Humanist Computing: Modelling With Words, Concepts, and Behaviours

Jonathan Rossiter

AI Research Group,
Department of Engineering Mathematics
University of Bristol,
Bristol, BS8 1TR, UK,
Jonathan.Rossiter@bris.ac.uk

Abstract. In this paper we present a new approach to the modelling of data and knowledge called Humanist Computing. We describe how the general principle of Humanist Computing, namely the modelling with words, concepts and behaviours defines a hierarchy of methods which extends from the low level data-driven modelling with words to the high level fusion of knowledge in the context of human behaviours. We explore this hierarchy and focus on a number of levels in the hierarchy. For each level we describe the general Humanist Computing approach and give specific examples based either on approximations to the real-world or the real-world itself.

1 Introduction: Motivating Humanist Computing

Humanist computing seeks to combine the following three elements of soft computing; 1. modelling with words, 2. conceptual specifications based on these words and their associated perceptions of natural concepts, and 3. simple models of human behaviour based on these conceptual definitions. Humanist computing is a methodology inspired by observations of human perceptions and behaviours. The general principle is that humans are good at solving problems, summarising information and making decisions with incomplete information. It seems natural therefore that when we are faced with the problem of building artificial systems to perform similar tasks we take some inspiration from the observation of humans. Although humans are not infallible (far from it!) we must surely have some faith in our own behaviours and reasoning processes.

Of course, human behaviour is a function of its environment and consequently a high degree of specialisation has evolved in some of our behaviour. We should therefore say at the start that mimicking human behaviour has an impact on the type of problems humanist computing can solve. A very simple analogy is in the realm of hard computing. The arithmetic processing of the human brain can in no way match the raw power and determinism of modern computers. We would therefore not wish to model arithmetic computing using humanist computing techniques. On the other hand, humans are particularly good at operating in new, unfamiliar, and uncertain environments. These environments would seem

to provide good problem domains in which humanist computing could give us some advantages.

We can also motivate our development of humanist computing from the point of transparency. In an increasing number of critical applications it is important for us to be able to understand and evaluate the computer model that has been developed. Examples include automated medical diagnosis and engineering risk assessment. It would seem natural therefore to base these models on forms that are natural to humans. What could be more natural for us to interpret than concepts and behaviours that we all exhibit? This neatly leads to the emerging characteristic of transparency. Transparency in artificial systems enables us to verify the system and to build up confidence in its representation and reasoning mechanisms. In developing a humanist model our assumption of transparency can be extended so that even non-expert observers will be able to understand the underlying reasoning processes. The key tenet here is that any motivation for transparency in artificial systems can be logically extended to modelling in a human-like manner. In other words, when building artificial systems we can most readily understand those that behave in some way similar to ourselves. As illustrated in figure 1, humanist computing generates models from a combination of domain data, perceptions, concept definitions, behaviours and expert knowledge through uncertain machine learning, modelling with words and information fusion. The resulting models give us some new (and clear) *insight* into the problem domain. As such these models are *glass-box* rather than *black-box* in nature.

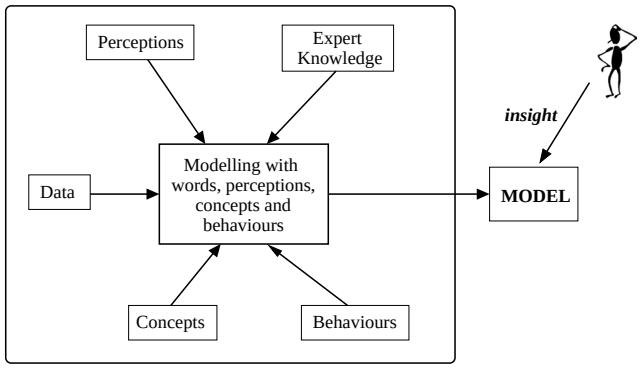


Fig. 1. Humanist Computing

As a final consideration with respect to this glass-box or transparency property, we might argue that any fully glass-box representation should provide the most clear, flexible and rich description of the model. In reality any computationally tractable calculus claiming to model semantics on words is likely to be severely restricted. Consequently the flexibility and richness of the representation suffers. In the case of modelling with words we may wish to represent our

words by fuzzy granules, and to subset these into just those granules representing simple perceptions. Given such a restriction our corresponding vocabulary of words will be much reduced. As a result we can only go as far towards true transparency as a *murky-box*, or *foggy-box* model. That said, we would advocate that in practical situations, a murky insight into a problem domain is better than no insight at all.

2 A Hierarchy of Methods for Humanist Computing

In order to fully describe a framework that will enable humanist computing we must consider how we can relate computing with words, perceptions, concepts and behaviours. To this end we define a simple ordering of these soft methods based on their modelling forms. We can thus illustrate how a mechanism for humanist computing can be built up from these soft computing paradigms.

At the lowest level in this humanist computing model ordering we will take the primitive to be the uncertain definition of a word. This is taken to be the granular definition put forward by Zadeh [24]. Such granular words are defined with labels such as $small$, $medium$, or $large$. We then subsume this form with the linguistic definition of perception also proposed by Zadeh [25]. Here the granular words are defined by labels such as $small\_weight$ or $high\_cost$. We now extend the ordering with a higher level form defining a concept. At this level a conceptual definition defines a natural, observable concept which can be described in terms of granular words and granular perceptions. Concepts also allow us to introduce rules linking perceptions and words. For example the concept *tall man* can be modelled by a soft concept that not only includes the granular perceptions $tall\_height$ but also the perception $large\_feet$ and the implication rule $large\_feet \rightarrow tall\_height$.

Finally we subsume our uncertain concepts inside the form of an uncertain behavioural model. A behavioural model extends a concept with some representation of human behaviour. These behaviours can range from the extremely objective and specific, such as driving above the speed limit, to the extremely subjective and unspecific, such as feeling jealousy or fear.

Lets us formalise this a little by constructing the hierarchy H of humanist computing defined using the ordering discussed above. Consider the set of granular words W. Using W we can perform the machine learning operation of inferring relations between this words. We term this process modelling with words (represented by the symbols M_W) and it, together with the set W, defines the lowest level class C_W in our humanist computing hierarchy H. Now consider the immediate superclass of C_W which defines perceptual computing. Here we define the class C_P which contains a set of perceptions P defined on granular words W. The operation on C_P is termed modelling with perceptions (i.e. M_P), that is, the inference of relations between perceptions. We next extend this class into the class of conceptual definitions C_C which contain richer conceptual descriptions of the real world than single perceptions allow. These richer descriptions typically define some structured conceptual relationships through

notions of dependence or causality. Where M_P and M_W can be defined, for example, using basic Bayesian operations the class C_C requires a richer representation and also a more complicated reasoning process such as those used in Bayesian networks [18] or uncertain conceptual graphs [6]. C_C therefore contains the set of conceptual descriptions and operations on these can be termed modelling with concepts, or M_C. Likewise the operation M_C on C_C is the inference of relations between concepts. Finally we extend our class of conceptual definitions with simple descriptions of human behaviour to give the top level class C_B. C_B therefore contains the set of simple behavioural models B and operations on C_B can be termed modelling with behaviours, or M_B. It should be clear that we have formed a hierarchy defined by a simple subsumption relation.

Class	*Focus*	*Operations*	*Operands*	*Examples*
C_B	Behaviours	M_B	$\{B, C, P, W\}$	decision making
$\downarrow$				
C_C	Concepts	M_C	$\{C, P, W\}$	$\{manager,\ car\}$
$\downarrow$				
C_P	Perceptions	M_P	$\{P, W\}$	$\{hot,\ fast\}$
$\downarrow$				
C_W	Words	M_W	$\{W\}$	$\{small,\ medium\}$

Table 1. The Humanist Computing Hierarchy

Table 1 illustrates the four classes in the humanist computing hierarchy, their scope of operation, and the operands that they act upon. The symbol $\downarrow$ is taken to mean that the class above subsumes the class below.

