On the Use of Vocabulary Knowledge for Learning Similarity Measures

Thomas Gabel

University of Osnabrück
Neuroinformatics Group
49069 Osnabrück, Germany
thomas.gabel@uos.de

Abstract. A very recent topic in CBR research deals with the automated optimisation of similarity measures—a core component of each CBR application—by using machine learning techniques. In our previous work, a number of approaches to bias and guide the learning process have been proposed aiming at more stable learning results and less susceptibility to overfitting. Those methods support the learner by incorporating background knowledge into the optimisation process. In this paper, we focus on one specific form of knowledge, namely vocabulary knowledge implicitly contained in the model of the respective application domain, as a source to enhance the learning of similarity measures.

1 Introduction

Similarity is a central concept in Case-Based Reasoning. Each case-based application is in need of adequate similarity measures in order to retrieve those cases that are most useful for a given query. The history of CBR has seen various approaches towards modelling similarity measures. On the one hand, simple distance metrics, that base their similarity assessment on a syntactic match between case and query, have been used frequently. Though being definable in a rather straightforward manner, they lead to poor retrieval results in many cases. The alternative is represented by the manual definition of similarity measures, that are highly sophisticated and complex in structure, by knowledge engineers who take the specifics of the respective application domain into consideration. Both alternatives feature certain inherent advantages and drawbacks [9].

Though there have been a number of approaches to learn some elements of a similarity measure (e.g., feature weights [12]), the completely automated construction of an entire similarity measure depicts a rather new idea. Stahl [10] presented a comprehensive framework that is capable of acquiring knowledge-intensive similarity measures from utility feedback and applicable not only to conventional tasks, such as classification, but also in typical CBR domains (e.g. in e-commerce).

In our recent work we have introduced a way to utilise evolutionary algorithms for the learning of similarity measures in the context of the framework mentioned before [11]. Although the application of these machine learning algorithms featured convincing results, under certain circumstances some weaknesses were revealed. For example, when learning on the basis of a small amount of training data, the algorithms tended to overfit. Tackling this and other problems, we developed a methodology to enhance the optimisation process by incorporating background knowledge [5]. The paper at hand directly builds upon that work and presents further possibilities to stabilise the learning of similarity measures.

The remainder of this article is structured as follows: Section 2 reviews in short the mentioned strategies on exploiting background knowledge to improve the learning process. For more details on that topic the reader is referred to [5, 4]. In Section 3 we introduce new concepts on the utilisation of vocabulary knowledge, which may be implicitly contained in the domain model of the CBR system's respective application, as a promising source to support the learning process. Section 4 presents some experimental results gained when applying the concepts explained in the preceding section, and Section 5 concludes.

2 Exploiting Background Knowledge When Learning Similarity Measures

When learning similarity measures, there is a high risk of overfitting the learning results to the training data given. This is due to the enormous search space being taken into consideration during learning. In particular, our representation of local similarity measures (similarity tables for symbolic data types with a small amount of of values and difference-based similarity functions for numeric types, see [11]) allows for learning very complex and specific measures. However, the learnt measures may eventually show poor performance when used for some independent test data set.

Consequently, a self-evident idea is to exploit easily available background knowledge in order to restrict the hypothesis space considered by the evolutionary learning algorithm and thus to bias the optimisation process.

Another crucial issue regarding similarity measures in CBR concerns the modelling effort. Completely manual definition and fully automated learning of knowledge-intensive similarity measures represent two extremes of modelling techniques which both feature certain advantages and drawbacks [9]. When considering a knowledge engineer's (partial) definition of a similarity measure as background knowledge and feeding that into the learning algorithm, a hybrid modelling approach is possible and allows for gaining benefits from both extremes.

2.1 Motivation

The knowledge exploited to enhance the learning can be divided into two groups. Here, we give only a short overview on those knowledge sources, a more detailed description can be found in [5].

Similarity Meta Knowledge determines general demands on the appearance of learnt similarity measures.

