Generalised Weighting:

A generic combining form for similarity metrics

Alex Ferguson Derek Bridge

Department of Computer Science
University College, Cork

a.ferguson@cs.ucc.ie d.bridge@cs.ucc.ie

Introduction

In Case-Based Reasoning and in Machine Learning, an important concept is the *similarity metric* (or measure), a function which rates how similar two objects are, according to some criteria. We concentrate here on case-base queries – finding a case in a case-base which most resembles some *probe* case. Selecting what metrics are most appropriate is a key part of both designing, and using, a case-base.

It is possible to think of devising similarity metrics as a two-step process: first, selecting a number of basis metrics, which compute the similarities for each case attribute; and then picking a combinator, which forms an *overall* judgement of case similarity by combining the *individual* attribute similarities. The customary means of combining metrics is to use a weighted average. However, in earlier work [FB99a, FB00] we have presented a number of distinct alternatives to this approach – product, strict prioritisation, and generalised prioritisation – which we argue in a number of respects are often more desirable than weighting.

In this paper, we formulate a general combinator which subsumes both our previous work, and numerical weighting. Thus we are able to use both the methodology of prioritisation, where that is indeed more appropriate for the reasons we claim, and also a weighting-like effect where that is required instead. Furthermore, the new combinator allows us to extend the weighting method to non-numeric types. We will also show that our generalisation admits new ways of combining metrics, by exemplifying an instance of our new combinator distinct from prioritisation and weighting.

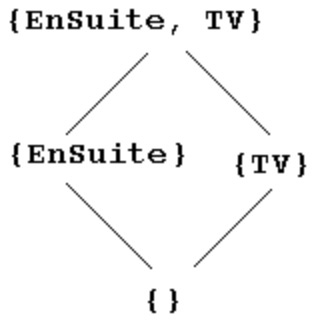
Metrics vs. measures

Traditionally, similarity measures return numeric values, which denote degrees of similarity. Our generalisation, which we term in contradistinction *similarity metrics*, is simply to allow in principle any type to denote degrees of similarity, provided only that it can be ordered in some way. To be precise, our requirement is that there exist a *partial order* on the desired result type. We discuss the detailed definitions of operators for combining metrics, and for finding the 'most' similar cases to a given probe within such a framework in earlier work [OB97, FB99b].

There are a number of reasons for wishing a more general representation than a single number; these relate both to the types of basis metrics, which as per our earlier work, and after Plaza [Pl95] may be non-numeric, and as we focus on more particularly

here, to the types obtained when metrics are combined. First though, we look at a simple example of a non-numeric basis metric.

For example, suppose we have a case attribute that is represented by a set. Let us consider a very simple example, of hotel amenities. These might include en suite bathrooms, and television in the room: let us call these `EnSuite`, and `TV`. The type of the attribute is, therefore, subsets of `{EnSuite, TV}`. One means of measuring similarity on cases in this respect, then, is simply to take the set intersection: that is, we are saying that two cases are similar *to the degree* that they share amenities. Thus not only are we representing amenities by sets, but their degrees of similarity are *also* represented by sets. This is a perfectly good result type for a metric in our framework, provided only that we can define an order on these similarity value sets. The intuitive means of doing this is to order by set inclusion: one degree of similarity is greater than another, if it includes everything the other does, and more. We illustrate this order in the figure below:



Or in the usual notation: `{}` ⊏ `{EnSuite}` ⊏ `{EnSuite}`, and `{}` ⊏ `{TV}` ⊏ `{EnSuite}`. Note that this is indeed a partial, and not total, order. Suppose we have a probe case, with amenities desired being both `EnSuite` and `TV`, and now consider comparing that to two extracted from the case base, one with `EnSuite` (only), and one with `TV` (only). Their respective degrees of similarity to the probe are, then, `{EnSuite}` and `{TV}`. Since these are not equal, and neither includes the other, then they are *incomparable* – the points are entirely unrelated to each other, according to the partial order. We therefore have a more complex structure to consider in dealing with such examples. The advantage of this is that we are not forced to make a predetermined choice to force this into a total order, which might well distort the intent of the user of a case base query system.

As we have noted, we require only that the result type have a *partial* order put on it. That is, we allow incomparable points in the set of degrees of similarity, which are neither equal, nor is one made less than the other. We require no ordering on the cases, or case attributes themselves – only on the results of whatever metrics we wish to define on them. The nature of this order has significance in terms of how the results of a case query are presented to, and interact with, the user, which we explore in earlier work [FB99b], but is motivated largely by wishing to allow the greatest amount of flexibility possible in constructing metrics appropriate to the desired effect.