Let us take for example the uncertain modelling of components of a car. At the lowest level measurements of individual components within the car will involve numerical uncertainty and this can be modelled using words (i.e. *modelling with words*, or M_W). Now let us consider a specific temperature sensor in the engine of the car. Here we are dealing with a perceptual quantity, namely temperature. At this level we would model the sensor using perceptions (i.e. *modelling with perceptions*, or M_P). At the next level, we could consider the whole car. here the concept of a car starts to come into play and we must consider the typical components that describe a car. For example, cars usually have four wheels, they usually have at least two seats, and so on. Here the concept of a car is a structured definition based on the components and corresponding uncertainties that go to make it up. At this level we need to model the uncertain concept that defines a car (i.e. *modelling with concepts*, or M_C). At the very highest level of humanist computing we also model behaviours, such as driving the car. Here we need to model not just the concept, perceptions, and words that describe the car, but also the behaviours that humans exhibit when performing the operation of driving. In this example we may also wish to divide driver behaviour into a number of prototypical classes such as *safe driver*, *aggressive driver*, and so on.

This hierarchy of modelling forms in some way shares some common components with one view of the knowledge induction mechanisms in humans. This perspective [22] links perceptions of the real world through a series of inference steps to the models of the real world that humans naturally construct. There are four key stages in this process; *belief*, *reasoning*, *evidence* and *perception*. The inference path from real world to model is shown in figure 2.

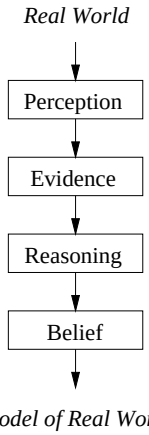


Fig. 2. Conceptual dependencies

Here the real world provides the measurable quantities (or data) with which we can perform modelling with words. Perceptions in figure 2 enable us to start to interpret the data from the real world. In addition, perceptions themselves can give us some measure of conceptual structure between words (for example we perceive that a *big* person has *large* shoes and is *tall*). Processing at the higher levels (reasoning and belief) involves the processing of concepts and behaviours.

This process of generating a model from perceptions of the real world assumes a linear flow through the four concepts above. More complex models can include other links, such as from belief to reasoning. In such a case an existing belief may colour how new sensory information is reasoned with.

The one key similarity between the dependencies in figure 2 and the ordering in table 1 is the inclusion of some model of perception. Since this is a focal point in both our interaction with the real world and our attempts to model it with humanist computing it bears some more careful examination.

2.1 Perception

In our daily lives we humans constantly rely on our perception systems. Through familiarity we rarely ponder on how sophisticated they are. The significant point

to make is that, no matter how we view our perception systems they are the only route that information can take from the outside world into the brain.

At the lowest level our perception systems are merely measurement devices, or transducers. The eyes turn photons into electrical signals and the ears turn sound energy into similar electrical signals. Experiments have shown that this transducing is not the complete picture. The eyes and the optic nerve, for example, contain some processing elements which detect important features. This processing also serves to reduce the amount of raw information fed to the brain.

At a higher level the perception system reaches what Goldman [12] calls a *terminus*. The terminus of the perception system is the final high level perceptual representation we use. This terminus of perception can involve complex structuring and organisation within the brain.

Take for example a face. The raw information entering the eye is simple colour-point information. The perceptual processing performed in the eye and brain may lead to a terminus of perception based on lines, points, regions of colour, relative position etc.

Figure 3 shows how a dictionary of perceptual concepts can be built up at the terminus of perception using simple two-dimensional shapes.

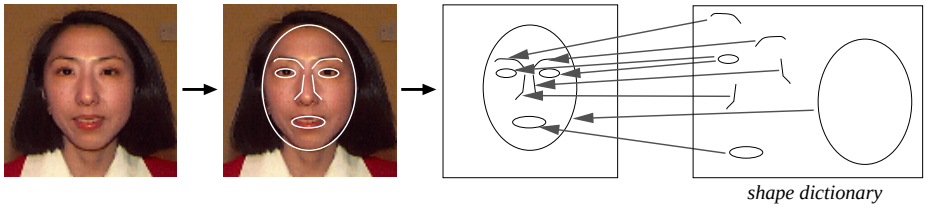


Fig. 3. Simple dictionary of perceptual concepts

Clearly in this example the perceptions that may be present at the terminus of perception in humans are somewhat more complicated than this typically modelling using computing with perceptions. In other words, the rigid ordering in table 1 is far less well defined in the human perception system and we could view a perception as a somewhat higher level concept than previously discussed. It is with this in mind that in later sections we give examples of modelling with concepts and words which, to some degree, encompass modelling with perceptions without explicitly referring to perceptions. In effect we have a reduced ordering where perceptions can be modelled either with simple words or with more complex concepts. This is shown in table 2.

2.2 Practical Approaches to Humanist Computing

In the next sections we will illustrate three approaches to modelling and learning from data using humanist computing which all involve modelling with words, concepts and behaviours:

Class	*Focus*
C_B	Behaviours
$\downarrow$	
C_C	Concepts $\cup$ Perceptions
$\downarrow$	
C_W	Words $\cup$ Perceptions

Table 2. The Humanist Computing Hierarchy

1. An object oriented approach to modelling with words.
2. A method for structured information fusion based on specificity.
3. A behaviour based approach to uncertain belief updating.

3 An Object Oriented Approach to Modelling With Words

Here we review our recent work on a framework for uncertain taxonomical knowledge representation and reasoning. We discuss how the implementation of these theories in Fril++ [3] has enabled the development of a toolkit for objected oriented machine learning with words. We focus here on machine learning by fuzzy rule induction. This object oriented approach to model induction using words is in keeping with humanist computing because of the emphasis on learning fuzzy rules which naturally lend themselves to a linguistic interpretation.

As presented in [20] an object oriented approach to modelling with words has the following features:

1. **Clear and natural representation**
   We apply class hierarchies to many parts of our daily lives. Indeed one might argue that uncertain taxonomies are *necessary* for us to function in, and understand, a world with so many varied objects. The shear impossibility of developing special responses to every perception has led us to develop a higher level taxonomical view of the world which does no require specific scientific knowledge. For example, we may class trees into 'large trees' and 'small trees'. We may then split 'large trees' into 'quite large trees' and 'very large trees', as shown in figure 4. The important thing to see here is that the linguistic terms commonly used in computing with words (large, small, very large, etc) may also be integral to class descriptions. Clearly we need underlying mechanisms for handling uncertainty in words in these hierarchies. This will be discussed in the next section.
2. **Scalability of knowledge representation**
   Object oriented modelling with words naturally has a measure of scale through different perspectives of the hierarchy. If we build a model that has hundreds of classes in our hierarchy we can focus on the appropriate level of the hierarchy for the appropriate linguistic description of the model we are looking for.

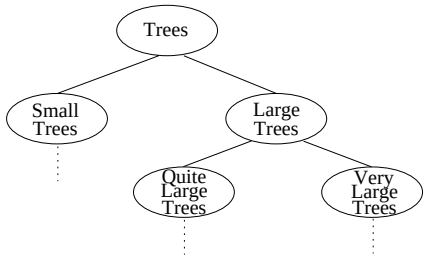


Fig. 4. A hierarchy of trees

In effect we can summarise the model at as many levels as there are levels in the hierarchy. A *complex* summary involves knowledge from *lower down* the hierarchy while a more *general* summary involves knowledge from just the *top* of the hierarchy.

3. **Power of inheritance, overriding, encapsulation and information hiding**

   From a knowledge representation point of view inheritance helps reduce unnecessary repetition in a class hierarchy. This aids model conciseness. Overriding, on the other hand enables us to form a richer hierarchical model, and in extreme cases to implement forms of non-monotonic reasoning.

   Encapsulation (the grouping of methods and attributes within a class) and information hiding (restricting access to properties with a class) are features that can make the final model more robust. Although these are essentially programming aids they are important in modelling in words where the models are produced to solve real-world problems.

3.1 The Underlying Framework

We have described a new approach to uncertain object oriented representation and reasoning in [8] and have implemented these theories in Fril++, the uncertain object oriented logic programming language. Fril++ enables us to define hierarchies where classes can contain uncertain properties (attributes and methods) and objects can have uncertain memberships in classes. Interval probabilities are used to represent uncertainty in class memberships and property applicability. Intervals across the [0, 1] domain give us a richer representation than point probabilities.We adopt a subsumption partial ordering in place of a fuzzy subset ordering for the sub class relationship. This avoids the problem of a symmetric sub class relation which could generate a network rather than a hierarchy, and thus reduces computational complexity. Membership of an object to a class can be calculated by matching uncertain properties followed by the application of an interval probability form of Jeffreys rule [15].