On the one hand, it is possible to define heuristic constraints on the "typical" syntactical shape of local similarity measures. These heuristics refer to several basic properties of similarity measures and can be implemented as weak or strong constraints restricting the search space. A descriptive example concerns the reflexivity of similarity measures: In most application domains a non-reflexive similarity measure would be unpropitious.

On the other hand, the case knowledge container holds a certain amount of unexploited knowledge potential, too. Assumed that the case base contains a sufficient number of cases, the distribution of those cases throughout the entire space of possible cases and especially the distribution of their attribute values, reveals interesting opportunities to improve the learning process. Here, it is possible to employ a statistical case base analysis ("mining" knowledge from the case base) to find out which regions of the space of similarity measures are really worth to be searched thoroughly.

Expert Knowledge The aid of a knowledge engineer and the incorporation of his/her expert knowledge into the learning process may be very valuable.

Defining similarity measure bottom-up is a complex, time-consuming and probably error-prone task that is reliant to a human domain expert. Searching the whole space of possible similarity measures with the help of a learning algorithm can be time-consuming and is often susceptible to overfitting, if the amount of available training examples is rather small. As already indicated above, here the idea is to meet in the middle, i.e. to incorporate the expert's partial knowledge into the learning process.

Furthermore, when building a CBR system, the expert settles the domain vocabulary, which in turn may include some implicit similarity knowledge to be exploited as well. This kind of expert knowledge is represented as vocabulary knowledge—which corresponds to one of the knowledge containers as introduced [8]. More precisely, the vocabulary provides possibilities to constrain the search space by utilising the information that is contained within *structured data types*, such as symbolic types with an ordered or taxonomic value range. This paper in particular focuses on that vocabulary knowledge source.

2.2 Knowledge-Based Optimisation Filters

To realise the actual restriction of the search space we have introduced the concept of knowledge-based optimisation filters (kbOF) [4]. With that name we denote objects that, on the one hand, store some gathered knowledge concerning the learning of similarity measures. On the other hand, they actively interfere with the learning process, in order to bias and direct the search for optimal similarity measures. For the implementation of the evolutionary algorithm this means that a kbOF exerts its influence during offspring generation: Newly generated individuals contradicting too much to the filter's knowledge are discarded. Furthermore, the kbOF is allowed to explicitly give advice to genetic operators adapting their behaviour in such a way that more realistic similarity measures are created.

During kbOF-based learning the evolutionary population for each local similarity measure to be learnt is equipped with a knowledge filter. Together with an additional filter for the feature weights, $n + 1$ filters are employed for a case representation consisting of n attributes.

3 Utilisation of Vocabulary Knowledge

Before having the first thought about the definition of similarity measures, a knowledge engineer has usually already invested a lot of modelling effort: Prior to settling similarity measures, the respective application domain—in terms of attributes and corresponding data types[1]—must be modelled. This involves tasks such as determining which case attributes are to be used, which are indiscriminant and which are possibly dependent on each other. Moreover, each attribute must be associated with an appropriate data type and domain, respectively. In other words, the expert is fully responsible for the definition of the CBR system's vocabulary knowledge. In this section we want to show how a little of this knowledge implicitly encoded into the domain vocabulary can be exploited to support the learning of similarity measures. We focus on two specific forms of symbolic data types whose domains can be organised in a structured way: taxonomic symbolic and ordered symbolic data types.

So far, we represented any local similarity measure for a symbolic data type as a similarity table and conducted the optimisation on corresponding matrices (consisting of ϕ^2 entries for an attribute A using a data type with $|D_A| = \phi$ elements in its domain). However, local similarity measures for attributes based on taxonomic and ordered symbolic data types are of course definable with respect to the characteristics of those types: This property will be exploited by the concepts we introduce in the following.