Having selected basis metrics for every case attribute we are interested in, we then desire to combine them in some way. For example, we may wish to construct a

similarity value by combining the results of an amenities metric, and a price metric. This might seem problematic in our framework, where such different metric result types are allowed, e.g., the amenities metric may return sets (as above), and the price metric might be a conventional numeric-valued function. But there is an obvious choice for the result type of the combined metric: a pair. The degree of similarity of two hotel rooms might be, e.g., (`{TV}` , 0.7), where `{TV}` is the similarity on the amenities metric, and 0.7 is the similarity on the price metric. In general, given two cases c_1 and c_2, to compute their overall similarity $m\ c_1\ c_2$, we form a pair from their similarities according to two basis metrics m_1 and m_2:

$$m\ c_1\ c_2 = (m_1\ c_1\ c_2,\ \ m_2\ c_1\ c_2)$$

But of course, it is not enough to simply say that the overall similarity is a pair. The pairs must be partially ordered so that we can tell which denote the higher degrees of similarity. A number of different orders might be chosen: one choice is to use a lexicographical order based on the orders of the components – that is, ordering first by the result of m_1, and then using m_2 as a 'tiebreaker'. Other choices will be described in detail later in this paper.

Generalising weights

We now review the conventional weighted-sum combinator for similarity *measures*, and then proceed to develop an analogous treatment for handling weighting-like combinators for similarity *metrics*. Since the types returned by a metric need not be in any sense numeric, conventional weighting requires at the least some extra work. Most simply, we can add a conversion function from any give type to numerals, and then proceed with the weighted-sum calculation as usual. Where the type is a totally-ordered non-numeric, this is fairly straightforward. A less straightforward case is where the result is indeed a true partial order. Where this is so, not only is how to do the conversion less clear in itself, but the results obtained, as we will see later in this section, may be less than satisfactory.

For clarity, we consider throughout combining only two metrics; in practice we may require many more, but the combinators we discuss compose, or where necessary scale up, without difficulty. Suppose we have a probe case and we are comparing it to two cases in the case base. According to the first metric (e.g., comparing listed hotel amenities), the similarities of the two cases to the probe case are x_1 and y_1. According to the second metric (e.g. comparing prices), the similarities are x_2 and y_2. The usual definition of a weighted comparison using two weights, w_1 and w_2, is as follows. First we apply a weighted sum function, *ws,* to each pair of the similarity values (s_i):

$$ws(s_1,s_2) = w_1.s_1 + w_2.s_2$$

These are then compared in the usual arithmetic fashion, and we can view this as defining an order $<_w$ on pairs of numeric similarity values:

$$(x_1,x_2) <_w (y_1,y_2) \ \triangleq\ ws(x_1,x_2) < ws(y_1,y_2) \ = \ w_1 \,.\, x_1 + w_2 \,.\, x_2 < w_1 \,.\, y_1 + w_2 \,.\, y_2$$

In order to generalise this definition to non-numeric types, it would be necessary to first devise analogues in the appropriate type to the individual weights, then perform

an operation playing the same role as the sum, and lastly carry out a comparison in the resultant type. The 'sum' here is especially problematic: given arbitrary partial orders, what sort of operator should be used to combine two elements, to produce a third, capturing a similar sort of behaviour to the weighted sum? Since there is no clear answer to this in general, and because in practice it would often require specially devising a new sum type, we wish to avoid having to perform the comparison of pairs in such a manner, in the non-numeric case.

By a simple algebraic rearrangement of the formula above, we can also regard weighting in a slightly different manner:

$$(x_1,x_2) <_w (y_1,y_2) = w_1 \,.\, (x_1 - y_1) < w_2 \,.\, (y_2 - x_2)$$

That is, we compare each component initially separately, in this case by a subtraction, and then we combine these computations, to see which is decisive. Essentially, we are determining whether our two criteria agree, and where they do not, are then using the weights to skew the comparison to the one which is the more important. This can be seen more clearly if we define a single relative weight, as follows:

$$w = w_1/w_2$$

Then the above is equivalent to:

$$(x_1,x_2) <_w (y_1,y_2) = w \,.\, (x_1\text{-}y_1) < y_2 - x_2$$

That is, a single parameter sufficiently determines the relative importance of each component (provided $w_2 = 0$).

