Affective State Detection With Dynamic Bayesian Networks

A Literature Survey

ing. M.A. de Jongh

8 December 2005

Man-Machine Interaction Group
Media and Knowledge Engineering
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology

Abstract

This literature survey reviews 17 papers that are related to the subject "Affective State Detection with Dynamic Bayesian Networks". The necessary theoretical background on affective computing and dynamic Bayesian networks is presented as an introduction to the papers. Papers have been reviewed from 8 topic areas: Active Affective State Detection, User Modeling, Education, Mental State Detection, Empathic and Emotional Agents, Cognitive Workload Detection, Facial Recognition and Natural Communication. Most promising is a paper on mental state detection. Generally the field has a very broad applicability although it hasn't matured very much in the ten years of its existence. Problems regarding computational complexity and lack of empirical data will have to be solved before real-world application will be possible. When designing an affective state detection system one has to make it multimodal, keep the model's complexity under control, have a lot of data for training and testing and to choose sensors that are as unobtrusive as possible.

Contents

1 Introduction

"What do women want?" This question haunts Mel Gibson's character in the movie "What Women Want". Nick Marshall (Gibson) works at a large advertising firm and just lost a promotion to a female rival, Darcy McGuire (Helen Hunt). Darcy challenges the senior staff to examine some common female products and to present marketing ideas for these products to gain insight in the way women thinks. Nick's attempt to understand the female psyche ends with him lying unconscious on the floor, after receiving an electric shock, resulting from multiple accidents with the female products. The next morning after he awakens from his freakish accident he suddenly is able to hear what women think. Although at first he finds this new ability horribly inconvenient, he later realizes he can exploit this ability to get what he wants. As the movie moves along Nick learns things about women he would have never known and although he eventually loses his ability, he is a better man than before. And of course, he gets the girl; this is Hollywood after all.

We humans are very complex, social creatures. There are many ways we can communicate with each other. Speech, facial expressions, gestures, body language, pheromones, the possibilities for expressing ourselves are almost endless. A picture can say more than a thousand words, but only one small subtle gesture by a friend may say much more. And still with our large arsenal of communication "sensors", we may be completely oblivious of someone's intentions. Perhaps because of ignorance or inexperience, but sometimes the detection system is defective. People diagnosed with autism or Asperger's Syndrome for example lack the ability to recognize subtle nonverbal communication. This makes life a lot harder for them, because they do not see the things that are completely obvious for the rest of us.

Now consider a computer's viewpoint: it doesn't have one. Computers are heartless and mindless machines. They are completely ignorant of the user's emotions, desires and intentions. You can scream and curse all you want, it will not care. And if the same situation repeats itself, its behavior will be identical. This of course has an advantage: computers can do our boring or difficult work without whining 24 hours a day, 7 days a week. The only problem is that in the best case scenario computers only do exactly what we tell them to do. And without a good user-friendly interface, getting the computer to do exactly what you want may still be quite difficult.

Until the mid 90's Human Computer Interaction (HCI) research mostly focused on designing better interfaces to prevent stressful situations and making HCI more natural [1][2]. This research has been effective, and has led to better programs, but still the human factor was not completely exploited.

In 1995 Rosalind Picard proposed a new way to tackle the problem: Affective Computing [1]. She believes that emotions play an important role in decision making and human intelligence. Because of this importance she believes that it is obvious that computers should also be able to work with emotions. Human intelligence does not function without it, so artificial intelligence agents should also have the same capabilities. How can we call agents intelligent if they lack the driving force behind our decision making ability?

Integrating affective computing in agents leads to programs that adapt their communication with the user depending on the user's state of mind. We all know that an angry person should not be approached in the same way as a happy person. The next time Microsoft Word fails to do exactly what you want and that annoying paperclip shows itself to "help"; it might start with an apology to break the ice. This is just one of many applications where insight in the affective state of the user could lead to better results. Other possibilities are educational tutor systems, driver vigilance detection systems or natural communication systems. So many systems can benefit from the extra acquired information to smoothen communication. Enhancing the computer's social abilities makes human computer interaction more natural and more "real". It will lead to better, smarter and easier user interfaces.

In conclusion granting computers the ability to recognize or show emotions brings us closer to a more natural way of interacting with computers. Maybe in time computers can assist us with answering that philosophical question: "What do women want?"

1.1 Survey overview

The subject for the survey is "Affective state detection with Dynamic Bayesian Networks". Due to the small number of papers available that exactly fit the subject description the subject has been broadened a bit. The new subject can be described as: "Human state detection with Bayesian Networks". By dropping the requirement that the Bayesian Network should be dynamic and by considering not only the affective state of a human but also other mental states, the number of available suitable papers became sufficient for writing a literature survey.