3.2 An Extendible Object Oriented Data Browser

We will now give an example of modelling with words in our uncertain object oriented language, Fril++. We overview an implementation of a tool for dataset modelling and data mining. The theory behind the object oriented data browser builds on research into a non-object oriented data browser in [2]. The data browser tool is required to approximate functions and to learn classification relationships from example databases.

To provide a clear and extendible hierarchy for the data browser we consider the general case of a machine learner. We split a typical machine learner into a number of parts; the inducer, the tester, and the data. We abstract these three components into high level Fril++ classes as follows.

- **the data**
  The data class is an abstraction of all forms of data into a two dimensional table. Tuples are ordered and attributes can be named. Each attribute can have a corresponding universe specified for it. Each of these universes can be split into any number of partitions. Partitions can be crisp or fuzzy. In the case of fuzzy partitions the partitions can define a list of words. In such a way the 3-attribute data object shown in figure 5 is described by the individual data points and the fuzzy words on the axes.

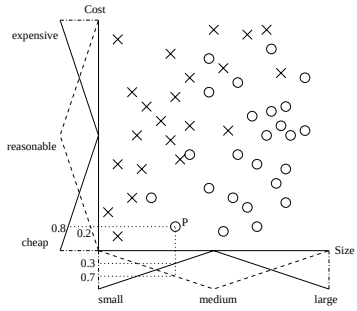


Fig. 5. A Dataset object

  We can represent any point value by a fuzzy set on words. Take for example the point P in figure 5. Within the Size domain the point P can be represented by the fuzzy set on labels, $\{small : 0.3, \quad medium : 0.7\}$.
- **the model**
  The model is the basic class for all objects that predict or classify from data. The model object, once built by an inducer, models or summarises the data that was used to build it. In this paper we concentrate on fuzzy rule models that are linguistic in nature and are expressions on words.
- **the inducer**
  The inducer class defines all the objects that perform induction from data. An inducer operates on data and generates models. These models can then

be used to classify or predict from new data. For all examples in this paper we use a simple example of an inducer that constructs fuzzy rulesets from data. The inducer class can be subclassed to implement a more complicated induction process.

– **the tester**
Once the model object (a fuzzy rulebase in our examples) has been built it can be tested using an object of the tester class. A tester object takes the model object and applies data to the model. Results of classification or prediction can be aggregated and presented as rms errors, classification errors and so on.

3.3 Data Modelling Examples

We present some examples of modelling with words using the extendible object oriented data browser. Each attribute in both datasets are partitioned into triangular fuzzy sets representing words.

2D Shape Databases Here we learn two 2d shapes; a doughnut, and a figure-eight shape. Using the learnt models we attempt to reclassify points to determine if they are legal (part of the shape) or illegal (outside the shape).

– Figure-eight
This is a classification problem based on a simple two dimensional figure eight shape. This is effectively the XOR problem, which is particularly difficult for decomposing machine learning algorithms like this one. Figure 6 shows the form of the shape and figure 7 shows the points from a regular grid that have been classified as legal. Notice the symmetry in the points across both X and Y while the actual shape is not symmetrical in X or Y. This is due to the decomposing nature of our simple rule induction algorithm. Of course, there are many more effective machine learning algorithms available but we choose this one here for its simplicity and for its use in the following section on information fusion. Even so this model correctly classifies points in the figure eight with an accuracy of 84%.

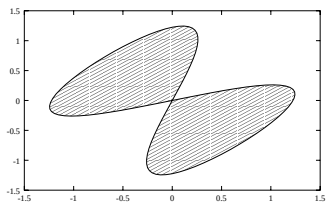


Fig. 6. Figure eight shape

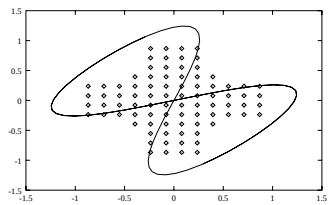


Fig. 7. Learnt figure eight shape: 84%

– Doughnut
  This is similar to the figure-eight above, again generating decomposition errors when learning using X and Y independently. Figure 8 shows the form of the shape and figure 9 shows the points from a regular grid that have been classified as legal. The decomposition error causes the shape to deform about the sides. This is simply because of the number of illegal class points in those projected regions swamping the legal points. The learnt model of the doughnut classifies points correctly with an accuracy of 85.5%.

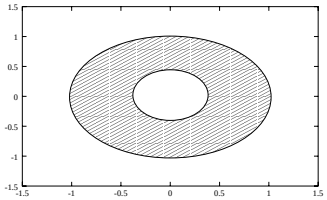


Fig. 8. Doughnut shape

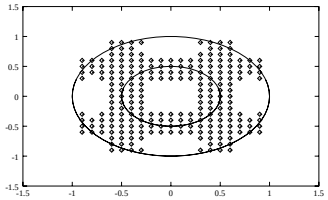


Fig. 9. Learnt doughnut shape: 85.5%

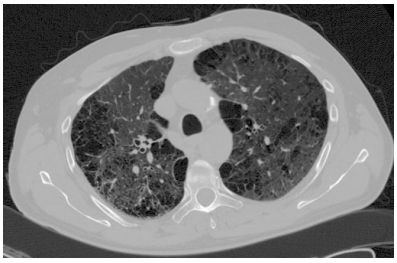
Facial Feature Database This database contains 168 instances of 18 measurements on human faces. The goal is to classify instances into sex. See [5] for more information about this dataset. The dataset was split randomly into a training set of 138 samples and a test set of 30 samples. Each of the 18 attributes was partitioned into 20 triangular fuzzy sets, with 50% overlap, representing 20 words. The generated model was represented by two Fril fuzzy rules, one for male and one for female. The generated model classified the test dataset with an result of 83% correct.

Lung Image Region Classification Recently the clinical diagnosis of lung disease has concentrated on non-invasive tests such as chest x-rays and CT and MRI scanning. For a single patient these methods produce a large number of cross sectional scans of the lungs. For example, a typical CT scan data set consists of over 12.5MB of data. The radiologist must examine all this data and generate a report that is typically 100 words in length and a diagnosis of typically 10 words in length. In information terms this is equivalent to an astonishing compression ratio of approximately 200,000:1. This is possible because the radiologist utilises two skills. Firstly, he uses his perceptual skills to process the low-level data in front of him, and secondly he uses his high-level expert knowledge and experience to guide his review and to help interpret the low-level data. Any approach to automating the review process must therefore involve a combination of low-level perceptual processing and high-level reasoning. This problem is therefore ideally suited to humanist computing

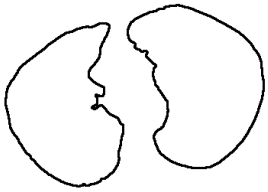
The first stage in this process is to extract lung fields (figure 10b) from lung CT scans (figure 10a). An image processing algorithm has been developed

which will process individual 2d images and detect lung fields. (For details of the background to this work see [13] and [14]). Unfortunately solely using image processing is not wholly effective and expert judgement is again required to sift correct lung fields from incorrect fields. This is shown in figures 11a and 11b where the trachea has been incorrectly included as part of the lung field. We therefore require a machine classifier which will determine if a candidate lung field is truly part of the lung.

We learn a model from a database of 1203 samples taken from 10 patients with varying degrees of lung disease. Each sample consists of the following eight attribute values; 3 coordinates for the centre of the region (X, Y, Z), the width and length of the region (H, W) and the number of pixels inside the region (SZ). Figure 12 shows 3d plots of the data projected into the 3-spaces $[X, Y, Z]$ and $[H, W, SZ]$. It is quite clear from figure 12 how the legal points (shown as diamonds) roughly form the structure of the lung shown in figure 11.