3.1 Taxonomic Symbolic Attributes

The elements of a symbolic data type for a specific attribute may be structurable in a taxonomy. Of course, that taxonomic ordering effects the belonging local similarity measure and the way the similarity between a query and a case is computed. In Figure 1 we show the example of a taxonomic data type used for a *MaritalStatus* attribute.

Definition 1 (Taxonomic Symbolic Attribute).
Let A be a symbolic attribute with value range $D_A = \{d_1, \dots, d_n\}$. A is called a ***taxonomic attribute****, if there exists an accentuated root element $r \in D_A$ for the taxonomy. Furthermore, the remaining elements $d \in D_A \setminus r$ must be arranged in a tree structure $\mathcal{T}_A$ with r as root of the tree. The predicate $isChild(d_i, d_j)$ is true, if d_i is a (possibly indirect) successor of d_j within $\mathcal{T}_A$.*

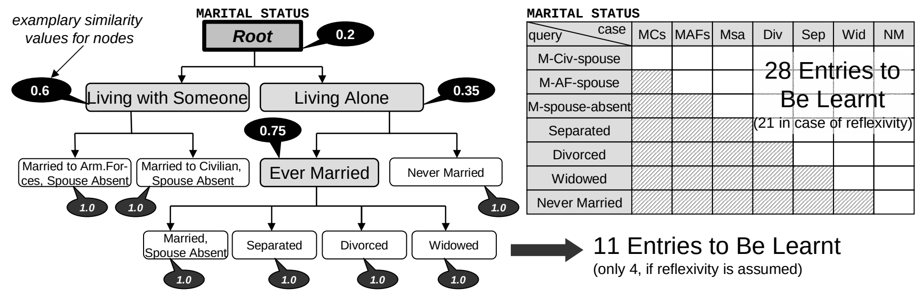

Fig. 1. Representation of Taxonomic Symbolic Similarity Measures as Individuals (left) and Comparison to a Representation as Similarity Table Individual (right)

In [1] the author describes very detailed how taxonomies can be used for representing case features and also addresses the resulting implications for appropriate local similarity measures. There, the focus is laid upon possible semantics for leaves and inner nodes as well as on the respective similarity assessments. Here, we employ a simplified approach to calculate the similarity for taxonomic attributes.

Definition 2 (Taxonomic Similarity Measure).
Let A be a taxonomic attribute with its taxonomic value range $D_A = \{d_1, \ldots, d_n\}$ arranged in $\mathcal{T}_A$. Moreover, let each node d of $\mathcal{T}_A$ be annotated with a similarity value $s_d \in [0;1]$ so that it holds:

$$s_i \leq s_j \quad \textit{iff} \quad dist(i,r) \leq dist(j,r) \tag{1}$$

where dist measures the distance (number of edges) between two tree nodes and r denotes $\mathcal{T}_A$'s root element.

Then, the ***taxonomic local similarity measure*** *for A is defined as*

$$\begin{array}{rl} sim_A : D_A \times D_A & \rightarrow [0;1] \\ (q,c) & \mapsto s_{NCP(q,c)} \end{array}$$

where $NCP(q,c)$ determines the nearest common parent node for q and c.

Sometimes, it can be advantageous to allow the leaves of $\mathcal{T}_A$ to be elements of D_A only, i.e. it holds $D_A \subsetneq nodes(\mathcal{T}_A)$. Then, the notation introduced above is still appropriate, if we consider D_A to be extended by $\mathcal{T}_A$'s inner nodes.

Representing Taxonomic Individuals

When intending to learn similarity measures for taxonomic attributes, we need to determine an appropriate representation of such measures as individuals. In [11] we made use of vectors of sampling points to represent difference-based similarity functions and of similarity matrices to represent symbolic attributes' similarity tables. Here, we propose to employ trees as individuals that correspond to the taxonomically ordered value range of the respective attribute.

[1] We assume the commonly used attribute-value based representation.