The survey starts with a review of the theoretical background of the subject. The fields Affective Computing and Bayesian Networks will be covered as an introduction to the topic. After the review of the background the different papers, subdivided by subject, will be discussed. The survey ends with a conclusion that gives an overview of the field; it shows the state of current research and the direction it is heading.

2 Theoretical Background

To give some insight in the theoretical background of the subject the two main theories are covered. These are Affective computing and Bayesian Networks. Affective computing deals with integrating human emotions into software and Bayesian Networks are used for probabilistic reasoning.

2.1 Affective Computing

The field of affective computing was proposed and pioneered by Rosalind Picard from the MIT Media Laboratory. Her definition of affective computing is: "computing that relates to, arises from, or deliberately influences emotions." Her argument for putting emotions or the ability to recognize emotions into machines is that neurological studies have indicated that emotions play an important role in our decision making process. Our "gut feelings" influence our decisions. Fear helps us to survive and to avoid dangerous situations. When we succeed, a feeling of pride might encourage us to keep on going and push ourselves even harder to reach even greater goals. Putting emotions into machines makes them more human and should improve human-computer communication. Also exploiting emotions could lead to a more human decision making process. Modeling fear for instance, could lead to robots that are better at self preservation. The extra information could be used to choose a safer route through an environment and not just the first available one.

Picard focuses her research mostly on the detection of the affective state of a person. Other groups do some work in software agents capable of portraying some emotion, but only as a reaction on the user's current affective state. Using an emotional model as part of a decision model for an intelligent agent has not been researched much, mostly because there is still quite some ethical debate about giving emotions to machines [1]. For example, try to imagine an automatic pilot for an airplane that has an explosive temper. Of course this is an extreme example but emotions can make decisions less rational and adding emotions will affect the behavior of a machine. Most of the time this will be desirable, but something could go wrong.

For detecting the affective state of a human, many sources of information can be used. Gestures, facial expression, pupil size, voice analysis, heart rate, blood pressure, conductivity of the skin and many more can be used as input data for the emotion model. This data then has to be interpreted to infer the affective state. Because emotions can vary in intensity and can be expressed in many ways affective state recognition generally is modeled as a pattern recognition or fuzzy classification problem.

Most research groups use a probabilistic approach like (Dynamic) Bayesian Networks (DBNs) or Hidden Markov Models (HMMs) for the pattern recognition process. The advantage of using these kinds of models is that emotions are a "state of mind". It is not possible to read them directly, so indirect evidence has to be used for inferring this hidden state. Both probabilistic methods have a natural ability for dealing with hidden states.

Neural Networks can also be used for Affective State Detection, but a drawback is that there is no way of knowing how the knowledge is distributed over the nodes of the neural network. This makes the neural network a black box and makes it is impossible to explain how the network classified an input pattern. DBNs and HMMs are not black boxes and are much better at explaining how a classification was reached, making them more useful than neural networks.

Once the affective state has been classified, this information can be used for different applications. Some applications are:

- Entertainment:

  Using affective information in games could make the game experience more intense. In First Person Shooter (FPS) games, it's very common that the player gets scared by the game. To induce this effect, game designers normally use a combination of music, background noise and lighting. The game DOOM3 for example, uses very dark scenes and has opponents jumping out of nowhere. A game using affective information could monitor the fright level and when this level reaches a certain threshold, a scripted event could be triggered. An example event could be that the lights in the level would fail, followed by an attack from several opponents.

- Expressive Communication:

  Nowadays we have many tools for communication to our disposal. Mobile phones, e-mail, instant messaging are a few of the current popular means of communication. When using speech and/or video communicating emotions is quite simple, but this requires a lot of bandwidth. Text messaging is very popular, either by mobile phone (SMS) or internet (MSN). A problem is that only text is sent and it is harder to see in what emotional state someone is. The use of emoticons (i.e. :-) as a happy face) is a simple way of adding emotional content to a text message. Using affective computing, the emotional state can be detected automatically and added to the text message. This could be done by adding emoticons or showing a standard picture at the receiver side. The addition of affective state detection could make the text messages more expressive and natural without using too much bandwidth.

- Educational Tutoring systems:

  The use of computers has slowly integrated in the curriculum of all the different educational institutes. From pre-school to universities, computers can be used to enhance the learning experience. Educational software could benefit from affective computing, being able to sense when a student is frustrated is very important. When a student is frustrated he is more likely to learn less [3]. When the program senses frustration it could suggest a less difficult exercise, give a hint or give some extra explanation on the subject.