This simple reformulation provides the basis of our generalisation: we replace subtraction by a generalised difference function on each component, and each multiplication that uses a weight, w_i, by the application of a weighting function, ω_i. The difference functions may yield a numeric or a non-numeric type, and these types may not necessarily be the same for each of the pair; the weighting functions must each return the same partially-ordered result. Lastly, we replace the numeric comparison, $<$, by whatever partial comparison, $\sqsubset$, is appropriate to the chosen order.

Accordingly, we define the generalised weighting of A and B, with respect to weights ω_i and difference functions d_i as $A \bowtie_{\omega d} B$, where the elements are pairs of A and of B, and the partial order is as follows:

$$(x_1,x_2) <_{\bowtie\omega d} (y_1,y_2) = \omega_1\,(d_1(y_1, x_1)) \sqsubset \omega_2\,(d_2(x_2,y_2))$$

As with the numeric case, it is sufficient to perform weighting only once, on a relative basis. That is, ω_2 can simply be omitted from the above, and a relativised ω_1 used instead. However, in some cases, it may be more convenient to retain two separate weighting functions, as above. Since weighting functions here accomplish both the intuitive weighting functionality, and often will also involve a change of type, this definition may be more natural and convenient, as it avoids constraining the second difference function and the single weighting function to return the same type.

It is straightforward to see that generalised weighting is indeed a generalisation of numeric weighting: we simply put $d_i(x_1,x_2) = x_2 - x_1$, and $\omega_i(d) = w_i.d$. Since the result type is numeric, then $\sqsubset$ is simply the usual < operation.

The purpose of using difference functions can be seen from the following example: let us extend the previous hotel amenities example, to also include the point `MiniBar`. Now, suppose we wish to measure the difference between the degrees of similarity `{MiniBar}`, and `{EnSuite, TV}`. We can do this fairly naturally by (asymmetric) set difference $d(x,y) = y - x$, giving us in this case the set `{EnSuite, TV}` as the difference between the first and the second. This is quite distinct from, for example, the points `{EnSuite}` and `{EnSuite, TV}`, where the difference would have been the set `{TV}` – the first two degrees of similarity are, intuitively, more different than are the latter pair, and the differences we calculate accurately capture this. This distinction which would have been lost had we first converted each set to a number – say, its cardinality – as we would have mapped the former set in each case to 1, and the latter to 2. So the numeric differences obtained from subsequent subtraction would have been identical, in contrast, implying, incorrectly or at least very misleadingly, that the two pairs of similarity values were equally different.

Product

One means of combining metrics is to make no determination as to their relative importance; this may because they are genuinely of 'equal' importance, or it may be that for purposes of facilitating interaction with the user, elicitation of relative degree of importance has been deferred. Using identical numerical weights is the obvious approach, but it often does not, in fact give satisfactory results – this is typically because of what we have termed 'spurious precision' [FB00]. Weighting gives the effect of pro-rating one metric against the other: a little better in one compensates for being a little worse in the other, a lot worse balances out a lot better. This introduces potentially very fine distinctions between these 'trade-offs', on which to decide which cases are strictly the most similar to the probe.

We have previously defined an alternative. In essence the approach is to allow no trade-offs between disparities in the two metrics: the method will consider even a marginal improvement in one to compensate for a large degradation in the other, and include all such cases in the result set. Thus the 'best' matches are those which are 'best in any one of a number of conflicting respects'. We term this combining form *product*.

We now define the product of two similarity values. Suppose m_1 returns a values x_1, of type A, and m_2 returns x_2, of type B. Then we construct as a composite similarity results value, the product of A and B, using pairs (x_1, x_2), ordered according to the relation $\sqsubseteq_{AxB}$, which we now define, in terms of the (partial) orders on A, and on B.

$$(x_1,x_2) \sqsubseteq_{AxB} (y_1,y_2) = x_1 \sqsubseteq_A y_1 \;\; \& \;\; x_2 \sqsubseteq_B y_2$$

Intuitively, one point is less than or equal to another only if *both* components of the first are less than or equal to those of the second. If for two points, one is less on one component, and one is less on the other, then the paired points will be *incomparable*.

