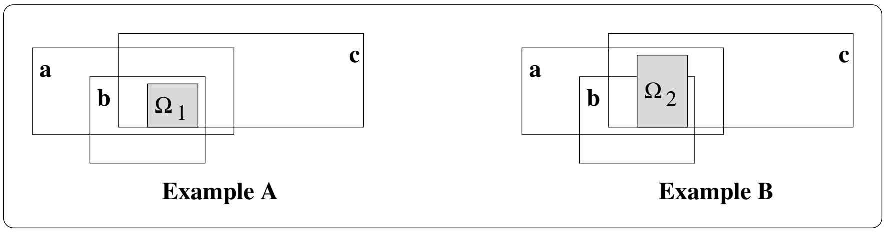

Figure 1: Monotonicity.

The usefulness of the monotonicity feature during the retrieval process was reported in [41, 33, 20]. On the one hand, if not enough cases have been retrieved from the case base, the initial context may be relaxed to retrieve additional "less similar" cases. On the other hand, if too many cases are retrieved, the initial context may be restricted to lower the number of retrieved cases, as depicted in Figure 1. Restricting the context increases the similarity between cases since more attributes are required to match.

4 $\mathcal{TA}$3 Performance Evaluation

The goal of the performance evaluation presented here is to show that the theoretical basis for the proposed approach to case-based classification is both effective and efficient. We are interested in two aspects of system performance: *accuracy of classification* and *scalability*. Classification accuracy is the most important measure of system functionality. We use this measure to support the claim of usability of the presented theory. Many times neglected, yet important, is the dependency of the system on the case base size and the problem complexity.

Performance evaluation must be conducted with a special care on the used data set and on the selected measures [2]. It is not only important to know what to measure when evaluating a system, but also to know how to interpret the results. There are many accepted benchmarks used in different fields and there are numerous evaluations available. Yet, many times wrong conclusions are made or even worse, useless measures are considered.

System performance should be determined with respect to *task* and *time*. The first characteristic reveals if the system can perform the specified task (e.g., retrieve all relevant cases from the case base). Usually, task-performance of a classification system is measured by evaluating its *accuracy*, i.e., percentage of correct classifications. The second characteristic measures how long it takes for the system to perform the specified task. In addition, scalability measures task/time performance dependability on the case base size.

It should be noted, that accuracy is not only system dependent but is also strongly domain dependent [2]. Kononenko and Bratko [32] present a fair evaluation criterion to measure classification accuracy. The motivation for the study is that a simple comparison of classification accuracy might be misleading. For example, 80% accuracy in a perfect domain with 2 classes can be achieved trivially whereas 40% accuracy in a domain with 26 classes (and possible missing information) might be hard to achieve. Moreover, accuracy depends on the probability of classes. If 90% of the cases fall into one class, then it is trivial to achieve 90% classification accuracy.

Based on the proposed theory, we have applied the research prototype $\mathcal{T}A\mathbf{3}$ to several real-world domains:[5] a medical domain ($\mathcal{T}A\mathbf{3}_{IVF}$) [29, 28], a servo-mechanism ($\mathcal{T}A\mathbf{3}_{Servo}$) domain, a robotic domain ($\mathcal{T}A\mathbf{3}_{Robot}$) [25, 27], a letter classification domain ($\mathcal{T}A\mathbf{3}_{Letter}$) [23] and a small software case base ($\mathcal{T}A\mathbf{3}_{SR}$) [22]. Each domain has different characteristics and thus allows for a more extensive performance evaluation. Retrieved cases were used for predicting unknown values for attributes, i.e., for classifying a case with respect to the given attribute. In some domains this required classification onto a finite set of classes (e.g., letter recognition); in other domains, there is an infinite number of possible classes (e.g., IVF). We used a simple cross-validation (leave-one-out method) and compared results to actual cases to avoid subjective bias in assessing their relevance. Prediction is successful only if relevant cases are used as a starting point. As a result, the accuracy of classification (i.e., percentage of correctly classified problems) depends on the retrieval capability of $\mathcal{T}A\mathbf{3}$. Therefore, the better the similarity-based retrieval, the better the classification accuracy.

The main objective of the evaluation presented here is to test task performance [26]. Time performance has been reported elsewhere [19].

4.1 *Problem 1: IVF case base*

The in-vitro fertilization (IVF) domain involves medical records about patients. The available case base consists of 788 cases and 55 attributes per case (after confidential information has been removed). Out of 788 cases, 149 are clinically successful pregnancies, 10 are pregnancies with ovarian hyper-stimulation complication, 12 are pregnancies ended by abortion, 4 are ectopic pregnancies and 632 are unsuccessful pregnancies (there is not information about pregnancy for 7 cases).

According to our experience, the two most common tasks performed by physicians are classification and knowledge mining. Here we report on using $\mathcal{T}A\mathbf{3}$ to support the classification task, namely suggesting a treatment and predicting pregnancy outcome.