- Affective Environment

  Picard writes about affective environments [1]. Buildings, rooms or software that adapts itself to the user's emotional state. Changing the look and feel to make him more comfortable. For rooms or buildings changeable parameters could be the lighting, background sounds, décor, or temperature. For software, the interface could be adapted to fit the user's affective state. Also the system could be used in the opposite way: choosing the parameters in such a way to promote a certain affective state. As an example, Picard mentions a digital disc jockey able to select music for creating a certain atmosphere for a party.

2.2 Bayesian Networks

There are a many different ways to model affective state recognition; the literature survey assignment requires that a (Dynamic) Bayesian Network is used. The necessary theory for understanding the general working of (Dynamic) Bayesian Networks will be treated in this paragraph. First an introduction to probability theory and probabilistic reasoning will be given, followed by an explanation of static and dynamic Bayesian networks.

2.2.1 Basic Probability Theory

A lot of games use dice. When dice are thrown, it is not known which numbers will end up top until the dice stop rolling. Is it impossible to calculate which numbers end up top? No, with a model that deals with every little detail of the world, models every necessary action, knows the exact mass and dimensions of the dice, the exact conditions of the surroundings, etcetera it is possible to exactly calculate which numbers will end up top. But this will result an equation with thousands, maybe millions of parameters, just for throwing dice. It's practically impossible and simply pointless to make such models.

There is another way of looking at the problem. A die has six sides; all are uniquely numbered from 1 to 6. When a die is thrown, only one number ends op top. There are six possible outcomes for throwing a die. These six outcomes form the so called sample space. In general, a sample space is the set containing all possible outcomes of an experiment. Using probability theory, probabilities can be assigned to each of the outcomes in the sample space. These probabilities give an estimate of the likelihood of the occurrence of an outcome. In the case of a fair die all the outcomes are equally probable. When a fair die is thrown a few hundred times, all the different outcomes then should occur roughly the same amount of times.

If the probability of throwing higher than 4 is required, the probabilities of throwing 5 and throwing 6 are added to give this probability. These probabilities now form a subset of the total sample space. In probability theory this is called an event.
All normal set operations can be performed on events. Intersections, unions, complements, are commonly used operations in probability theory.

Formally a probability function P is defined on a sample space Ω to assign a numerical value P(A) to an event A in Ω the range [0, 1] such that:

1. The probability of the whole sample space, P(Ω), is equal to 1.
2. If events A and B are disjoint the probability of the union of the events equals the sum of the probabilities of the events:
   $P(A \cup B) = P(A) + P(B)$

To formalize the previous statement even further, the axioms of probability are stated:

1. All probabilities are between 0 and 1:
   $0 \leq P(A) \leq 1$
2. True events have probability 1, false events have probability 0:
   P(true) = 1, P(false) = 0
3. The probability of a disjunction is given by:
   $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

These axioms are known as Kolmogorov's Axioms, named after the Russian mathematician, who showed how to build the rest of probability theory from these axioms [4].

Conditional Probabilities

Knowledge of the occurrence of event A might cause a reassessment of the likelihood of the occurrence of event B. For instance, imagine a guessing game. The objective of the game is to guess the month someone is thinking about. If it is assumed that this person does not have a certain preference for some months, the probability for correctly guessing a month can be set at 1/12. This means that all the months have equal probability. When this person says that he is thinking about a winter month, the probabilities will have to be reassessed. Instead of twelve possible months, now there are only three possible months left. The new probability for guessing the right month is obviously 1/3, since there is still indifference between the different months.

This kind of probability is known in probability theory as a conditional or posterior probability and the notation used for this kind of probability is P(A|B), or in the example P(M|S). This is read as: "the probability of the occurrence of M(onth) given that S(eason) has occurred". The conditional probability of an event A given another event B can be calculated using the following equation:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Equation 1: Conditional Probability

This equation holds when P(B) > 0.

Sometimes it is desirable to compute P(B|A), but most of the time it is hard to find or calculate P(A∩B). Using Bayes' Rule this problem can be solved:

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Equation 2: Bayes' Rule

A more general form exists, making use of the law of total probability:

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_j P(A|B_j)P(B_j)}$$

Equation 3: General form of Bayes' Rule

The law of total probability simple states that P(A) can be found by adding all the probabilities of conjunctions of A with all other events. Probabilities of conjunctions can be written as a product of a conditional and a prior probability, this can be derived from Equation 1.