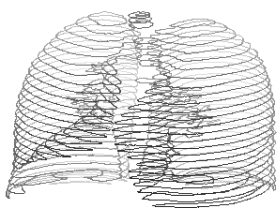


a. Typical CT scan

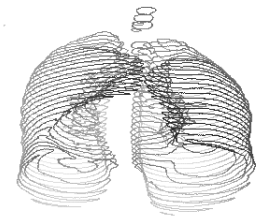


b. Corrected extracted lung fields

Fig. 10. Processing a 2d lung image



a



b

Fig. 11. Lung volume reconstruction

After learning a model for this data using our object oriented data browser we reclassify the database using the learnt model. The decomposing learner achieved an accuracy score of 87.4%. The misclassified points are shown in figure 13 as diamonds.

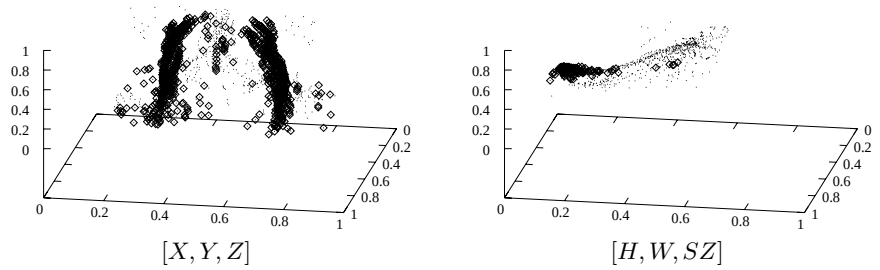


Fig. 12. 3d lung data visualisations

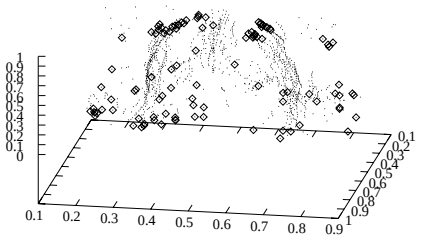


Fig. 13. Misclassifications (large points)

Given the decomposing nature of this machine learner and by inspection of figure 12 it is clear that this problem must be decomposable. Even so, in a medical environment the accuracy of a diagnosis tool must be extremely high in order to be put into everyday use. In the next section we attempt to improve on the accuracy of such models by fusing in high level, expert knowledge.

4 A Method for Structured Information Fusion Based on Specificity

In the previous section we showed how an object oriented implementation of machine learning can be used to learn models based on words from data. Let us now consider how we can merge such learnt models with higher level expert-derive knowledge. This is particularly relevant to humanist computing where humans can not only provide the inspiration for model construction but they may also have important high level knowledge to impart that may be difficult to learn directly from data. The goal for any fusion methodology is to perform better at decision making (classification, regression, etc.) using the fused model then using either of the component models on their own.

4.1 Deriving High-Level Knowledge and Low-Level Information

Hierarchies are, in themselves, consistent with expert and real world information. Humans naturally categorise and rank the world around them and this

impinges on the way we handle and communicate information. Lung disease for example can be split into sub classes such as cancer and emphysema. The class of cancer can be split up further into more specific sub classes, and so on. Clearly information higher up the hierarchy is more general and information lower down the hierarchy is more specific.

Given our chosen framework of uncertain class hierarchies for the representation of both expert and learnt information we must now consider how the two can be merged, or 'fused'. Any approach to fusion must accept an importance ordering on the information to be fused. Such an importance ordering gives us three base cases;

1. expert knowledge is given more weight than learnt information,
2. expert information has a lower weight than learnt information,
3. learnt and expert information are equally weighted.

We must also consider how expert or learnt information provides prior information that can be used in the construction of the class hierarchy, the transcription of expert information or the way in which learning is performed. Some circumstances may demand that we take expert knowledge to be background (i.e. prior) information which then influences the learning process. Alternatively we might say that learnt information has roots in empirical analysis and results from machine learning should be fed back to experts who can then revise the transcription of their expert knowledge. In this paper we give equal weight to expert and learnt knowledge.

Let us consider the concept T which we are required to model. The expert transcribes his knowledge of T and this is used to construct an uncertain class C_1. At the same time we perform some machine learning from a database of exemplars of T. The learnt knowledge is used to construct a second uncertain class C_2. We now require that C_1 and C_2 be fused. Here we crucially assume equal weight to expert and learnt information and ensure that C_1 and C_2 are constructed independently. Since, in our example, we know that C_1 and C_2 are related through the common concept T we rank the two classes at the same level in the class hierarchy, i.e. they have the same immediate superclass C_s. We can now use the mechanisms of uncertain hierarchical reasoning to query the fused hierarchy. Figure 14 illustrates this simple fusion of classes defining the same concept T.

4.2 Hierarchical Structure Fusion

Where in the previous section we fused a single class into a hierarchy at a known point we now consider the problem of fusing hierarchies where we have no prior knowledge of where the fusion should occur. In the simplest case we can just focus on the maximal superclass of each hierarchy. Let us take two simple hierarchical structures. Depending on prior bias and on the some measure of matching and/or consistency we may find that the most appropriate fusion is that shown in figure 15a. Notice that in this case the sub-hierarchies retain their structures. A more

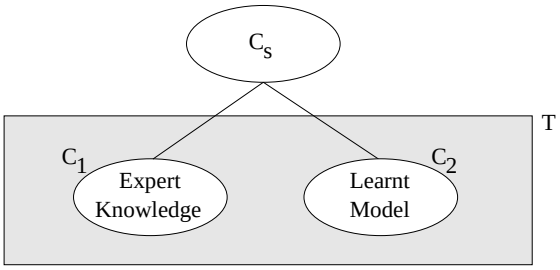


Fig. 14. Fusing two, conceptually related, classes

complicated form of hierarchical fusion may result in the structure shown in figure 15b. Here consideration has been given not only to the place at which to fuse the maximal superclass, but also the most appropriate places at which to fuse subclasses. Clearly this later approach is considerably more complicated.

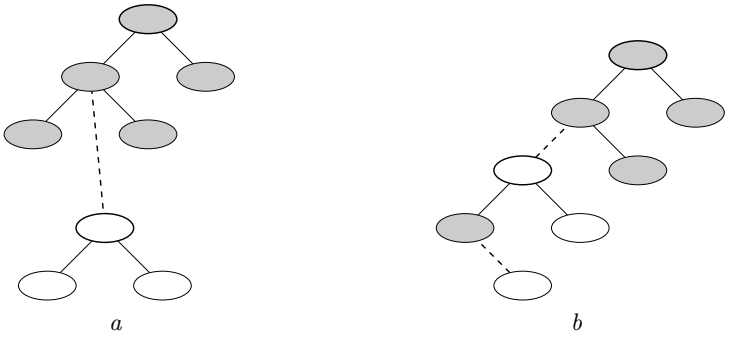


Fig. 15. Example fusions

We now suggest a number of ways in which the most appropriate fusion point for a maximal superclass could be determined. This is the simplest structural fusion task as illustrated in figure 15a.

1. **Concept matching.** By concept matching we mean the matching of all properties within the classes that contribute to the definition of the relevant ontology. For example, let us consider classes C_1 and C_2 where C_1 contains properties $\psi_{1,1}$, $\psi_{1,2}$ etc. and C_2 contains properties $\psi_{2,1}$, $\psi_{2,2}$ etc. By comparing $\psi_{1,1}$ with $\psi_{2,1}$ and $\psi_{1,2}$ with $\psi_{2,2}$ and so on, we can determine a match of C_1 to C_2. If properties are crisp values or fuzzy sets we may use semantic unification [4] to determine values for $Pr(\psi_{1,1}|\psi_{1,2})$, $Pr(\psi_{2,1}|\psi_{2,2})$, and so on. We can then insert the maximum superclass at the same level as the maximum matching class.

2. **Cross entropy.** Here we consider probability distributions across all possible objects that are, to some degree, consistent with a class definition. We use the principle of minimising cross-entropy [23] to compare a class definition from one hierarchy with a class definition from the other hierarchy. In this way we can determine and select the class having the minimal cross entropy, and hence having the maximal similarity.
3. **Empirical testing.** Perhaps the simplest comparison method is to force a classification or prediction test upon each class and determine the classes which are most consistent in test results. This may be computationally very expensive and, we would argue, is rather inelegant and brutish.