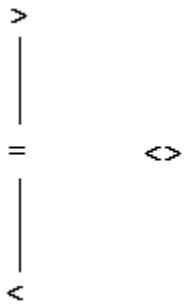
At first sight, this seems very different from a weighting-style combining form: there is no overt weighting, or summing at work in the definition. However, an essential similarity exists in that it works by componentwise comparisons, which are then themselves compared and combined. And indeed, we can utilise this to define product *as* a generalised weighting combinator. To see this, we begin by defining some difference functions. Note first of all that the degree of difference is not significant here: there is no distinction made between components which are only marginally comparable, as opposed to those which are greatly so. Accordingly, we define difference functions which have only four distinct results, according to whether the first argument is less than, equal to, greater than, or incomparable with the second.

$d_i\ (x,y) = [<]$, if $x \sqsubset y$
$d_i\ (x,y) = [=]$, if $x = y$
$d_i\ (x,y) = [>]$, if $x \sqsupset y$
$d_i\ (x,y) = [<>]$, otherwise

No weighting is necessary (or indeed meaningful) here, so the weighting functions are each simply the identity:

$\omega_i\ d = d$

The final matter is the ordering to be put on the difference function's result type. As might have been intuitively expected, we put $[<] \sqsubset [=] \sqsubset [>]$; this gives the effect that if at least one component is less than its counterpart, and the other is either less than or equal to the corresponding one, then the differences will similarly be comparable, hence so will the points being compared. Note that if either component is incomparable, then so are the pairs; accordingly, the difference point [<>] is *not* put comparable to any other points, so that this effect is indeed achieved. Thus the resulting partial order has a chain of three points, and a single isolated point, as shown below:



This order is somewhat unusual in having neither a unique top nor bottom element. This does not present any difficulty for our technique, however. It might best be regarded as a technical device for providing a weighting combinator with behaviour corresponding to that of the desired form, product.

Prioritisation

Another combining form from our earlier work is generalised prioritisation, which has as a special case, strict prioritisation. We demonstrate that these too can be regarded as instances of generalised weighting. We first of all describe strict prioritisation: the idea is simple that one criterion is certainly more important than the other, but the second is not to be discarded entirely. Accordingly, we rank according to the result of the first metric, unless that gives a dead heat, whereupon we use the second. This is the same as the lexicographical order mentioned earlier, except that we explicitly consider the possibility that either component order be partial, resulting in a final order that is also partial. We define the strict prioritisation of A over B, A≫B, by pairs of A and B, as with product, but with the following partial order:

$$(x_1,x_2) \sqsubseteq_{A \gg B} (y_1,y_2) = x_1 \sqsubset_A y_1 \ \vee \ (x_1 = y_1 \ \& \ x_2 \sqsubseteq_B y_2)$$

Although it is well-defined on partial orders, it introduces no partiality itself; given two total orders, it yields a third total order. If, however, the first components are incomparable, or the first are equal and the second incomparable, then the compounds are incomparable also.

Conceptually, this is similar to weighting with an arbitrarily large weight. However, the two are not precisely the same in the general case, as choosing a precise weight to give this effect is tricky, and error-prone. Infinite weights have the effect of completely negating the effects of the second metric, even when there is a first-metric tie; choosing a finite, but sufficiently large weight can only be done by knowing something about the sensitivity and granularity of the metrics' result types.

Generalised prioritisation extends the idea of strict prioritisation as follows: the less important metric will be taken into account not simply when the first is exactly equal, but when it is within a specified degree. We specify this degree by an *indifference relation*, $\simeq$, so-called as it characterises the region in which we do *not* wish to prioritise on the first component. We define the generalised prioritisation of A over B, with indifference $\simeq$, as A $>_{\simeq}$B. Again the result type consists of pairs of *A* and *B*, ordered as follows:

$$(x_1,x_2) \sqsubseteq_{A >_{\simeq} B} (y_1,y_2) = x_1 \sqsubseteq_A y_1 \ \& \ (x_1 \simeq_A y_1 \Rightarrow x_2 \sqsubseteq_B y_2)$$

That is, in order for two composite points to be comparable, then the first component must be in any case comparable, and if that component is within the specified degree of indifference, then the second components must be comparable too.

We discuss this in detail in [FB99a]; it is motivated by considerations of using combining forms which interact well with user notions of how metrics might be combined, though here it is sufficient to note that is a possible means of combination, which is distinct from weighting, both in being quite naturally defined on more general data types, and in giving variant results even on numeric data.