Definition 3 (Similarity Tree Individual).
*An individual I representing a taxonomic local similarity measure sim_A for the taxonomic symbolic attribute A is coded as a tree T_A^I whose structure is identical to $\mathcal{T}_A$. The nodes of that **similarity tree** T_A^I are real numbers whose values are determined by the node annotations s_i ($i \in \{1, \ldots, n\}$) that are made to the nodes in $\mathcal{T}_A$.*

When realising similarity tree individuals according to Definition 3, essentially the condition given by Equation 1 must be fulfilled. This means, when creating a new similarity tree individual care must be taken that the similarity values in a similarity tree are increasing from the root towards the leaves of the tree.

Genetic Operators

The application of mutation operators (for their accurate definition see [11]) is straightforward, when using the similarity values of a tree node's predecessor node (i.e., s_p) and the maximum of its m child nodes ($max\{s_{c_1}, \cdots, s_{c_m}\}$) as lower and upper bound, respectively, for the determination of the adapted (mutated) value. Arithmetic crossover's [7] application is trouble-free as well, since an averaged similarity tree individual created from two parent individuals, that fulfill all relations, is consistent with those relations, too.

Crossover variants that make use of splitting the parental genome and composing the offspring by genome parts from its parents (e.g. simple crossover) are also applicable to similarity tree individuals. On the one hand, the similarity tree may be mapped to a flat vector of similarity values, so that the operators introduced for similarity function and similarity table individuals [11] can be used without change. However, in addition to the existing crossover operators we also suggest and make use of another specialised genetic operator that is designed for taxonomic individuals specifically:

- *Sub-tree crossover:* Given a taxonomic attribute A whose value range is arranged in a tree $\mathcal{T}_A$ and two parent similarity tree individuals $T_A^{I_1}$ and $T_A^{I_2}$, sub-tree crossover randomly picks a node d from $\mathcal{T}_A$. Then, it creates the offspring $T_A^{I_{new}}$ according to
$$t_i^{I_{new}} = \begin{cases} t_i^{I_1} & \text{if } d \triangleright i \\ t_i^{I_2} & \text{else} \end{cases} \qquad \text{for all } i \in \mathcal{T}_A$$
where the operator $\triangleright$ denotes that i is a node within the sub-tree of $\mathcal{T}_A$ having root d.

Search Space Restriction

Analysing the restriction of the search space that can be achieved, when utilising information about an attribute's taxonomic value range, we ought to consider the worst case scenario in which the minimal restriction is reached.

First of all, we need to stress that our current realisation does not yet allow to use inner tree nodes as values for queries or cases, i.e. the values from D_A are to be found in the leaves exclusively. This entails the following implications:

– Differentiated semantics for inner nodes, as depicted by [1] cannot be employed. As asymmetries in the respective similarity measure result from different interpretations of inner nodes when used as query or case, only symmetric taxonomic similarity measures can be learnt up to now.
– Since we allow values from D_A ($|D_A| = \phi$) in the leaves of a similarity tree individual only, additional nodes (including a root element) are necessary to form the tree. Thus, in the worst case (binary tree) $\phi - 1$ additional nodes are needed, i.e. a similarity tree individual maximally consists of a total of $2\phi - 1$ similarity values.

Accordingly, for a similarity tree individual I for attribute A with $D_A = \{d_1, \ldots, d_\phi\}$ maximally $2\phi - 1$ independent similarity values have to be optimised. If we had treated A as a (non-taxonomic) symbolic attribute and used similarity matrices instead (assuming symmetric measures only), there would have been $\frac{\phi^2+\phi}{2}$ free parameters to be tuned[2]. Hence, the utilisation of vocabulary, here taxonomic, knowledge in fact results in a search space restriction as it holds $2\phi - 1 \leq \frac{\phi^2+\phi}{2}$ for all $\phi \in \mathbb{N}$. Note, that the actual gain is in fact higher than maybe presumed from that inequality: Due to the taxonomic structuring of D_A and due to the relations associated with all nodes (cf. Equation 1) within the tree, most of the $2\phi - 1$ similarity values cannot be chosen from $[0; 1]$ but from a smaller subinterval of $[0; 1]$ only.