Having initial information about the patient (age, diagnosis of infertility, previous treatment history, etc.), the task is to retrieve similar patients from the case base and to *suggest a treatment* for the current patient where the probability of a successful pregnancy is increased. This involves suggesting the day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG). After retrieving similar cases, cases with pregnancy are considered as successful examples and cases without pregnancy as negative cases. DAY_HCG and NO_HMG serve as classes and since both have continuous values, this constitutes classification onto a continuous class, i.e., classification onto an infinite number of possible classes instead of classification onto a finite number of discrete classes.

After the initial treatment is completed, additional attributes are available (e.g., response to the current treatment). The task here is to *predict the outcome* of the whole treatment,

[5] $\mathcal{T}A\mathbf{3}_X$ denotes application of $\mathcal{T}A\mathbf{3}$ into a domain X.

i.e., to predict pregnancy outcome. This involves determining values for the following attributes: pregnancy, abortion, ovarian hyper-stimulation syndrome and ectopic pregnancy. This process can be described as a classification onto discrete classes with possible values "yes" or "no".

In order to statistically evaluate classification, we have conducted a series of tests. Using a simple cross-validation (leave-one-out method and random case selection with results averaged over 20 trials), we obtained the results presented in Table 4. For predicting pregnancy outcome, the system was able to achieve 60.6% accuracy using only the first eight attributes and 71.2% accuracy using a series of estrogen values in addition (15 more attributes).

Table 4: IVF domain – Statistical results with 95% confidence level. Absolute and relative errors between suggested and actual values for the treatment, namely day of human chorionic gonadotrophin administration (DAY_HCG) and the number of ampoules of human menopausal gonadotrophin (NO_HMG).

Attribute	*Absolute Error*	*Relative Error*
NO_HMG	3.2 ± 1.98	0.15 ± 0.09
DAY_HCG	0.9 ± 0.558	0.07 ± 0.04

4.2 *Problem 2: Learning control*

The servo domain[6] was previously used for testing various machine learning algorithms [39, 40]. This data set covers a non-linear phenomenon – predicting the rise time of a servo-mechanism in terms of two (continuous) gain settings and two (discrete) choices of mechanical linkages. Thus, the rise time is an attribute with an infinite number of values, i.e., there is an infinite number of classes.

The data set consists of 167 instances with five attributes: *motor, screw, pgain, vgain* and the class *rise time*. All attributes in the case are divided into three Telos-style categories: category 1 – *motor* and *screw*, category 2 – *pgain* and *vgain*, and category 3 – *rise time*. Even though this domain is neither complex nor large, it is of interest to us since several machine learning algorithms have previously been evaluated and compared with respect to it. Table 5 summarizes results for $\mathcal{T}A\mathbf{3}_{Servo}$ and other machine learning techniques, obtained by using simple cross-validation (leave-one-out method) and averaged over 10 and 60 random trials respectively.

$\mathcal{T}A\mathbf{3}_{Servo\,1}$ used value relaxation where only the *motor* attribute was relaxed and only to the left side (e.g., value B would be relaxed to $\{A, B\}$). $\mathcal{T}A\mathbf{3}_{Servo\,2}$ results were obtained by relaxing only the value for *motor* attribute but in both directions (e.g., value B would be relaxed to $\{A, B, C\}$). It should be noted that relaxing A and E in both scenarios would

[6] Data description and performance results obtained by other machine learning systems applied to the servo domain are available at ftp://ics.uci.edu/pub/machine-learning-databases/servo/.

be the same – $\{A, B\}$ and $\{D, E\}$. In situations where the initial relaxation did not yield a result, additional methods are used. First, reduction for *motor* and *screw* is tried, i.e., all cases that match *motor*, *screw* or both are retrieved. Second, reduction for *pgain* and *vgain* is applied.

Table 5: Absolute and relative errors on servo-data in terms of a difference between actual and computed value of the rise time of the servo-mechanism (IBL - instance-based learning). $\mathcal{T}A\mathbf{3}_{Servo}$ results are presented for 60 random trials, with 95% confidence levels.

Method	*Absolute Error*	*Relative Error*
Guessing mean	1.15	1.00
Instance-based	0.52	0.26
Regression	0.86	0.49
Model trees	0.45	0.29
Neural nets	0.30	0.11
Regression+IBL	0.48	0.20
Model trees+IBL	0.30	0.17
NN+IBL	0.29	0.11
$\mathcal{T}A\mathbf{3}_{Servo\,1}$	0.118 ± 0.042	0.073 ± 0.032
$\mathcal{T}A\mathbf{3}_{Servo\,2}$	0.09 ± 0.035	0.056 ± 0.012

Even though direct error comparison can be used for performance evaluation and system comparison, it is not sufficient in general. In the data presented in Table 5 there is no information on the confidence in the results, nor is there any information about the significance of the errors or about the confidence intervals. In other words, statistical evidence is missing. Thus, we have also conducted a statistical evaluation of our system (confidence intervals for $\mathcal{T}A\mathbf{3}_{Servo\,1}$ and $\mathcal{T}A\mathbf{3}_{Servo\,2}$). The results show that it is advantageous to have an option to change the relaxation technique and also that the servo domain has better informativeness then the IVF domain since higher accuracy can be achieved with the same system.