4.3 Default Reasoning for Inconsistency Resolution

Unfortunately, although the expert knowledge and learnt knowledge could be entirely consistent it is common for there to be some inconsistency between the two. This may be the result of errors or generalisations in expert knowledge, or there may be some problem with the learnt model, either through generalisation or noise. These inconsistencies are especially important in the context of uncertain class hierarchies. Here an object can have a graded membership in a class depending on property matching. It holds that since properties can be uncertain, an object may have a finite (although often very small) membership in any or all classes. In this case any reasoning about a property ψ must take into account *all* definitions for ψ in the hierarchy. Any reasonably large and diverse hierarchy is likely to incorporate some inconsistent information with respect to ψ. Of course there are mechanisms for dealing with independent and conflicting information (such as Dempster's rule) but these are often blunt instruments that do not take into account the information contained in the hierarchical structure itself. Another approach is to isolate the most inconsistent information and eliminate it from the reasoning until a consistent conclusion is produced. The problem with this approach is that we would effectively be searching for all possible consistent subsets of contributing evidence and selecting the most consistent. This is an exponential time complexity problem.

A better approach is to focus on a fundamental characteristic of our uncertain hierarchy: information lower down the hierarchy is more specific. Cao described a default reasoning algorithm for uncertain property inheritance using such a specificity ranking assumption which is polynomial in time complexity [7]. In this section we apply a modified form of Cao's algorithm to the problem of consistency resolution in fused uncertain hierarchies.

4.4 Two Simple Examples

We illustrate uncertain hierarchical information fusion using the two simple example class hierarchies of expert and learnt information shown in figures 16 and 17 which define the conceptual shapes shown in figures 6 and 8 respectively. The learnt models of the figure eight and doughnut are those that have been learnt using the object oriented data browser described previously

in this paper. In this way we show how object oriented modelling with words and information fusion fit into the proposed humanist computing framework. The expert knowledge is coded as simple fuzzy rules on a vocabulary of five words, $\{very\_small, small, medium, large, very\_large\}$, with the following linguistic equivalent forms:

Figure eight

Class is **illegal** if x is $small$ or $very\_small$ and y is $small$ or $very\_small$ or if x is $large$ or $very\_large$ and y is $large$ or $very\_large$ or if x is $very\_small$ and y is $very\_large$ or if x is $very\_large$ and y is $very\_small$.
Class is **legal** if x is $large$ and y is $small$ or if x is $small$ and y is $large$.

Doughnut

Class is **illegal** if x is $medium$ and y is $medium$ or if x is $very\_large$ or $very\_small$ or if y is $very\_large$ or $very\_small$ or if y is $large$ and x is $large$ or if y is $large$ and x is $small$ or if y is $small$ and x is $small$ or if y is $small$ and x is $large$.
Class is **legal** if x is $large$ or $small$ and y is $medium$ or if y is $large$ or $small$ and x is $medium$.

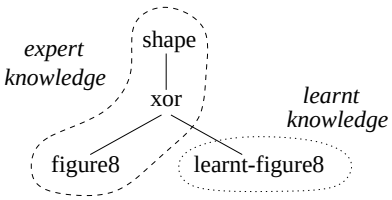


Fig. 16. Figure eight hierarchy

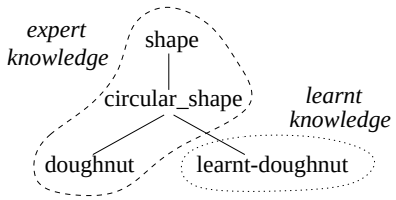


Fig. 17. Doughnut hierarchy

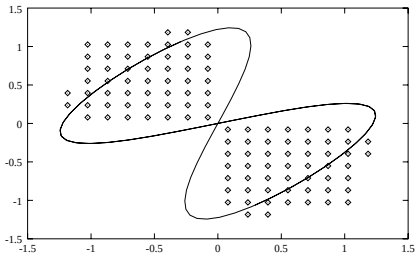


Fig. 18. Expert's figure eight: 81.5%

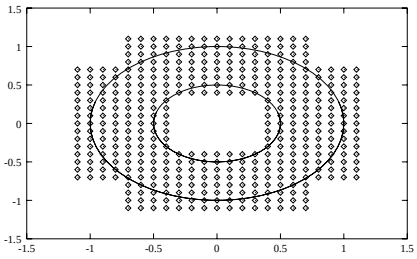


Fig. 19. Expert's doughnut: 81.3%

Figures 18 and 19 show the classification results using just the expert hierarchies. The accuracy values show the percentage of test points classified correctly. Notice how the expert knowledge, represented using simple linguistic fuzzy rules, captures the general shapes but, especially in the doughnut case, tends to overgeneralise the structure of the concepts.

When we fuse the expert and learnt classes and apply Cao's default reasoning algorithm to resolve inconsistencies we generate the classification results shown in figures 20 and 21. It is interesting to observe the effects of the fusion and reasoning processes on the structure of the classification results. Fusion gives slightly worse results for the figure eight problem and slightly better results for the doughnut problem. The important point here is that we have fused learnt and expert knowledge in a common hierarchical representation and have been able to draw measured conclusions about the shape of the concepts being represented through the stratified selection of consistent knowledge. Further to this, we can examine the linguistic rules that have been combined and this may be of great use to experts who wish to learn more about a problem domain from the learnt model.

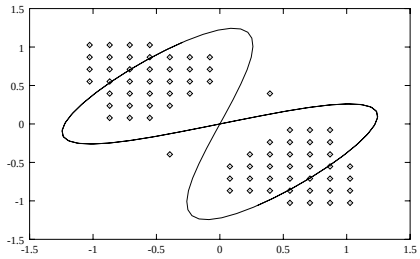


Fig. 20. Fused figure eight: 79.5%

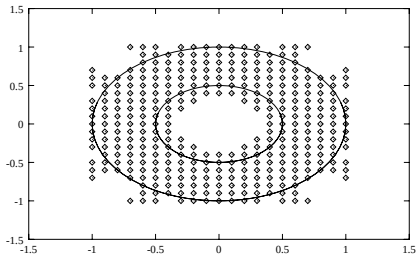


Fig. 21. Fused doughnut: 88.8%

4.5 An Alternative Default Fusion Strategy

In Cao's inconsistency resolution algorithm the final step is to evaluate the following equation:

$$S(\psi|O) = \bigcup_{p_i \in P'} (\bigcap x \mid x \in p_i) \tag{1}$$

where $S(\psi|O)$ is the support for property ψ being true given object O and P' is a set of consistent subsets selected from the class hierarchy.

While the aggregation function shown in equation 1 is formally reasonable it is questionable in practice. The tendency in the union operation is to widen the support region without taking into account the intervals of uncertainty *within* the supports. Take for example the support intervals $[l_1, u_1]$ and $[l_2, u_2]$ which we combine by interval union to yield $[\mathrm{MIN}(l_1, l_2), \mathrm{MAX}(u_1, u_2)]$. If the union

operation gives us $[l_1, u_1]$ then this clearly has not taken into account the specificity of the narrower interval $[l_2, u_2]$. A better approach may be to join these supports in a more considerate way, such as through an interval disjunction. Another alternative is to defuzzify the supports since we commonly require the final support for a property to be expressed as a singleton. In the following alternative aggregation the interval conjunction is implemented as in equation 2 and the interval disjunction is implemented as in equation 3.

$$[l, u] \wedge [n, p] = [l \times n, \quad u \times p] \tag{2}$$

$$[l, u] \vee [n, p] = [l + n - l \times n, \quad u + p - u \times p] \tag{3}$$

Motivation for the use of logical rather than set operations is based on the selective nature of the default reasoning algorithm. The algorithm selects a set of consistent subsets which form a *theory* for the applicability of property ψ. Determining the applicability of ψ can therefore be thought of as the resolution of this theory into a single conclusion. Logical operations are a natural alternative for resolving this theory. Our new aggregation operator is shown in equation 4.

$$S(\psi|O) = \bigvee_{p_i \in P'} (\bigwedge x \mid x \in p_i) \tag{4}$$

Figures 22 and 23 show the results of classifying the figure eight and doughnut shape using this alternative aggregation operator. Notice the improvement in both the classification accuracy and also in the general shape being modelled.