3.2 Ordered Symbolic Attributes

A simpler ordering of a symbolic attribute's allowed values $d_1, \ldots, d_n$ (simpler in comparison to a taxonomic ordering) is given, when the elements d_i are ordered totally.

Definition 4 (Ordered Symbolic Attribute).
Let A be a symbolic attribute with value range $D_A = \{d_1, \ldots, d_n\}$. A is called an ***ordered symbolic attribute****, if each d_i is associated with a numeric scaling value $o_i \in \mathbb{R}$, so that it holds $o_i < o_j$ for all $i < j$.*

The mentioned associations o_i may then be employed as representative for the actual elements d_i in order to realise an ordered symbolic similarity measure on the basis of a difference-based similarity function.

Definition 5 (Ordered Symbolic Similarity Measure).
Let A be an ordered symbolic attribute with its ordered value range $D_A = \{d_1, \ldots, d_n\}$ and belonging numeric associations $o_1, \ldots, o_n$. Then, the ***ordered symbolic local similarity measure*** *is defined as*

$$\begin{aligned} sim_A : D_A \times D_A &\rightarrow [0; 1] \\ (q, c) &\mapsto sim_{A_0(q,c)} \end{aligned}$$

where sim_{A_0} is a difference-based similarity function that is defined on $I_d = [o_1 - o_n, o_n - o_1]$ and that assigns $sim_{A_0}(o_i, o_j)$ on the basis of the difference $o_i - o_j$.

[2] The denominator is 2 as we only consider symmetric similarity measures, here.

Representing Ordered Symbolic Individuals

An ordered symbolic similarity measure sim_A shall be represented by an individual that consists of two real-valued vectors. The first one is to define the association o_i mentioned above and in so doing affects the scaling of A's value range. Thus, it basically determines the x-axis on which the second vector with its similarity values operates: Figure 2 illustrates how we represent individuals for local similarity measures with an ordered value range, giving an example for an ordered symbolic attribute *RainbowColour*.

Definition 6 (Ordered Symbolic Individuals).
*Let A be an ordered symbolic attribute with value range $D_A = \{d_1, \ldots, d_n\}$ and belonging associations $o_1, \ldots, o_n$. An **ordered symbolic individual** representing an ordered symbolic similarity measure sim_A is coded as a tuple of two vectors: $I = (O_A^I, V_A^I)$. Here, $O_A^I = (o_1^I, \ldots, o_n^I)$, called **scaling vector**, represents a normalisation of the associations $o_1, \ldots, o_n$ according to*

$$o_i^I = \frac{o_i - o_1}{o_n - o_1} \qquad \text{for all } i \in \{1, \ldots, n\}$$

so that it holds $o_i^I \leq o_j^I$ for all $i < j$ as well as $o_1^I = 0$ and $o_n^I = 1$.

*$V_A^I = (v_1^I, \ldots, v_s^I)$, called **similarity vector**[3], represents a collection of a difference-based similarity function's similarity values at sampling points distributed equally over $[o_1^I - o_n^I, o_n^I - o_1^I] = [-1; 1]$, where s denotes the number of sampling points used.*