Random Variables

The combination of a sample space and a probability function is sufficient to probabilistically describe an experiment. But sometimes this combination gives too much information and most of it is not interesting enough or not necessary. To focus on specific features of an experiment random variables are used.

A definition of a random variable based on Dekking et al [5]:

> Let Ω be a sample space. A random variable is a function X : Ω -> $\Re$ which transforms a sample space Ω into another sample space Ω' whose events are more directly related to the features of the experiment which are to be studied.

An example of a random variable is when playing a game with two dice, Monopoly for instance. When the dice are thrown the number of steps on the board is equal to the sum of the dice. The sample space Ω for this experiment and the random variable S can be defined as following [5]:

$$\begin{aligned}
\Omega &= \{(\omega_1, \omega_2) : \omega_1, \omega_2 \in \{1,2,3,4,5,6\}\} \\
&= \{(1,1),(1,2),\ldots,(1,6),(2,1),\ldots,(6,5),(6,6)\} \\
S(\omega_1, \omega_2) &= \omega_1 + \omega_2
\end{aligned}$$

An event that the variable S attains the value k is denoted by {S = k}.
Possible values of S range from 2 to 12. For a complete probabilistic definition of the experiment probabilities have to be assigned to every possible event of the new sample space, or formally: the probability distribution of S has to be determined.

In the case of S, a discrete random variable, a probability mass function has to be defined. This function assigns a probability to every possible event of the variable S. For a continuous random variable a probability density function is used to assign values to continuous events in a similar manner.

For the random variable S, the probability of an event depends on the frequency of occurrence of the event. The event {S = 2} gets the probability 1/36, because there is only one possible way of throwing 2 (1,1). The event {S = 7} on the other hand gets the probability 1/6, because there are 6 possibilities to throw 7 (6/36).

Continuous and discrete random variables differ in the way the probability distributions are assigned. Continuous variables use densities, areas and integrals to calculate probabilities, discrete variables use discrete events and summations. More similar is calculating the probability F(a) = P(X≤ a). The function F(a) is known as the distribution function or cumulative distribution function. In the discrete case F(a) is calculated by summing the probability mass function, in the continuous case the function is calculated by integration the probability density function.

Definition by Dekking et al [5]:

> The distribution function F of a random variable X is the function
> F: $\Re$ -> [0,1], defined by:
>
> F(a) = P(X≤a) for -∞ < a < ∞.

Joint Probability Distributions

If an experiment has multiple random variables, it is possible that these different variables have an influence on each other. How the variables influence each other can be determined by assigning probabilities to all possible combinations of the variables. By doing so joint probability functions are created. These functions behave just like their discrete and continuous single variable counterparts discussed above.

It is possible to reduce the amount of variables in a joint probability distribution. This process is called marginalization. Marginalization can be performed by using the probability distribution function, the probability mass function or the probability density function. When using the probability distribution function the following procedure is used [5]:

$$F_X(a) = P(X \le a) = F(a,+\infty) = \lim_{b\to\infty} F(a,b)$$
$$F_Y(b) = P(Y \le b) = F(+\infty,b) = \lim_{a\to\infty} F(a,b)$$

Equation 4: Marginalization of the joint probability distribution function

When marginalizing using either the probability mass or density function the procedure is as following:

$$p(a) = \sum_{i=1}^{\infty} p(a, b_i)$$

$$p(b) = \sum_{i=1}^{\infty} p(a_i, b)$$

Equation 5: Marginalization of the probability mass function

$$f_X(x) = \int_{-\infty}^{\infty} f(x, y)dy$$

$$f_Y(y) = \int_{-\infty}^{\infty} f(x, y)dx$$

Equation 6: Marginalization of the probability density function

In the case of more than 2 random variables the process of marginalization can easily be extended by adding the necessary limit conditions, summations or integrals, regarding the variables that have to be marginalized.

Independence

Although multiple random variables can have an effect on each other this does not have to be the case. When two random variables do not have an effect on each other they are called independent of each other.
For the joint probability distribution this has the effect that an entry of the distribution is the product of the marginal distributions of the two independent variables. Dekking et al [5] define the independence of random variables in the following way:

> The random variables X and Y, with joint distribution function F_{XY}, are independent if
>
> $P(X \leq a, Y \leq b) = P(X \leq a)P(Y \leq b)$,
>
> that is,
>
> $F_{XY}(a, b) = F_X(a)F_Y(b)$,
>
> for all possible values a and b. Random variables which are not independent are called dependent.