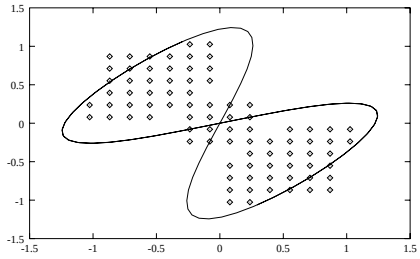


Fig. 22. Improved fused figure eight: 85%

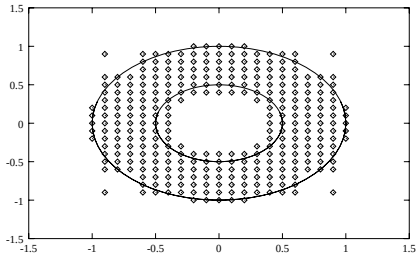


Fig. 23. improved fused doughnut: 92.1%

4.6 A Problem of Priorities

We have shown in the previous section how combining expert and learnt knowledge can yield a more accurate model, while at the same time retaining some degree of transparency. Now let us look at a case where such fusion is less successful.

Previously we showed how a model of the lung region database can be learnt using object oriented modelling with words. Given the requirement for very high success rates in medical applications we desire that the accuracy of 87.4% be improved upon. By examining the misclassified points in figure 13 we could conclude that by adding some expert knowledge defining the valid lung volume we could classify quite a number of these points correctly. To this effect we define the uncertain class hierarchy in figure 24.

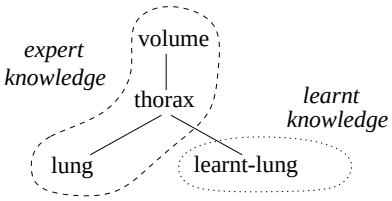


Fig. 24. Lung hierarchy

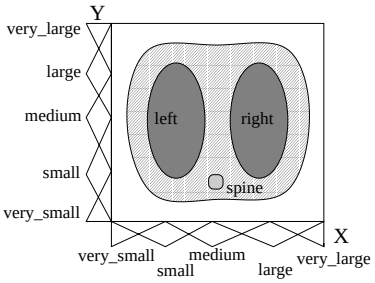


Fig. 25. Simplified model of 2d lung image

We now define the expert knowledge in classes volume, thorax and lung in terms of five fuzzy words defined on each of the x and y domains shown in figure 25. For the volume class we simple define both legal and illegal to be true for any point in the unit cube. For the thorax class we define the legal points as those in the cube further restricted by the condition $x = y = \{very\_small : 0,\ small : 0.5,\ medium : 1,\ large : 0.5,\ very\_large : 0\}$. For the lung class the rules for legal and illegal are further restricted to remove the points between the left and right lung regions. These restrictions are shown graphically in figure 26.

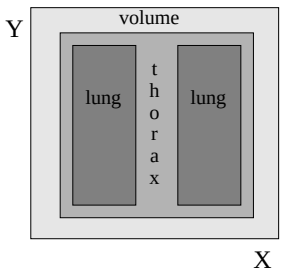


Fig. 26. Restrictions on expert knowledge through the hierarchy

Unfortunately the expert classes perform badly (68%, the same as the majority class) when called upon to classify test points. This is because the specifi-

cation of expert knowledge in the domain x and y is not enough to fully capture the shape of the volume. Consequently, when we perform the information fusion with this class and the learnt model from the object oriented data browser we get a result of just 68.5%.

The reason for the poor fusion result stems from the equal priority that the default reasoning algorithm puts on both the expert and the learnt models. Clearly in this instance the expert knowledge, though at first seeming reasonable, should have a much lower weighting in the fusion. At present no mechanism is included in this approach for weighting knowledge from different sources. This is an important next step in the development of this research.

5 A Behaviour Based Approach to Uncertain Belief Updating

In this final section we briefly describe recent work on modelling human behaviours in the context of user modelling. Since we are modelling humans, this is an area where humanist computing is a natural approach to adopt.

The problem of user recognition centres on the temporal aspect of user behaviour. We have some set of known user prototypes $\{U_1, \ldots, U_n\}$, the behaviours of which we know and to which we provide a corresponding set of services. An unknown user u at time t behaves in the fashion b_t, where behaviour is commonly the outcome of some crisp or fuzzy choice, such as whether or not to buy expensive wine. We wish to determine the similarity of u to each of $\{U_1, \ldots, U_n\}$ in order that we provide the appropriate service to u at time t. We must repeat this process as t increases.

In an object oriented environment we construct a hierarchy of n user classes, $\{c_1, \ldots, c_n\}$, and we try to determine the support $S_t(u \in c_m)$ for user u belonging to user class c_m at time t. This support is some function f of the current behaviour b_t and the history of behaviours $\{b_1, \ldots, b_{t-1}\}$. This is shown more generally in equation 5.

$$S_t(u \in c_m) = f(\{b_1, \ldots, b_t\}) \tag{5}$$

$$S_t(u \in c_m) = g(S_{t-1}(u \in c_m), b_t) \tag{6}$$

We can solve this problem at time t if we have the whole behaviour series up to t. Unfortunately at time $t+1$ we will have to do the whole calculation again. Where t is very large the storage of the whole behaviour series and the cost of the support calculation may be too expensive. An alternative approach is to view the support $S(u \in c_m)$ as some belief in the statement *"user u belongs to class c_m"* and this belief is updated whenever a new behaviour is encountered. This belief updating approach is more economical in space since the whole behaviour series no longer needs to be stored. In computation this approach is more efficient since we now must calculate the function g of just the previous support $S_{t-1}(u \in c_m)$ and the latest behaviour b_t. This belief updating approach is shown more generally in equation 6.

In this paper we examine the case where belief is used to represent the membership of a user object in one or more prototype user classes. As implemented in our uncertain hierarchy, memberships are represented by a support interval $[n, p]$ which is a sub interval of $[0, 1]$.

We now examine two approaches to uncertain interval belief updating. The first method defines a simple moving average which takes new evidence to be *absolute* and updates belief to a degree defined entirely by the new evidence. The second method is more interesting in the light of humanist computing since it is inspired by studied of real human behaviour. This second method, Einhorn and Hogarth's anchor and adjustment belief revision, assesses new evidence in the context of our current beliefs and updates our beliefs *relatively.*

5.1 A Moving Average Update Function

The generalised moving average update function [20] calculates the current support S_{n+1} for a hypothesis given the previous support S_n and new evidence $x_n + 1$ as shown in equation 7.

$$S_{n+1} = \frac{n^{\lambda} S_n + n^{1-\lambda} s(x_{n+1})}{n^{\lambda} + n^{1-\lambda}} \tag{7}$$

Where λ would typically lie in the interval $[0, 1]$. This function is applied to the upper and lower bounds of the belief support interval $[l, u]$ independently. The averaging nature of this function ensures that the validity of the support interval is maintained (i.e. $l \leq u$). If $\lambda = 1$ we have a simple moving average expression where current belief is n times as important as new evidence. If $\lambda = 0$ we have an expression that weights new evidence n times as important as current belief. This flexibility may be important in cases where we know that users change their style of behaviour and must therefore be re-classified quickly. For example, a user p is a student for a number of years and therefore cannot afford expensive food or wine. Upon graduation p gets a well paid job and starts to spend their earnings on expensive groceries. From their behaviour we should detect that p has moved from a class of low income to a class of high income. If $\lambda \approx 1$ and n is large this averaging function will react very slowly to such a change.

Notice that we have some freedom in the λ parameter to vary the function's updating from a *recency* bias (the greater influence of recent evidence) to a *primacy* bias (the greater influence of initial evidence). This flexibility is an important characteristic to be taken into account when considering the suitability of this approach, and the value of λ, for a particular application.

5.2 Anchor and Adjustment Belief Revision

If we are to classify human users we should look at how humans might perform this classification task. Hogarth and Einhorn have done much work on models of belief updating that bear some relation to human behaviour [10] [11] . They

have suggested that the strength of current belief can have a major effect on how new evidence updates that belief. For example, the stronger the belief a person has in the trustworthiness of a friend, the greater the reduction in this belief when the friend commits an act of dishonesty. This is a kind of 'knee jerk' or 'shock' reaction. The typical patterns of behaviour is shown in figure 27. Here the negative evidence e^- has two differing effects depending on how large the belief was before e^- was presented. Likewise there are two differing effects of the equal and opposite positive evidence e^+.