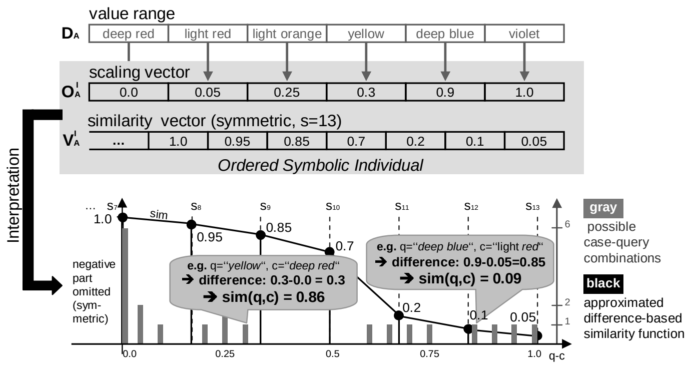
value range
D_A
deep red
light red
light orange
yellow
deep blue
violet
scaling vector
O^I_A
0.0
0.05
0.25
0.3
0.9
1.0
similarity vector (symmetric, s=13)
V^I_A
...
1.0
0.95
0.85
0.7
0.2
0.1
0.05
Ordered Symbolic Individual
Interpretation
... s7 s8 s9 s10 s11 s12 s13
sim
1.0
0.95
0.85
0.7
0.2
0.1
0.05
negative part omitted (sym-metric)
e.g. q="yellow", c="deep red" ➔ difference: 0.3-0.0 = 0.3 ➔ sim(q,c) = 0.86
e.g. q="deep blue", c="light red" ➔ difference: 0.9-0.05=0.85 ➔ sim(q,c) = 0.09
0.0
0.25
0.5
0.75
1.0
q-c
6
2
1
gray
possible case-query combinations
black
approximated difference-based similarity function

Fig. 2. Representation of Ordered Symbolic Similarity Measures as Individuals

During optimisation, when ordered symbolic individuals are used in the scope of an evolutionary algorithm, both vectors are learnt simultaneously. For the constraints related to the scaling vector O_A^I, the functionality provided by kbOFs can be utilised. Here, we mainly need to meet relational constraints ($o_i^I < o_j^I$ for all $i < j$) as well as predefined values ($o_1^I = 0$ and $o_n^I = 1$).

[3] Note, that a similarity vector V_A^I corresponds to a similarity function individual, as defined in [11].

Genetic Operators

Regarding genetic operators the situation for ordered symbolic individuals (in particular for its first component, the scaling vector) is not much different from the situation for taxonomic ones (see Section 3.1). In addition to the previous operators, we introduce a further specialised, heuristic crossover operator that is responsive to the constraints that must be fulfilled by the respective individual's scaling vector:

– *Information exchange crossover:* Given two parent scaling vectors $O_A^{I_1}, O_A^{I_2} \in [0;1]^n$, this operator mingles both vectors' elements and sets
$$z = (z_1, \ldots, z_{2n}) \text{ with } z_i \leq z_j \text{ for all } i < j$$
$$\text{and } \forall i \exists j \text{ so that } z_i = o_j^{I_1} \text{ or } z_i = o_j^{I_2}$$
The offspring is formed according to
$$O_A^{I_{new}} = (z_1, z_3, \ldots, z_{2n-1}) \text{ or } O_A^{I_{new}} = (z_2, z_4, \ldots, z_{2n}).$$

Search Space Restriction

Let us again assume an ordered symbolic attribute A with an ordered value range $D_A = \{d_1, \ldots, d_\phi\}$ (note, that $\phi = n$). Treating that attribute as a "normal" symbolic one and modelling its similarity measure with a similarity table—using, so to say, an empty knowledge filter—, there would be ϕ^2 independent entries to be adjusted. Making use of the total ordering of D_A's value range, however, the search space becomes restricted: Doing so, the modelling of ordered symbolic local similarity measures requires the scaling vector, which consists of ϕ elements, and the similarity vector, which represents a similarity function and thus consists of s sampled similarity values. Consequently, (using a kbOF that contains the above-mentioned constraints concerning the scaling vector) learning on the basis of ordered symbolic similarity measures requires the optimisation of $\phi + s$ similarity values only.

Having a look at the example from above ($\phi = n = 6$) a search space restriction occurs, when choosing $s < 30$. Since a reasonable choice for s is, for instance, $s = 13$ (as depicted in Figure 2), the achieved restriction is in this case almost 50%. Note, that the actual search space restriction is, however, even higher. The total ordering of the scaling vector's elements causes that those values cannot be chosen arbitrarily from $[0;1]$, but from a subinterval of $[0;1]$ instead. Moreover, for $o_1^I = 0$ and $o_n^I = 1$ the values are even fixed.