One note of caution when working with more than two random variables: although the combination of all variables could make them independent, i.e. entry of joint probability distribution is multiplication of all marginal distribution, smaller subsets of random variables may still be dependent on each other.

Checking independence can be done with the following formulas:

$P(A|B) = P(A)$,
$P(B|A) = P(B)$,
$P(A \cap B) = P(A)P(B)$

If any of the formulas is correct all are valid, they are equivalent statements.
The first two statements, using conditional probability can easily be derived from Equation 2.

The possibility exists that random variables are only independent given certain other random variables. This is called conditional independence.
For conditional independence the same rules apply as for normal independence. For random variables A and B, given variable C:

$P(A|BC) = P(A|C)$,
$P(B|AC) = P(B|C)$,
$P(A \cap B|C) = P(A|C)P(B|C)$

One could even say that "normal" independence is a special case of conditional independence, because "normal" probabilities are actually a special case of conditional probabilities. Consider a probability of an event A in sample space Ω. Normally you would write P(A), but also a conditional probability P(A|Ω) could be written. Since by definition P(Ω) equals 1 and P(A∩Ω) equals P(A), using Equation 2 there can be written:

$$P(A|\Omega) = \frac{P(A \cap \Omega)}{P(\Omega)} = P(A \cap \Omega) = P(A)$$

2.2.2 Probabilistic Reasoning

In the past, reasoning was done almost exclusively with symbolic systems, which are also called expert systems. These systems are rule based and use propositional or first order logic to do inference on facts using rules. The rules represent the expert system's knowledge of the world and the facts are the information the systems receives from sensors or user input. As long as the systems did not have to work in an uncertain environment or handle incomplete data or sensor failure, they performed quite well. But when unexpected situations started to arise, the performance of most expert systems dropped significantly.

Since the world is full of uncertainties and unexpected situations, it is imperative that when designing an expert system it should be capable of handling these kinds of situations.

A solution was found for this problem in the form of certainty factors. These factors were added to rules and facts to give a sort of degree of belief. Although mathematically unsound, it worked well. A famous example is the MYCIN system, which diagnosed certain types of blood diseases better than junior doctors.

Using Probability theory

Another way of dealing with uncertainty is the use of probability theory.
The inputs of a probabilistic inference system are random variables that function as evidence variables. The conclusions which are to be inferred are implemented as random variables named hypothesis or query variables.
Using expert knowledge or a large amount of statistical data the joint probability distribution is derived and can be used to answer any query.
Probabilistic inference generally means "the computation from observed evidence of posterior probabilities for query propositions" [4]. In practice this means that a conditional probability or probability distribution is computed for a certain hypothesis set, containing one or more hypothesis variables with their values either set or unset, given a set of evidence variables (also with set or unset values). Also a common practice is to calculate the a priori distribution of one of the variables. This is done by marginalizing all but the desired variable out of the joint distribution. The procedure can be done by using Equation 5 or the other forms, but also a conditional variation exists, it is called the conditioning rule[1]:

$$\mathbf{P}(\mathbf{Y}) = \sum_{\mathbf{z}} \mathbf{P}(\mathbf{Y}|\mathbf{z})P(\mathbf{z})$$

Equation 7: Conditioning rule

When computing conditional probabilities Bayes' rule is normally used, but if this rule is examined more closely one can see that the denominator actually only works as a normalization constant. For instance when computing **P**(**A**|b) of the binary variables A and B, i.e. the conditional distribution of A given that variable B is true, the denominator in Bayes' rule is P(b) in both the cases that the variable a is true or false. Since the following holds:

$$\begin{aligned}
&P(a|b) + P(\neg a|b) = 1 \\
&\frac{P(a \cap b)}{P(b)} + \frac{P(\neg a \cap b)}{P(b)} = 1 \\
&\frac{1}{P(b)}\left[P(a \cap b) + P(\neg a \cap b)\right] = 1 \\
&\alpha\left[P(a \cap b) + P(\neg a \cap b)\right] = 1 \\
&\alpha = \frac{1}{P(a \cap b) + P(\neg a \cap b)}
\end{aligned}$$

P(b) does not have to be computed, simple computing the joint probability distribution **P**(**A**,b) and normalizing the probabilities so they sum up to one is enough to calculate **P**(**A**|b).

[1] The layout of the probabilities is used from [4], bold probabilities or variables show that a whole probability distribution is being calculated or every possible value of a variable is used in the calculation and not just one probability or one specific value of a variable.