4.3 *Problem 3: Inverse kinematic task*

The inverse kinematic task (IKT) involves predicting the joint angles, φ_1, φ_2 and φ_3, for a three-link spherical angular robot as depicted in Figure 2, assuming given desired end-effector coordinates. This task can be characterized as a classification into a continuous class, namely classification into an infinite number of real values (the triplet φ_1, φ_2 and φ_3).

IKT is a complex task which might be computationally intractable for complex systems. In such situations, other methods are used to approximate the solution. In our study we used an existing small robot as an example [27]. The robot's parameters are presented in Table 6, where L_1, L_2, L_3 denote the lengths of the links (robotic arms) and $\varphi_1, \varphi_2, \varphi_3$ are the ranges for joint angles.

For the given architecture, the following equations are used to compute the end-effector

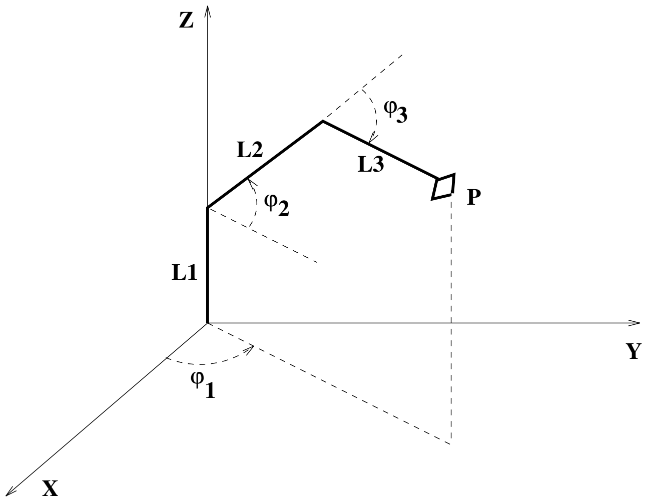

Figure 2: Spheric angular robotic system.

Table 6: Robot characteristics. L_i are robot links; φ_i are joint angles between links.

L_1	L_2	L_3	φ_1	φ_2	φ_3
0.203 m	0.178 m	1.178 m	$(-90^o, 90^o)$	$(-35^o, 145^o)$	$(-150^o, 0^o)$

coordinates P (a forward kinematic task):

$$P_x = L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \cos\varphi_1 \cos(\varphi_2 + \varphi_3) \quad (1)$$

$$P_y = L_2 \cos\varphi_1 \cos\varphi_2 + L_3 \sin\varphi_1 \cos(\varphi_2 + \varphi_3) \quad (2)$$

$$P_z = L_1 + L_2 \sin\varphi_2 + L_3 \sin(\varphi_2 + \varphi_3) \quad (3)$$

The inverse kinematic computation does not necessarily yield only one solution. Moreover, complex kinematic structures might not have a known analytical solution at all. In such situations, some optimality criteria need to be defined and used for computation. The other possibility is to use previous knowledge and compute only increments from the current position. Nevertheless, the following are equations for inverse kinematic in our architecture:

$$\varphi_1 = \arctan \frac{P_y}{P_x} \quad (4)$$

$$\varphi_2 = \arctan \frac{P_z - L_1}{\sqrt{P_x^2 + P_y^2}} \pm \arccos \frac{L_2^2 - L_3^2 + P_x^2 + P_y^2 + (P_z^2 - L_1)^2}{2L_2\sqrt{P_x^2 + P_y^2 + (P_z^2 - L_1)^2}} \quad (5)$$

$$\varphi_3 = \pm \left[\pi - \arccos \frac{L_2^2 + L_3^2 - P_x^2 - P_y^2 - (P_z^2 - L_1)^2}{L_2 L_3}\right] \quad (6)$$

One obvious need for the inverse kinematic task is during planning – a robot's end-effector is at point X and should be moved to point Y. For example, in welding the usual task is to follow a particular curve during the process. Thus, the inverse kinematic must be computed reasonably fast.

The presented results are for a uniformly generated data set which consists of 2,000 instances, where each instance is represented by nine attributes containing real values: three lengths describing robotic arms, three attributes describing a desired end-effector position and three joint angles. Individual attributes are grouped into three Telos-style categories: robot's parameters, end-effector position and joint angles.

Table 7: Robotic domain – Statistical results with 95% confidence level. Absolute and relative errors between actual and computed values for three joint angles.