To cast prioritisation as a generalised weighting combinator, we take a very similar approach to that taken with product. We must make additional allowance for an extra

case, specifically for where prioritisation is to take place. If we are not indifferent on the first component, then the behaviour may differ, hence, we extend the first difference function (only) to identify that as a distinct case:

$$d_1(x,y) = [<<], \text{ if } x \sqsubset y \;\&\; x \not\simeq y$$

with the remaining cases of d_1 and all cases of d_2 being as per the product definitions. The same trivial weighting functions are used, as before. The partial order on the result type is the also the same on the original four points, and is extended as follows: $[<<] \sqsubset [<]$, $[<<] \sqsubset [<>]$. That is, [<<] is added as a new bottommost point.

Returning to strict prioritisation, we note that it can simply be seen as a special case of generalised prioritisation, by putting $(\simeq) \triangleq (=)$: that is, we are 'indifferent' between two elements only if they are exactly the same. Hence it too is a special case of generalised weighting.

Non-numeric weighting

The examples we have looked at so far are all known techniques, either which are standard practice, or from our own previous work. Beyond the application of weighting to non-numeric types by numeric conversion functions discussed in the second section, we have introduced no new combining form: rather we have constructed an 'umbrella' description for those techniques. We believe, however, that there exist a considerable number of possible means of combining metrics described by our formulation, which can be tailored for particular purposes with a greater degree of suitability and flexibility than generic techniques such as weighted sum. In this section, we suggest one particular novel combinator, with a view both to proposing it for its own utility, and to illustrate the power of our generalisation.

Our example is suggested by the work of Vollrath [Vo00]; he proposes the case where two requirements such as price and quality are in conflict. On one level this might be interpreted as being as simple as strict prioritisation, or perhaps product. A common feature of such examples, however, is that while there is a trade-off between these aspects, it is neither a linear one, nor is it a crisp prioritisation. Often, there is a certain threshold value on, say, quality, corresponding to a degree of quality which is strongly desired; until this level of quality is attained, price is less important. But beyond this threshold, 'bonus' quality is regarded as less crucial, and price consequently more so. In other words, there is a sort of bipartite behaviour, where the metric acts differently above and below the specified threshold.

For ease of presentation, we consider here a very simple case, where each of the basis metrics returns a numeric type. However the technique is by no means specific to numeric results, or indeed to total orders – that would, however, lead to a more complex example. Considering our earlier example of a hotel case base, suppose we have metrics on two more attributes – price, and star rating. A particular user may consider a certain number of stars – 2, say – to be crucial, and more important than price, but star rating above that to be less important, while not irrelevant. That is, beyond 2 stars, there exists a certain trade-off between numbers of stars, and price. In principle, this trade-off could itself be captured by one of several distinct combining forms, such as the ones previously discussed, but, again for ease of presentation, we

will assume a conventional weighted-sum treatment here. For example, a relative weight of 30 would mean that each 'extra' star was effectively equivalent to a room-rate higher by thirty pounds.

We define the thesholded weighting of A and B, with threshold θ on the first parameter, and weights w_i, by $A \bowtie_{\theta w} B$ in terms of our generalised combinator. We begin by defining the required distance functions, first on the component with threshold:

$$d_1\ (x,y) = \Theta_\theta\ (y) - \Theta_\theta\ (x)$$

where Θ is a thesholding function. Its role is to split a value into those parts above the given theshold, and those below. We define:

$$\Theta_\theta\ x = (min\ x\ \theta,\ max\ x\ \theta)$$

and where the subtraction in the above is defined in the following, pairwise, fashion:

$$(x_1,x_2) - (y_1,y_2) = (x_1-y_1,\ x_2-y_2)$$

Difference on the second component is straightforward.

$$d_2\ (x,y) = y-x$$

Now we need to devise weighting functions. These must perform two roles: firstly, an actual numerical weighting; and secondly, we need to ensure that both return the same type. In this case, we seek to retain the strict prioritisation of the first component, so we chose that type to return from both weighting functions. Accordingly, we define:

$$\omega_1\ (d_1,\ d_2) = (d_1,\ w_1\ .\ d_2)$$

and:

$$\omega_2\ d = (0,\ w_2\ .\ d)$$

That is, in both cases, the second components are traditionally weighted; for the first components, we retain the part to be prioritised, where present, and where not, we create a 'dummy' value, to ensure type compatibility. By choosing the value 0, we ensure that the second components are only significant where the first component from the first difference function is also 0, that is, the original case attributes were equal on the most important part of the range, as desired.