A more general inference procedure can be constructed using this insight [4]:

"Let X be the query variable, let **E** be the set of evidence variables, let **e** be the observed values for them, and let **Y** be the remaining unobserved variables, the query **P**(X|**e**) can then be evaluated as

$$\mathbf{P}(X|\mathbf{e}) = \alpha\mathbf{P}(X,\mathbf{e}) = \alpha\sum_{\mathbf{y}}\mathbf{P}(X,\mathbf{e},\mathbf{y}),$$

Equation 8: General probabilistic inference procedure

where the summation is over all possible combinations of the unobserved variables **Y**."

The general inference procedure can be summarized as a marginalization over unobserved variables, followed by a normalization of joint probabilities.

Using this procedure probabilistic inference can be done, but practically the procedure is useless. The problem is that using the full joint probability distribution for probabilistic inference is very inefficient when the amount of random variables increase. When using a joint probability distribution consisting out of n Boolean random variables, the distribution has 2^n entries. So when n becomes larger; the size of the joint probability distribution and the time necessary for computing the probabilities start to increase very fast. Using computational complexity theory there can be said that the problem has a time complexity and a space complexity of $O(2^n)$. This means that both the necessary time and space increase exponentially when the amount of random variables increases.

Since in real world problems there are hundreds or thousands of potential variables it is totally impractical to use this kind of probabilistic inference. The joint probability distribution needs an enormous amount of space, the algorithm needs an enormous amount of time and it is simple impossible to get enough data for accurate probabilities for the joint distribution.

To make probabilistic inference in real world domains feasible something has to be done about the size of the joint probability distribution. A popular way to reduce the size of problems is to exploit structures in the problem. This is exactly what Bayesian Networks do.

2.2.3 Bayesian Networks

The solution to reduce the size of the joint probability distribution is to exploit independence assertions. If several evidence variables are independent of the query variable and the other evidence variables, they can be removed from the evidence when the query is being computed. This can greatly reduce the amount of work necessary to compute the query probability. Because of the exponential complexity of the algorithm, for every variable eliminated the necessary computation time is halved. Independence assertions like this one do not occur very frequently in real life. Independence assertions that do occur a lot are conditional independence assertions. As stated earlier, a variable is conditional independent of all other variables given a certain set of variables. The usefulness of conditional independence comes from the application of Bayes' Rule on a joint probability distribution. A joint probability distribution can be broken up into a product of smaller conditional probability distributions:

$$\begin{aligned}&\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}) = \mathbf{P}(\mathbf{A}|\mathbf{B},\mathbf{C},\mathbf{D})\mathbf{P}(\mathbf{B},\mathbf{C},\mathbf{D}) = \\ &\mathbf{P}(\mathbf{A}|\mathbf{B},\mathbf{C},\mathbf{D})\mathbf{P}(\mathbf{B}|\mathbf{C},\mathbf{D})\mathbf{P}(\mathbf{C},\mathbf{D}) = \\ &\mathbf{P}(\mathbf{A}|\mathbf{B},\mathbf{C},\mathbf{D})\mathbf{P}(\mathbf{B}|\mathbf{C},\mathbf{D})\mathbf{P}(\mathbf{C}|\mathbf{D})\mathbf{P}(\mathbf{D})\end{aligned}$$

This is just one of many possible ways of breaking down the joint probability distribution. All decompositions effectively give the same joint distribution. The most effective one however, is the one that exploits conditional independence assertions between variables. For example, if it is assumed that A and B are conditionally independent of the other variables given D, the decomposition becomes:

$$\mathbf{P}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}) = \mathbf{P}(\mathbf{A}|\mathbf{D})\mathbf{P}(\mathbf{B}|\mathbf{D})\mathbf{P}(\mathbf{C}|\mathbf{D})\mathbf{P}(\mathbf{D})$$

In the case of Boolean variables the original joint probability distribution has 2^4 - 1= 15 entries that have to be chosen (the last one has to make sure the distribution sums up to 1). The first conditional decomposition needs to define 8+4+2+1 = 15 entries, the exact same amount as the joint distribution. But the second one, using conditional independence assertions, only needs to define 2+2+2+1 = 7 entries. This set of distributions combined needs to define less than half of the entries the original distribution needed to define, and if variables are added the difference grows exponentially.
Another example of conditional decomposition is called the naïve Bayes model. In this model it is assumed that all evidence variables are conditionally independent of each other given one single cause variable. The joint distribution of this model can be written as:

$$\mathbf{P}(Cause, E_1,...,E_n) = \mathbf{P}(Cause)\prod_i \mathbf{P}(E_i|Cause)$$

Equation 9: Naive Bayes Model

The model is called naïve, because the assumption that all evidence variables are conditionally independent is almost never true. But in practice, naïve Bayes systems work quite well [4].