Method	*Absolute Error*			*Relative Error*		
	φ_1	φ_2	φ_3	φ_1	φ_2	φ_3
$\mathcal{T}A\mathbf{3}_{Robot\,1}$	0.486 ± 0.213	0.0	0.0	0.006 ± 0.003	0.0	0.0
$\mathcal{T}A\mathbf{3}_{Robot\,2}$	0.0	2.5 ± 1.096	0.0	0.0	0.027 ± 0.012	0.0

During evaluation we used the leave-one-out method and we computed significance intervals and variances from the average values over 20 random trials, as illustrated in Table 7. The uniform generation of cases is especially useful for off-line learning to guarantee a reasonable answer in any possible situation. Another possibility is to "observe" actual cases while the robot is performing normal tasks. The problem with this approach is that it requires on-line operation and that the working space may not be represented evenly within the case base. Thus, the system will be able to work in similar situations to the ones observed during learning, but the performance will degrade if there is a change in the robot task. On-line learning can be a useful experience added to the off-line generate knowledge base, which can be used for generalizations and for knowledge-based adaptation of retrieved cases.

During the experiment, different criteria for selecting relevant cases were tested. In general, case representation was organized into three Telos-style categories. Only reduction was used in $\mathcal{T}A\mathbf{3}_{Robot\,1}$, i.e., for two categories, only m_of_3, $1 \leq m \leq 3$, attributes were required to match. $\mathcal{T}A\mathbf{3}_{Robot\,2}$ used attribute value relaxation first, and only when this failed was reduction performed.

4.4 *Problem 4: Character recognition domain*

The letter classification task has previously been used for machine learning algorithm testing [13, 2].[7] The data set consists of 20,000 instances described by 17 attributes: letter, horizontal and vertical position of a box, width and height of a box, total number of pixels,

[7] Data description and performance results obtained by other machine learning systems applied to the letter recognition domain are available at ftp://ics.uci.edu/pub/machine-learning-databases/letter-recognition/.

etc. All attributes are grouped into five different categories. The objective is to classify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images are based on 20 different fonts and each letter within these 20 fonts is randomly distorted to produce a data set. Each stimulus was converted into 16 primitive numerical attributes, statistical moments and edge counts, which were then scaled to fit into a range of integer values from 0 through 15.

Table 8: Character recognition domain – Statistical results with 95% confidence level. Percentage of correct letter classifications over 20 random trials using 20,000 and 2,000 cases (accuracy A and B).

Method	*Accuracy A*	*Accuracy B*
Backprop.	—	$81.9 \pm 0.6\%$
IB1	$95.7 \pm 0.4\%$	$81.7 \pm 0.7\%$
CN2	$87.9 \pm 0.8\%$	$68.7 \pm 1.0\%$
C4	$86.4 \pm 0.7\%$	$67.4 \pm 0.8\%$
Genetic Alg.	82.7	—
$\mathcal{T}A\mathbf{3}_{Letter\,1}$	$90\% \pm 0.1461$	$80\% \pm 0.2011$
$\mathcal{T}A\mathbf{3}_{Letter\,2}$	100%	$85\% \pm 0.1742$
$\mathcal{T}A\mathbf{3}_{Letter\,3}$	100%	$95\% \pm 0.1001$
$\mathcal{T}A\mathbf{3}_{Letter\,4}$	100%	100%

For the classification task the system is presented with 16 attributes describing a letter and the task is to classify the letter, based on previously seen cases. Performance evaluation, based on simple cross-validation is summarized in Table 8. All algorithms but $\mathcal{T}A\mathbf{3}_{Letter}$ were trained on the first 16,000 and 1,600 cases respectively and then tested on the remaining 4,000 (and 400) cases. Reported results are the average values over 10 trials. Since $\mathcal{T}A\mathbf{3}_{Letter}$ does not need a learning period, we used the leave-one-out testing method on the first 2,000 and on all 20,000 cases respectively. Results are the average values over 20 random trials.

In $\mathcal{T}A\mathbf{3}_{Letter\,1}$ we used a modified nearest-neighbor approach; attributes were grouped into categories for selective reduction/expansion. $\mathcal{T}A\mathbf{3}_{Letter\,2}$ was obtained by using a value relaxation (i.e., relaxing attribute values to include values of immediate neighbors). If more cases were needed, reduction was performed. $\mathcal{T}A\mathbf{3}_{Letter\,3}$ was obtained by using a combination of $\mathcal{T}A\mathbf{3}_{Letter\,1}$ and $\mathcal{T}A\mathbf{3}_{Letter\,2}$, with equal voting for each of them. $\mathcal{T}A\mathbf{3}_{Letter\,4}$ was obtained by using different category groupings for attributes. In the preious test, categories were created sequentially, each group having four attributes (letter attribute was an extra category). In the latter test we used a knowledge mining tool to find out the most relevant attributes in recognizing individual letters and we grouped attributes accordingly.

4.5 *Summary of case studies*

Besides the presented domains, we have also applied the $\mathcal{T}A$**3** system to support software reuse [22] and iterative browsing and intelligent retrieval [21]. Each domain has different characteristics and thus allows for a more comprehensive performance evaluation. In the presented studies, we have measured not only task-performance (accuracy) but time-performance (scalability) as well. Especially in the robotic domain, the results must not only be accurate but they must be delivered in real time. Time constraints in the manufacturing process demand fast and accurate results.