4 Experiments

In the following we present some experimental results gained from an evaluation of the concepts presented in the previous section. We here focus on the exploitation of taxonomic symbolic attributes only, leaving an analysis of using ordered symbolic individuals for future work.

For this purpose we have chosen a classification domain from the UCI Machine Learning Repository [3]—the *Adult* domain where the task is to predict whether a person earns more than 50k dollar a year—in which the employment of taxonomically structured data types seemed advisable. The case representation in this domain consists of 14 attributes, six of which are real-numbered, the remaining eight ones being symbolic attributes. We processed these attributes as follows:

1. First, we removed all the numeric attributes, since in the scope of this evaluation we want to analyse the influence of taxonomically structured data types only. Of course, we are aware that this removal of case attributes will significantly impair the system's overall performance in producing correct classifications. However, we need to stress that it is not our aim here to create the best CBR-based classifier possible, but to examine the usability of vocabulary background knowledge for the process of learning similarity measures.
2. The data records contained in the *Adult* domain originate from the the "CPS Current Population Study"[4]. With the help of a description of the data [6] and using our common-sense knowledge we derived taxonomies for the domains of six out of the eight symbolic attributes (the domains of two symbolic attributes turned out to be not structurable in a taxonomy, so these attributes were removed as well). An example for such a taxonomy is depicted in Figure 1 for the attribute *MaritalStatus*.
3. In a third step, we constructed a kbOF for each of the taxonomies created in the previous step. These so-called t-Filters primarily included the relations resulting from Definition 2. Apart from these constraints, those filters were not enriched with any further knowledge—this is in contrast to the experiments described in [5] where filter classes were introduced and each single filter incorporated a number of different knowledge pieces. As a consequence, we here can investigate the effects of taxonomic vocabulary knowledge solely.

The actual experiments we conducted were based, on the one hand, on filterless learning (i.e. not using any background knowledge at all and thus learning conventional similarity matrices, cf. the right part of Figure 1) and, on the other hand, on the t-Filters we created in step 3.

In Figure 3 (left) we summarise the achieved error reductions in the *Adult* domain for increasing training data sizes (x-axis). The baseline similarity measure represents, on the one hand, a knowledge-poor (default) similarity measure, into whose construction no further knowledge engineering effort has been put, i.e. the similarity assessment here is based on an uninformed syntactic match. On the other hand, we illustrate the accuracies that resulted from a similarity measure that was obtained from filterless learning. The charts' third data row sketches the increased accuracies that could be gained due to the incorporation of explicit vocabulary background knowledge through the use of a t-Filter.

Obviously, the classification accuracy on some independent test data set can be increased by up to 2%, depending on the amount of training data used (left chart). It is worth remembering that this improvement can be reached with almost no effort, simply by "switching a flag", i.e. by prompting the learning

[4] `http://www.bls.census.gov/cps`

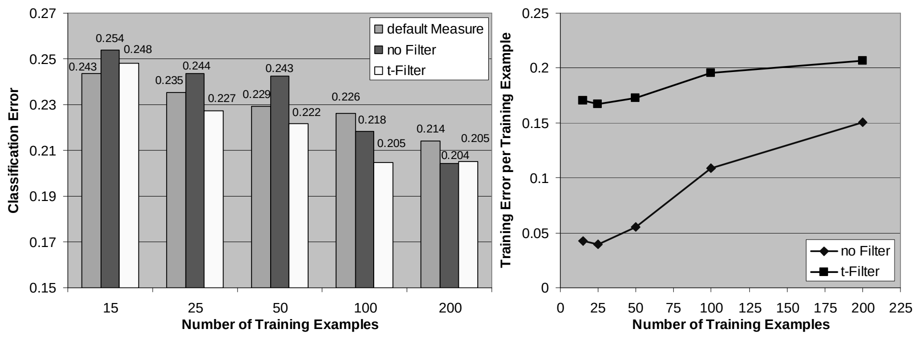

Fig. 3. Experimental Results: Gains Due to the Incorporation of Vocabulary Knowledge (left) and Hindering from Overfitting (right)

algorithm to make use of the taxonomic structures found in the case representation's attributes.