The most attractive feature is that the space complexity of a naïve Bayes system is only O(n), i.e. when the amount of variables double, so does the amount of necessary space. This, in contrast with using a full joint probability distribution with $O(2^n)$ complexity, where the amount of necessary space is squared when the amount of variables is doubled. The time complexity is also linear in the amount of evidence variables, so inference is also quite fast.

Definition of Bayesian Networks

Naïve Bayes models are mostly used for classification problems, when more complex worlds have to be modeled Bayesian networks can be used.
Russell and Norvig [4] define a Bayesian network as following:

> A Bayesian network is a directed graph in which each node is annotated with quantitative probability information. The full specification is as follows:
>
> 1. A set of random variables makes up the nodes of the network. Variables may be discrete or continuous.
> 2. A set of directed links or arrows connect pairs of nodes. If there is an arrow from node X to Node Y, X is said to be a parent of Y.
> 3. Each node X_i has a conditional probability distribution $P(X_i|Parents(X_i))$ that quantifies the effect of the parents on the node.
> 4. The graph has no directed cycles (and hence is a directed, acyclic graph or DAG).

The graph is a visualization of the conditional independence relationships between the different variables. If two variables are connected with each other by an arrow this means that the two variables are conditionally dependent. More specifically: if there is an arrow from node X to node Y, node Y is conditionally dependent on X.

The other half of a Bayesian network consists out of the conditional probability tables (CPTs) which define the conditional probabilities for each node, given its parents. If a node does not have any parents, a prior probability distribution is defined.

Bayesian networks represent a joint probability distribution in the following way:

$$\mathbf{P}(x_1,\ldots,x_n)=\prod_{i=1}^{n}\mathbf{P}(x_i|parents(X_i))$$

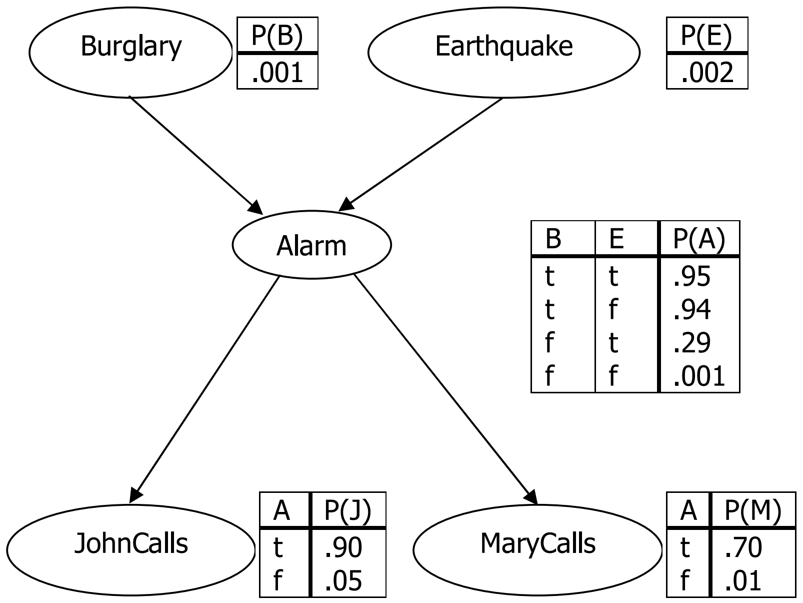
Equation 10: joint probability representation

This representation of the joint probability distribution is correct when using the earlier mentioned method to break up a joint probability distribution into smaller conditional probability distributions, the ordering of the variables fits the decomposition and the variables are conditional independent of their predecessors in this variable ordering given their parents.

From the graph perspective you can write for the conditional independence relations [4]:

1. A node is conditionally independent of its non-descendants, given its parents.
2. A node is conditionally independent of all other nodes in the network, given its parents, children, and children's parents (also known as its Markov Blanket).

An example Bayesian network looks like this [4]:



P(B)
.001

P(E)
.002

B	E	P(A)
t	t	.95
t	f	.94
f	t	.29
f	f	.001

A	P(J)
t	.90
f	.05

A	P(M)
t	.70
f	.01

Figure 1: Example Bayesian Network

Nodes that have an influence on other nodes are connected by arrows and the CPTs show the necessary conditional probabilities. The joint probability distribution belonging to this "world" is represented in the following way:

$$\mathbf{P}(\mathbf{J},\mathbf{M},\mathbf{A},\mathbf{B},\mathbf{E}) = \mathbf{P}(\mathbf{J}|\mathbf{A})\mathbf{P}(\mathbf{M}|\mathbf{A})\mathbf{P}(\mathbf{A}|\mathbf{B},\mathbf{E})\mathbf{P}(\mathbf{B})\mathbf{P}(\mathbf{E})$$

In this case JohnCalls and MaryCalls are conditionally independent of Burglary and Earthquake given Alarm.