There are several factors affecting performance: the size of a case base (measured in terms of the number of cases in the case base), the size of a case (measured in terms of an average number of attributes used to describe cases) and context (measured in terms of the number of attributes defined and constraints specified), the query complexity (measured in terms of the size of a query and the complexity of the operations required) and the relaxation/restriction strategy used. We considered average queries and varied the case base size and the context-modification strategies. The computer time (in ms) required to perform three consecutive context modifications was measured, considering the same query for different case base sizes.[8]

Comparing naive (standard retrieval algorithm) and incremental algorithms shows that k consecutive context transformations require k applications of the naive algorithm [27, 19]. Thus, a naive approach requires $k\times \mid CB \mid \times \mid \Omega \mid$ evaluations ($\mid CB \mid$ denotes a case base size and $\mid \Omega \mid$ denotes the size of a context). For a given context, all matching cases are included in SC, which is obtained by evaluating SC_{part} – a partial result of the matching, composed of the partial results of the existing cases. In contrast to this, an incremental algorithm is evaluated only once to produce SC_{part} and subsequent context transformations are handled by incremental changes to the result. Thus, assuming that the size of the initial retrieval result $\mid SC_{part} \mid$ is substantially smaller than the case base size $\mid CB \mid$, producing SC' (modified set of retrieved cases) incrementally is significantly more efficient. For all categories that are not affected by a context transformation, no recomputation is necessary. For categories affected by a change, local changes handle the recomputation as presentedin [27]. It should be noted that both naive and incremental algorithms can be improved by indexing to avoid accessing all cases in a case base; however, this is not a simple issue and is beyond the scope of this paper.

Figure 3 presents retrieval cost in a logarithmic scale. We compare both standard (S) and incremental (I) approaches during generalization (G), reduction (R), specialization (Sp) and expansion (E). Average queries are considered and the case base size is varied. The computer time required to perform three consecutive context transformations was measured, considering the same query for different case base sizes. These results are compared to the results when robotic, servo, medical and character recognition domains were used, using logarithmic scale. In [19] we discuss performance improvements using incremental algorithm, considering the cost of retrieval for context relaxation and restriction as a function of case base size and context.

[8] $\mathcal{T}A$**3** is implemented in *C++* on a SUN workstation. Presented experiments were obtained on the SparcStation IPC with 32MB of memory.

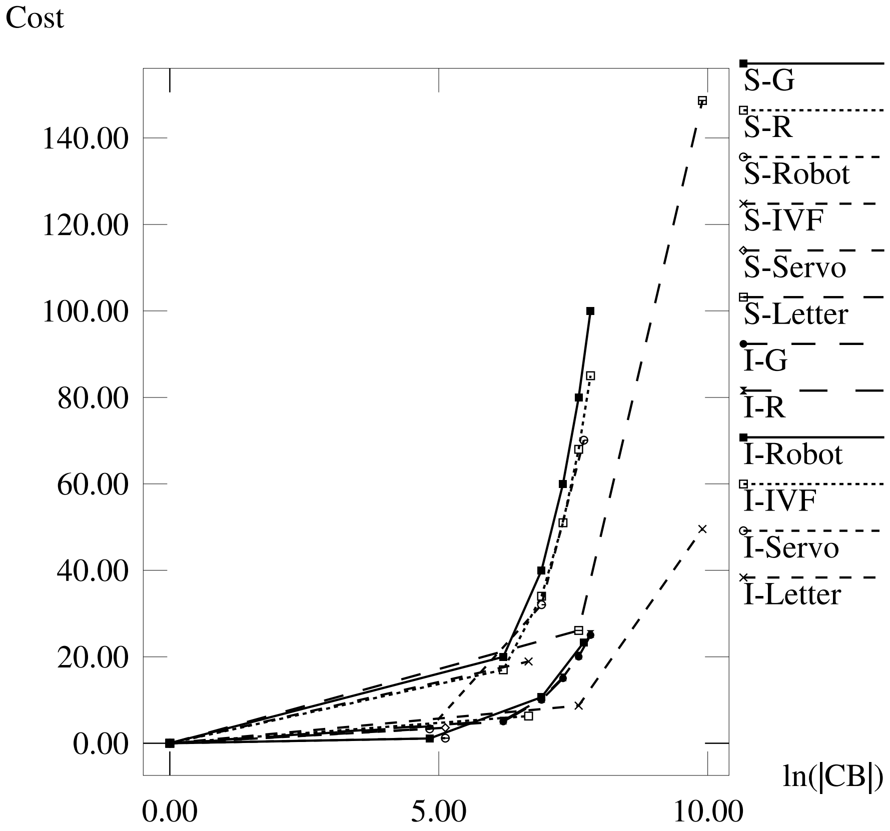

Figure 3: Cost of retrieval for three context transformations in logarithmic scale for robotic, servo, medical and character recognition domains. | CB | – case base size, Cost – CPU time. S/I – standard/incremental context transformation. G – generalization, R – reduction.