To complete the example, we need only define the constituent metrics: we pick a simple numeric similarity for the price, negated absolute difference:

$$m_2\ x\ y = -\ (abs\ (x-y))$$

For the second, the probe plays no significant role, so we define:

$$m_1\ x\ y = y$$

This completes the definition of the required metric, *m*: we now look at an example of applying it. Suppose we put a threshold of $\theta = 2$, and select weights $w_1 = 1$, $w_2 = 30$, then we obtain the effect described above: the quality threshold is two stars, and beyond that, a star is 'worth' the same as £30 extra cost. Consider a probe case: p = (2 stars, £30). (For conciseness, we present cases as if they were pairs, though in practice they will of course also contain many other attributes.) Now suppose that our case base contains the following cases: a = (1 star, £30), b = (2 stars, £70), c = (3 stars, £60) and d = (4 stars, £80).

Comparing each case to the probe in turn, we obtain the following similarity values: $s_a = m\ p\ a$ = (1 star, £0), $s_b = m\ p\ b$ = (2 stars, £40), $s_c = m\ p\ c$ = (3 stars, £30) and $s_d = m\ p\ d$ = (4 stars, £50). To find out which of these are the best matches, we compare these in turn using the partial order $\sqsubset_{\bowtie \& w}$. First of all, applying the thresholding function to the first component of each in turn, we obtain $\Theta_2\ s_a = (1, 2)$, $\Theta_2\ s_b = (2,2)$, $\Theta_2\ s_c = (2,3)$, and $\Theta_2\ s_d = (2,4)$. From this, we can immediately tell that a is he least similar case, as it has the lowest value on that portion which we are to prioritise. This is despite it being only one star less than case b, and £40 cheaper, which would make it better by a weighting-only method. The other points, being equal on the below-the-theshold, are then ordered by the remaining attributes, appropriately weighted. So case b is worse than case c: it is worse on both components. Case d is in turn better than case c; it is one star better, above the threshold, and only £20 more expensive, less than the weighting factor.

Conclusions

We have presented a generalised characterisation of different means of combining distinct case base metrics. We have demonstrated that this is broad enough to encompass both the standard weighted-sum technique, and our own proposed methods of combining non-standard metrics, product and prioritisation. Furthermore, we have seen that this methodology is still particularised enough to be suggestive of other instantiations, combining metrics in distinct ways to achieve a given effect with regard to user expectations.

Implementation work is underway on a case-base system utilising our framework and the combining forms discussed herein. This will facilitate subsequent empirical investigation, in which we propose to elict actual user queries, and compare their expression with more conventional methods. We hope that this will provide both verification and application of the ideas presented.

References

[FB99a] A. B. Ferguson and D. G. Bridge. Generalised Prioritisation: A New Way of Combining Similarity Metrics. *Procs. of Tenth Irish Conference on Artificial Intelligence & Cognitive Science*, pp.137-142, 1999, Derek Bridge, Ruth Byrne, Barry O'Sullivan, Steven Prestwich and Humphrey Sorensen (eds.).

[FB99b] A. B. Ferguson and D. G. Bridge. Options for Query Revision when Interacting with Case Retrieval Systems. *Procs. of Fourth UK Case-Based Reasoning Workshop*, pp.1-17, 1999, Ian Watson (ed.). also published in: *Expert Update Vol. 3*

No. 1 Special Issue on Case-Based Reasoning, pp. 16-27. Watson, I. (Ed.). ISSN: 1465-4091

[FB00] A. B. Ferguson and D. G. Bridge. Indifference Relations: Being Purposefully Vague in Case-Based Retrieval. To appear in: European Workshop on Case-Based Reasoning, 2000 (EWCBR2K).

[OB97] H.R.Osborne and D.G.Bridge: Similarity Metrics: A Formal Unification of Cardinal and Non-Cardinal Similarity Measures, D.B.Leake and E.Plaza (eds.), *Case-Based Reasoning Research and Development* (Procs. Of Second International Conference on Case-Based Reasoning), Lecture Notes in Artificial Intelligence 1266, pp.235-244, Springer, 1997

[Pl95] Plaza, E.: Cases as terms: A feature term approach to the structured representation of cases, in M.Veloso & A.Aamodt (eds.), *Case-Based Reasoning Research and Development* (Procs. of 1st International Conference on Case-Based Reasoning), LNAI-1010, pp.265-276, Springer, 1995.

[Vo00] Vollrath, Handling Vague and Qualitative Criteria in Case-Based Reasoning Applications, *8th German Workshop on Case-Based Reasoning*, 2000.