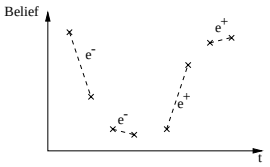


Fig. 27. Order effects in anchor and adjustment

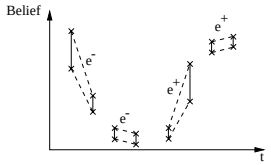


Fig. 28. Order effects in interval anchor and adjustment

Hogarth and Einhorn's anchor and adjustment belief revision model [11] updates a belief given new evidence through two processes. Equation 8 shows how belief S_k is updated given new *negative* evidence. Equation 9 shows how the same belief S_k is updated given new *positive* evidence.

$$S_k = S_{k-1} + \alpha S_{k-1}[s(x_k) - R] \qquad \text{for } s(x_k) \leq R \tag{8}$$

$$S_k = S_{k-1} + \beta(1 - S_{k-1})[s(x_k) - R] \qquad \text{for } s(x_k) > R \tag{9}$$

R is a reference point for determining if the evidence $s(x_k)$ is positive or negative, and typically $R = 0$ or $R = S_{k-1}$. α and β are constants which define how sensitive the model is to negative or positive evidence respectively.

Since evidence in our uncertain environment can be presented as a support interval we must consider the implications of an interval representation of evidence on the anchor and adjustment model. For a piece of evidence e with the associated support interval $[l, u]$ we can view l as the positive evidence associated with e and $1-u$ as the negative evidence associated with e. The general principle is that, given a current belief $[n, p]$ and a piece of evidence with support $[l, u]$, belief increases by an amount proportional to $1 - p$ and belief decreases by an amount proportional to n.

We can apply equations 8 and 9 to the support interval to yield equations 10 to 13 where S^- and S^+ are the lower bound and the upper bound of belief respectively.

$$S_k^- = S_{k-1}^- + \alpha S_{k-1}^-[s^-(x_k) - R^-] \qquad \text{for } s^-(x_k) \leq R^- \tag{10}$$

$$S_k^- = S_{k-1}^- + \beta(1 - S_{k-1}^+)[s^-(x_k) - R^-] \qquad \text{for } s^-(x_k) > R^- \tag{11}$$

$$S_k^+ = S_{k-1}^+ + \alpha S_{k-1}^-[s^+(x_k) - R^+] \qquad \text{for } s^+(x_k) \leq R^+ \tag{12}$$

$$S_k^+ = S_{k-1}^+ + \beta(1 - S_{k-1}^+)[s^+(x_k) - R^+] \qquad \text{for } s^+(x_k) > R^+ \tag{13}$$

R^- is a reference point for determining if the lower bound of the presented evidence is positive or negative with respect to the lower bound of belief and R^+ is the corresponding reference point for the upper bound of belief, and where $R^- = S_{k-1}^-$ and $R^+ = S_{k-1}^+$ and $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$. The precise effects of negative evidence e^- and positive evidence e^+ is determined by α and β respectively. The typical pattern of belief updating behaviour for the interval case is shown in figure 28. Notice how the support interval may contract or expand depending in the interval, magnitude and sign of the new evidence and also on the previous belief.

This new interval version of Hogarth and Einhorn's belief updating model has a number of advantages over the previous generalised averaging method. Recency characteristics allow the anchor and adjustment model to reclassify users quickly. The order effects of this model are related to human behaviour and this seems an important consideration where we are recognising human users and where we are working towards humanist computing. In addition, this method allows us to control the effects of positive and negative evidence separately. This last feature may be especially important in medical applications where false negative classifications have far more serious consequences than false positive classifications.

5.3 A User Recognition Example: The Iterated Prisoner's Dilemma

The n player iterated prisoner's dilemma problem [1] [16] is a good testbed for user recognition due to the production of streams of human-like behaviour that are a result of the pair-wise interaction of simulated prisoners. The problem is most easily understood by looking at the non-iterated problem with $n = 2$. Two prisoners are due to be sentenced. They each have the choice to cooperate together or to defect. If the players both cooperate they will both serve 3 years. If they both defect they will both serve 1 year. If they choose to behave differently then the defecting player will serve 0 years but the cooperating player will serve 5 years. The iterated problem simply continues the game after each round.

A wide range of strategies are possible, ranging from trusting behaviour (always cooperate) to defective behaviour (always defect) and including more complex strategies such as conditional cooperation (cooperating unless the opponent's last m behaviours were defect).

5.4 The User Recognition Example in Fril++

If we were to construct a class hierarchy of prisoners in Fril++ it could resemble figure 29. The subclasses of *prisoner* are the classes that define prototypical

prisoners and their behaviours. For each of the prisoner classes we construct at least one instance. A tournament is then initiated between these instances. After each round of the tournament the memberships of each instance in each prototypical class is determined. The supports for these memberships, i.e. $\{(\alpha_T\ \beta_T)$, $(\alpha_D\ \beta_D)$, $(\alpha_R\ \beta_R)$, $(\alpha_P\ \beta_P)\}$, are calculated using the belief updating methods discussed previously. [8] explains how object recognition is performed, and hence how memberships are calculated, in Fril++.

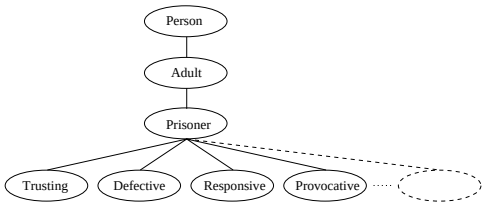


Fig. 29. A class hierarchy for the prisoner's dilemma problem

As an example let us consider the population of ten prisoners $\{p_1, \ldots, p_{10}\}$ where there are two instances of each of the classes {*cooperative*, *defective*, *tit-for-tat*, *random*, *respd*} and whose behaviours are defined in table 3.

Behaviour	**Description**
cooperative	always cooperate with opponent
uncooperative	always defect against opponent
tit-for-tat	cooperate unless last opponent defected
random	equal random chance of defect or cooperate
respd	defect unless the last 6 opponents chose to cooperate

Table 3. Behaviour classes

A population of ten prisoners was created and a game of 75 rounds was initiated. Each round involved picking pairs of prisoners at random from the population until non were left and for each pair recording the behaviours they exhibit (defect, cooperate, etc). From the past history of each player, and using the techniques described earlier ($\alpha = \beta = 0.3,\ \lambda = 1$), they were classified into the five behaviour classes. The winning class was taken as the class in which minimum membership (i.e. the lower bound of the membership interval) was greatest. If the winning class matched the actual class in table 4 then the classification was recorded as a success.

To recreate the situation where user behaviour changes, after 60 rounds the behaviours of all 10 prisoners was changed, as shown in the third column of table 4. After this point the game was continued for 15 rounds. We compared

Individual	Behaviour before 60^{th} round	Behaviour after 60^{th} round
1	random	cooperative
2	random	uncooperative
3	cooperative	tit-for-tat
4	cooperative	respd
5	uncooperative	random
6	uncooperative	respd
7	tit-for-tat	cooperative
8	tit-for-tat	random
9	respd	tit-for-tat
10	respd	uncooperative

Table 4. The prisoner population

classification results using the interval anchor and adjustment belief updating method with the generalised moving average method described above. The whole process was repeated five times and the mean of the results was take.

Interval a-a method	Generalised average method
63.6%	63.3%

Table 5. Classification results before 60^{th} round

Interval a-a method	Generalised average method
57.3%	22.2%

Table 6. Classification results after 60^{th} round

As can be seen from table 5, classification results before the 60^{th} round (the point of behaviour change) are quite similar between the two methods. After the 60^{th} round, however, there is a marked difference in the results, with a large fall in the performance of the generalised average approach. The effect can be seen most clearly in figure 30 where the generalised average approach is slow to react to the change of behaviour of prisoner number 3 from cooperative to tit-for-tat. For clarity figure 30 shows only the lower bound of the belief interval defining the support (current belief) for membership in each class.