Presented results support the claim that incremental context transformations are more efficient than the naive approach. Performance improvement is increased when the case base size is substantial, cases have many attributes and several subsequent context transformations are required. However, for simple retrievals (case base and/or no consecutive context transformations) the naive approach would be more efficient, due to smaller memory requirements.

5 Conclusions

Similarity-based retrieval tools can advantageously be used in building flexible retrieval and classification systems. Case-based classification uses previously classified instances to classify unknown instances. Classification accuracy is affected by the retrieval process – the more relevant the instances used for classification, the higher the accuracy.

Using different context and criteria affects the time spent on searching for the solution as well as the quality of the solution. $\mathcal{T}A$**3** allows for tuning the reasoning process to meet different requirements, as suggested by Cunningham et al. [11]. On the one hand, in the medical domain [29, 28], a more accurate suggestion for the hormonal therapy has a positive impact on pregnancy and is also cost effective, since it minimizes the quantity of hormones given to the patient. Thus, even though the treatment should be suggested reasonably fast, clearly accuracy is more important. Using the terminology of normative rationality introduced in [16], medical application of the $\mathcal{T}A$**3** system follows type I rationality, i.e., inference that is consistent with the axioms of decision theory regardless of the cost of inference. On the other hand, robotic application of $\mathcal{T}A$**3** implements type II rationality, since the cost of reasoning is considered and time resources are limited. Thus, even an approximate solution provided in real time is more valuable than an accurate solution delivered late. In the inverse kinematic task, there are available fast techniques that produce an accurate solution from an approximate one. However, for some robotic architectures, there might not be a computational solution to the problem. Thus, the main objective is to have a solution available within given time constraints.

Flexible systems require strategies that continuously refine the value of partial results with time. $\mathcal{T}A$**3** supports this by context relaxation/restriction. Namely, based on obtained results (even partial) the system can change m in m_of_n matching to increase/reduce the number of retrieved relevant cases. By controlling this process, $\mathcal{T}A$**3** allows for improving retrieval precision while preserving perfect recall. This has a positive effect on a classification accuracy.

The flexibility of the $\mathcal{T}A$**3** system is further enhanced by incorporating a knowledge discovery algorithm that allows for finding a set of attributes and their values for representing cases optimally from two points of view: precision of results and efficiency of computation. Thus, the importance weight for attributes may dynamically change, which i turn changes the retrieval strategy and classification accuracy.

$\mathcal{T}A$**3** is different from other case-based approaches since instead of using predefined retrieval strategies, case retrieval is custom-tailored and dynamically changed for a particular domain and specific application. $\mathcal{T}A$**3** can be configured for individual tasks easily and allows for flexibility in certain aspects of computation, such as trading off the accuracy or precision of the computation process for time resources.

We introduced the $\mathcal{T}A$**3** system, which uses context-based similarity to retrieve relevant cases and then uses them for the classification task. The validity of the proposed approach is tested on real-world domains and the performance of the proposed system is compared to the performance of other machine learning algorithms. The main advantage of the context-based approach is that classification does not degrade with irrelevant attributes, since they can be ignored during the retrieval process and thus do not affect the solution.

References

[1] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[2] D. W. Aha. Generalizing from case studies: A case study. In *The 9th International Conference on Machine Learning*, pages 1–10, Aberdeen, 1992.

[3] D. W. Aha. An implementation and experiment with the nested generalized exemplars algorithm. Technical Report AIC-95-003, Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington, DC, 1995.

[4] D. W. Aha and R. L. Bankert. Feature selection for case-based classification of cloud types: An empirical comparison. In *AAAI-94 Workshop on Case-Based Reasoning*, Seattle, WA, 1994.

[5] D. W. Aha and S. L. Salzberg. Learning to catch: Applying nearest neighbor algorithms to dynamic control tasks. In P. Cheeseman and R. W. Oldford, editors, *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springer-Verlag, 1994.

[6] E. Bareiss, B. W. Porter, and C. Wier. The exemplar-based learning apprentice. Technical Report AI87-53, The Univ. of Texas at Austin, 1988.

[7] L. K. Branting. Integrating generalizations with exemplar-based reasoning. In *Proc. of the 11th Annual Conference of the Cognitive Science Society*, pages 139–146, Ann Arbor, MI, 1989.

[8] T. Cain, M. J. Pazzani, and G. Silverstein. Using domain knowledge to influence similarity judgments. In *Proceedings of the Case-Based Reasoning Workshop*, pages 191–198, Washington, DC, 1991.

[9] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A Bayesian classification system. In *Proc. of the 5 th International Conference on Machine Learning*, pages 54–64, Ann Arbor, MI, 1988.

[10] W. W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson. CoBase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 6, 1996.