In the right chart we illustrate how the learning algorithm yields those improvements: There the values of the error on the training data (a specialised measure corresponding to the fitness of individuals within the scope of the evolutionary algorithm, see [4] for details) is shown. Apparently, the additional knowledge included in the t-Filter impedes the learner from specialising too much to the training data—especially for small amounts of training data used—, which means that its tendency to overfitting is reduced and its ability in generalising is increased.

5 Conclusions

Different forms of knowledge may be exploited when intending to bias and support the learning of knowledge-intensive similarity measures in CBR. Similarity meta knowledge comes with almost no acquisition effort [4]. Expert knowledge, on the other hand, is more expensive in terms of acquiring it, but is likely to yield higher learning improvements [5]. This paper dealt with a specific sub-form of expert knowledge: vocabulary expert knowledge. Modelling a CBR application's domain vocabulary represents a time-consuming and demanding process which, however, a knowledge engineer indispensably must perform as one of the first steps when creating a CBR system. Accordingly, after that modelling phase, the vocabulary is fixed and the knowledge it inherently contains can be exploited for supporting the settling of similarity measures without any further cost.

We developed concepts to utilise the knowledge implicitly contained in structured (taxonomic and ordered symbolic) data types during the learning process. Our evaluation proved empirically that the knowledge encoded in taxonomic data types indeed helps to bring about better learning results. Possible directions for future research involve more comprehensive evaluation studies, in particular an investigation on how the gains due to the incorporation of vocabulary knowledge described in this article can be combined with other knowledge sources and to which extent synergies can be achieved. Moreover, an extension of our framework to incorporate background knowledge into the optimisation process, so that it can handle even more complex forms of vocabulary knowledge, e.g. by utilising object-oriented case representations [2], is aspired.

Acknowledgements
The author would like to thank Gilbert Esser for his support during data collection and preparation.

References

1. R. Bergmann. On the Use of Taxonomies for Representing Case Features and Local Similarity Measures. In *Proceedings of the 6th German Workshop on Case-Based Reasoning*, 1998.
2. R. Bergmann and A. Stahl. Similarity Measures for Object-Oriented Case Representations. In *Proceedings of the 4th European Workshop on Case-Based Reasoning*. Springer, 1998.
3. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.
4. T. Gabel. Learning Similarity Measures: Strategies to Enhance the Optimisation Process. Master thesis, Kaiserslautern University of Technology, 2003. http://www.inf.uos.de/tgabel/publications/2003Gabel_EnhanceLearning-SimMeasures.ps.
5. T. Gabel and A. Stahl. Exploiting Background Knowledge When Learning Similarity Measures. In *Proceedings of the 7th European Conference on Case-Based Reasoning*. Springer, 2004.
6. J.M. Jungblut. LES — Methods and Variables of CPS 97, 2000.
7. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1996.
8. M.M. Richter. The Knowledge Contained in Similarity Measures. Invited Talk, The First International Conference on Case-Based Reasoning, Sesimbra, Portugal, 1995.
9. A. Stahl. Defining Similarity Measures: Top-Down vs. Bottom-Up. In *Proceedings of the 6th European Conference on Case-Based Reasoning*. Springer, 2002.
10. A. Stahl. *Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning*. Ph.D. thesis, Technical University of Kaiserslautern, 2003.
11. A. Stahl and T. Gabel. Using Evolution Programs to Learn Local Similarity Measures. In *Proceedings of the 5th International Conference on Case-Based Reasoning*. Springer, 2003.
12. D. Wettschereck and David W. Aha. Weighting Features. In *Proceeding of the 1st International Conference on Case-Based Reasoning*. Springer, 1995.