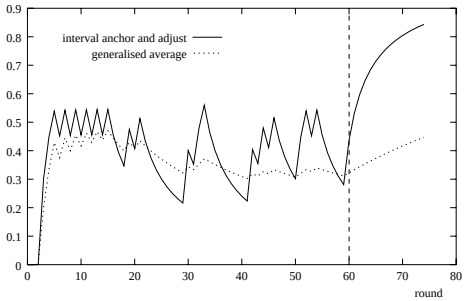


Fig. 30. Membership of prisoner 3 in class tit-for-tat

6 Further Applications of Humanist Computing

6.1 Autonomous Robots in Dangerous Environments

A robot must learn how to move in dangerous environments in order to perform some helpful task - such as searching for humans in fires. This will involve learning rules from sensor input such as audio, visual and motion. (For an example of the low-level fusion of such sensor data see [17]). Unfortunately, when training the robot we cannot teach it using real hazardous situations because, firstly, these situations are difficult to manage and expensive to set up, and secondly, the robot could be damaged before it learns how to behave safely.

We need, therefore, to give the robot some expert knowledge and human-like behaviour concerning dangerous environments and the tasks it must perform. This knowledge may be common-sense in nature such as 'do not walk on soft ground if you have thin legs and are quite heavy' or may be more specialised information such as 'move toward badly injured humans, especially those that are not moving and are in rooms with dense smoke'.

6.2 Recreating Human Behaviour in Robots

In the near future human-like robots will have to operate in the world of the human. Their behaviours will have to be complementary and integrated with ours. On the other hand, we require that intelligent machines perform tasks more efficiently and more rapidly than humans. Unfortunately speed and pleasant behaviours at not always compatible. For example, when two people are about to walk through a door together it is common for the first person to hold the door open for the second. This is a simple courtesy. On the other hand, when the first person is very busy he may just rush through the door on his own and not be courteous. If a robot has been trained with a reinforcement learning algorithm it may learn that the fastest way to complete its tasks is to never hold the door open for anyone else. This will be particularly unhelpful when the robot fails to help someone who is carrying a large load with both hands.

Here the robot needs not only to sense the environment and work towards its immediate goal but also to take into account expert high-level knowledge and human behaviour such as 'unless you are very busy, always hold the door open for someone else'.

7 Conclusions

In this paper we have proposed Humanist Computing as an intuitive extension to modelling with words, perceptions, concepts and behaviours. We have given three concrete examples of humanist computing in application to data modelling, information fusion and user classification. While models constructed using humanist computing may not necessarily perform classification or regression tasks better than alternatives, the advantage of the humanist computing approach is in the natural representations of knowledge and behaviour. Using such natural, human-inspired, representations and behaviours we generate transparent glass-box models (in reality these are more murky-box in nature) that we can inspect and query and in which we can have some intuitive confidence.

In the simplest terms we have motivated humanist computing by the assumption that a model which reasons in the same way as us and perceives the world in the same way as us is likely to make the same decisions as us. Clearly, where humans are not good at making the correct decisions we should not use humanist computing. On the other hand there are a large number of applications where humans work naturally and efficiently with vague and uncertain perceptions and concepts. It is in these environments that humanist computing offers the most promise.

Acknowledgements

I would like to thank James Luckraft and Paul Brassnett for their contributions in the respective problem areas of lung region extraction and information fusion. I would also like to thank John Malone for his work on hand labelling the large number of extracted lung regions generated by James' extraction program.

References

1. Axelrod R.: The evolution of cooperation New York: Basic Books, (1985)
2. Baldwin J.F.: Fuzzy Rule Automation from Data using Mass Assignment Theory. Proceedings of FUZZ-IEEE/IFES 1995, no. 4, pp. 674-679, (1995)
3. Baldwin J.F., Cao T.H., Martin T.P., Rossiter J.M.: Towards Soft Computing Object Oriented Logic Programming. Proc. of the Ninth IEEE Int. Conf. on Fuzzy System, FUZZ-IEEE 2000, 768–773, (2000).
4. Baldwin J.F., Lawry J., Martin T.P.: Efficient Algorithms for Semantic Unification. Proceedings of IPMU 1996, (1996)
5. Brunelli R., Poggio T.: Caricatural Effects in Automated Face Perception. Biological Cybernetics, Vol. 69, pp. 235-241, (1993).

6. Cao T.H.: Fuzzy Conceptual Graphs: A Language for Computational Intelligence Approaching Human Expression and Reasoning. In Sincak, P. et al. (Eds): The State of the Art in Computational Intelligence, Advances in Soft Computing, Physica-Verlag, pp. 114-120, (2000).
7. Cao T.H.: Uncertain Inheritance and Reasoning as Probabilistic Default Reasoning. Int. J. of Intelligent Systems, **16**, pp. 781–803, (2001).
8. Cao T.H., Rossiter J.M., Martin T.P., Baldwin J.F.: Inheritance and recognition in uncertain and fuzzy object oriented models Proceedings of NAFIPS 2001, (2001).
9. Cao T.H., Rossiter J.M., Martin T.P., Baldwin J.F.: Inheritance and Recognition in Uncertain and Fuzzy Object Oriented Models. Proc. of the 1st Int. Joint Conf. of the Int. Fuzzy Systems Ass. and the North American Fuzzy Information Processing Soc., IFSA/NAFIPS, (2001)
10. Einhorn H.J., Hogarth R.M.: Ambiguity and uncertainty in probabilistic inference. Psychological Review, Vol 93, pp. 433-461, (1985).
11. Einhorn H.J., Hogarth R.M.: Order effects in belief updating: the belief-adjustment model. Cognitive Psychology, Vol 24, pp. 1-55, (1992).
12. Goldman A.I.: Epistemology and cognition. Harvard University Press, pp 344-358, (1986).
13. Horwood A., Hogan S.J., Goddard P.R., Rossiter J.M.: Computer-assisted Diagnosis of CT Pulmonary Images. Applied Nonlinear Mathematics Technical Report #200109, University of Bristol, (2001).
14. Horwood A., Hogan S.J., Goddard P.R., Rossiter J.M.: Image Normalization, a Basic Requirement for Computer-based Automatic Diagnostic Applications. ANM Technical Report #200110, University of Bristol, (2001).
15. Jeffrey R.C.: The logic of decision. McGraw-Hill, (1965).
16. Martin T.P.: Incremental Learning of User Models - an Experimental Testbed. Proceedings of IPMU2000, (2000).
17. Mukai T., Ohnishi N.: Sensor Fusion of a CCD Camera and an Acceleration-Gyro Sensor for the Recovery of Three-Dimensional Shape and Scale. Proc. of the 2nd Int. Conf. on Information Fusion (FUSION'99), pp.221-228, (1999).
18. Pearl J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. San Francisco, CA: Morgan Kaufmann (1988).
19. Rossiter J.M., Cao T.H., Martin T.P., Baldwin J.F.: User Recognition in Uncertain Object Oriented User Modelling Proceedings of the Tenth IEEE International Conference on Fuzzy System, FUZZ-IEEE 2001, (2001)
20. Rossiter J.M., Cao T.H., Martin T.P., Baldwin J.F.: object oriented modelling with words Proceedings of IEEE Workshop on Modelling with Words, FUZZ-IEEE 2001, (2001)..
21. Rossiter J.M., Cao T.H., Martin T.P., Baldwin J.F.: object oriented modelling with words. Proc. of the Tenth IEEE Int. Conf. on Fuzzy System, FUZZ-IEEE 2001, (2001).
22. Rossiter J.M.: Humanist Computing for Knowledge Discovery from Ordered Datasets. University of Bristol, (2000).
23. Shore J.E., Johnson R.W.: Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy. IEEE Trans. Inform. Theory, IT-26(1), 26-37, (1980).
24. Zadeh L.A.: Fuzzy Logic = Computing With Words. IEEE Transactions on Fuzzy Systems, Vol. 4, No. 2, (1996).
25. Zadeh L.A.: From Computing With Numbers to Computing with Words - From Manipulation of Measurement to Manipulation of perceptions. IEEE Transactions on Circuits and Systems, Vol. 45, No. 1, (1999).