[11] P. Cunningham, A. Bonzano, and B. Smyth. An incremental case retrieval mechanism for diagnosis. Technical Report TCD-CS-95-01, Trinity College, Dublin, Ireland, 1995.

[12] J. Frawley and G. Piatetsky-Shapiro. *Knowledge Discovery in Databases*. AAAI Press, 1991.

[13] P. W. Frey and D. J. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2), 1991.

[14] T. Gaasterland. Restricting query relaxation through user constraints. In *Proc. on International Conference on Intelligent and Cooperative Information Systems*, pages 359–366, Rotterdam, 1993.

[15] T. Gaasterland, P. Godfrey, and J. Minker. Relaxation as a platform of cooperative answering. In *Proc. International Workshop on Nonstandard Answers and Queries*, Toulouse, France, 1991.

[16] I. Good. Relational decisions. *J. R. Stat. Soc. B*, 14:107–114, 1952.

[17] R. Greiner. *AAAI Fall Symposium Series on Relevance*. AAAI Press, Menlo Park, CA, 1994.

[18] A. D. Griffiths and D. G. Bridge. On concept space and hypothesis space in case-based learning algorithms. In *Proc. of 8th European Conference on Machine Learning*, 1995.

[19] I. Jurisica, , and J. Glasgow. An efficient approach to iterative browsing and retrieval for case-based reasoning. 1997. Submitted.

[20] I. Jurisica. How to retrieve relevant information? In R. Greiner, editor, *AAAI Fall Symposium Series on Relevance*, pages 101–104, Menlo Park, CA, 1994.

[21] I. Jurisica. $\mathcal{T}A\mathbf{3}$: Case-based intelligent retrieval and advisory tool. In *ACM Conference on Society and the Future of Computing*, Durango, CO, 1995.

[22] I. Jurisica. A similarity-based retrieval tool for software repositories. In *The Third Workshop on AI and Software Engineering: Breaking the Mold. IJCAI-95*, Montreal, Quebec, 1995.

[23] I. Jurisica. Inductive learning and case-based reasoning. In *Canadian AI Conference, Workshop on What is inductive learning?*, Toronto, 1996.

[24] I. Jurisica. Supporting flexibility. A case-based reasoning approach. In *The AAAI Fall Symposium. Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities*, Cambridge, 1996.

[25] I. Jurisica and J. Glasgow. Applying case-based reasoning to control in robotics. In 3^{rd} *Robotics and Knowledge-Based Systems Workshop*, St. Hubert, Quebec, 1995.

[26] I. Jurisica and J. Glasgow. Case-based classification using similarity-based retrieval. In 8^{th} *IEEE International Conference on Tools with Artificial Intelligence*, Toulouse, France, 1996.

[27] I. Jurisica and J. Glasgow. A case-based reasoning approach to learning control. In *5*th *International Conference on Data and Knowledge Systems for Manufacturing and Engineering, DKSME-96*, Phoenix, Arizona, 1996.

[28] I. Jurisica, J. Mylopoulos, J. Glasgow, H. Shapiro, and R. F. Casper. Case-based reasoning in IVF: Prediction and knowledge mining. *AI in Medicine*, 1997. In Press.

[29] I. Jurisica and H. Shapiro. A computer model for case-based reasoning in IVF. In *The 51*st *Conference of the American Society for Reproductive Medicine*, Seattle, Washington, 1995.

[30] D. Kibler and D. Aha. Learning representative exemplars of concepts: An initial case study. In *Proc. of the 4*th *International Workshop on Machine Learning*, pages 24–30, Irvine, CA, 1987.

[31] T. Kokeny. Constraint satisfaction problems with order-sorted domains. *International Journal on Artificial Intelligence Tools*, 4(1 & 2):55–72, 1995.

[32] I. Kononenko and I. Bratko. Information-based evaluation criteria for classifier's performance. *Machine Learning*, 6(1):67–80, 1991.

[33] D. Lauzon and T. Rose. Task-oriented and similarity-based retrieval. In *Conference on Knowledge-Based Software Engineering*, 1994.

[34] J. Mylopoulos, A. Borgida, M. Jarke, and M. Koubarakis. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362, 1990.

[35] J. Ortega. On the informativeness of the DNA promoter sequences domain theory. *Journal of Artificial Intelligence Research*, 2:361–367, 1995.

[36] B. W. Porter, E. Bareiss, and R.C.Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263, 1990.

[37] J. R. Quinlan. The effect of noise in concept learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 149–166, Los Altos, CA, 1986. Morgan Kaufmann.

[38] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[39] J. R. Quinlan. Learning with continuous classes. In *Proc. 5th Australian Joint Conference on AI*, pages 343–348, 1992.

[40] J. R. Quinlan. Combining instance-based and model-based learning. In *Proc. of the 10* th *International Conference on Machine Learning*, Amherst, MA, 1993.

[41] E. L. Rissland, J. J. Daniels, Z. B. Rubinstein, and D. B. Skalak. Case-based diagnostic analysis in a blackboard architecture. In *Proc. of AAAI-93*, Washington, DC, 1993.

[42] D. Schuurmans and R. Greiner. Learning to classify incomplete examples. In *Computational Learning Theory and Natural Learning Systems: Addressing Real World Tasks*. MIT Press, 1995.

[43] J. Shavlik, R. Mooney, and G. Towell. Symbolic and neural learning algorithms - An experimental comparison. *Machine Learning*, 6(2):111–143, 1991.

[44] R. E. Stepp and R. S. Michalski. Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 471–498. Morgan Kaufmann, 1986.

[45] P. R. Thagard, K. J. Holyoak, G. Nelson, and D. Gotchfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310, 1990.

[46] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.

[47] D. Wettschereck and T. Dietterich. An experimental comparison of the nearest neighbor and nearest hyperrectangle algorithmns. *Machine Learning*, 19(1):5–27, 1995.

Acknowledgments

This research was supported by the Information Technology Research Center of Ontario and IBM Toronto Lab, Centre for Advanced Studies.

A Theorem Proofs

Theorem 3.1

If Ω_1 is a relaxation of Ω_2 then all cases that satisfy Ω_2 also satisfy Ω_1:

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_2) \wedge \Omega_1 \succ \Omega_2) \rightarrow sat(\mathcal{C}, \Omega_1).$$

Proof of Theorem 3.1:

Assume for a given case $\mathcal{C}$, $sat(\mathcal{C}, \Omega_2)$ and $\Omega_1 \succ \Omega_2$.

From Definition 3.3 and $sat(\mathcal{C}, \Omega_2)$:

$$\forall a_j \langle a_j : CV_j \rangle \in \Omega_2 \ \rightarrow \ \exists V_j \langle a_j : V_j \rangle \in \mathcal{C} \ \wedge \ V_j \in CV_j. \tag{1}$$

From Definition 3.7 and $\Omega_1 \succ \Omega_2$:

$$\forall \langle a_i : CV_i \rangle \in \Omega_1, \ \exists \langle a_i : CV_j \rangle \in \Omega_2 \ : \ CV_i \supseteq CV_j \ \wedge \ \Omega_1 \neq \Omega_2. \tag{2}$$

Assume for some a_i: $\langle a_i : CV_j \rangle \in \Omega_1$.

From (1) above: $\exists V_j \langle a_i : V_j \rangle \in \mathcal{C} \ : \ V_j \in CV_j.$ (3)

From (2) and (3) above: $V_j \in CV_i.$ (4)

From (3) and (4) above:

$$\exists V_j \langle a_i : V_j \rangle \in \mathcal{C} \ \wedge \ V_j \in CV_j. \tag{5}$$

Therefore:

$$\forall a_i \langle a_i : CV_j \rangle \in \Omega_1 \ \rightarrow \ \exists V_j \langle a_i : V_j \rangle \in \mathcal{C} \ \wedge \ V_j \in CV_j. \tag{6}$$

Thus by Definition 3.3: $sat(\mathcal{C}, \Omega_1)$.

□

Theorem 3.2

The retrieval function based on context-based similarity assessment is monotonic. More precisely, given a case base Δ and contexts Ω_1, Ω_2:

$$(\Omega_1 \succ \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \supseteq retrieve(\Omega_2, \Delta).$$

$$(\Omega_1 \prec \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \subseteq retrieve(\Omega_2, \Delta).$$

Proof of Theorem 3.2:

Assume $\Omega_1 \succ \Omega_2$.

For any given $\mathcal{C}$, if $\mathcal{C} \in retrieve(\Omega_2, \Delta)$, $sat(\mathcal{C}, \Omega_2)$ by Definition 3.5. (1)

From Theorem 3.1, $\Omega_1 \succ \Omega_2$ and (1): $sat(\mathcal{C}, \Omega_1)$. (2)

From Definition 3.5 and (2) above: $\mathcal{C} \in retrieve(\Omega_1, \Delta)$. (3)

Therefore from (3): $(\Omega_1 \succ \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \supseteq retrieve(\Omega_2, \Delta)$.

Directly from above: $\Omega_1 \succ \Omega_2 \rightarrow \Omega_2 \prec \Omega_1$.
Therefore $(\Omega_1 \prec \Omega_2) \rightarrow retrieve(\Omega_1, \Delta) \subseteq retrieve(\Omega_2, \Delta)$.
$\square$

Corollary 3.1

If Ω_1 is a restriction of Ω_2 then the set of cases that satisfy Ω_1 is a subset of the set of cases that satisfy Ω_2.

$$\forall \mathcal{C} \in \Delta : (sat(\mathcal{C}, \Omega_1) \wedge \Omega_1 \prec \Omega_2 \ \rightarrow \ sat(\mathcal{C}, \Omega_2)).$$

Proof of Corollary 3.1:
Trivial from Definition 3.8 and Theorem 3.1.
$\square$