To represent the complete domain only 10 conditional probabilities are needed, the full joint distribution would have needed 31. Bayesian networks have a property which makes them much more compact than full joint probability distributions. This property is called local structure.
In locally structured systems not all nodes have an effect on each other. The influence of nodes on other nodes is restricted to a certain amount of neighbors. This property keeps the growth of the model linear and not exponential. There still is an exponential factor in the system, the size of a CPT is exponential in the amount of parents the node has. In the case of Boolean variables the size of the CPT is 2^k with k parents. The total amount of probability entries for a Bayesian network with n Boolean variables is $O(n2^k)$.

This is better than the complexity of a joint probability distribution ($O(2^n)$), but now the complexity is dependent on the specific structure of a problem. In Bayesian networks where nodes have a small amount of parents, the amount of necessary probability entries will be small, but in the situation that every node has an influence on each other the total size of the CPTs could be equal to the full joint probability distribution.

Efficient probability representations

To reduce the amount of conditional probabilities for a model even further, special uncertain relationships can be used to define a whole CPT with only a few probabilities. One example is the noisy-OR relation. It generalizes the logical OR. This model allows for uncertainty that a parent causes the child to be true. The idea is that although the parent has occurred there is a certain probability that the child does not have to occur, i.e. the causal relationship between parent and child may be inhibited. The model makes two assumptions [4]:

1. All the possible causes are listed
2. The inhibition of each parent is independent of inhibition of any other parents.

Using these assumptions the whole CPT can be constructed. For a node A with three parents B, C and D the procedure would be as following:

1. The probabilities $P(\neg A|B,\neg C,\neg D)$, $P(\neg A|\neg B,C,\neg D)$ and $P(\neg A|\neg B,\neg C,D)$ should be acquired or calculated.
2. Using the probabilities from 1 the rest of the CPT can be calculated.

The rest of the probabilities are calculated by multiplying the known probabilities with each other. This means that to define a CPT for a node with k parents that there are only k probabilities necessary. Now a complexity of O(k) can be reached for the size of the CPTs and a complexity of O(nk) for the whole Bayesian network. An example of how the noisy-or model works is explained in [4], the results of this noisy-or calculation are shown in Table 1.

Table 1: CPT calculated with noisy-OR

B	C	D	P(A|BCD)	P(¬A|BCD)
F	F	F	0.0	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	0.02=0.2x0.1
T	F	F	0.4	**0.6**
T	F	T	0.94	0.06=0.6x0.1
T	T	F	0.88	0.12=0.6x0.2
T	T	T	0.988	0.012=0.6x0.2x0.1

Only the bold probabilities are stored in the model, all the other probabilities are calculated by multiplication of the stored probabilities. The exception is when all parent variables are false; this is a result of the first assumption of the noisy-OR model: all possible causes should be listed. Because all the causes are false A cannot occur, hence $P(A|\neg B\neg C\neg D)$ is set to 0.

Exact Inference

With every part of the Bayesian network model defined, only inference in Bayesian networks still has to be explained. There are two kinds of inference procedures: exact and approximate inference. With exact inference every query can be computed without error. With approximate inference only an approximation of the probability of the query can be computed. The problem with exact inference is that for most models it is intractable; it takes an extremely long time to compute a query. This explains the need for approximate inference algorithms, this way probabilities can be computed in a reasonable amount of time.

Using Equation 8 from page 12 and the independence assumptions on page 15, a query for a Bayesian network can be computed exactly. But this algorithm is terribly inefficient for Bayesian networks, according to [4] the computational complexity for inference for a Bayesian network with n Boolean variables is $O(n2^n)$ in the worst case scenario. This is even worse than doing inference with the full joint probability distribution. To improve efficiency, algorithms using techniques from Dynamic Programming calculate intermediate results of the calculation only once and reuse them later. An example of an algorithm that uses Dynamic Programming techniques is the Variable Elimination algorithm.

The Variable Elimination algorithm works by evaluating query expression in right-to-left order. First the necessary summations (to sum variables out) are moved inwards as far as possible. Then for every conditional probability a factor is created. A factor is represented as a matrix or a vector and contains the probabilities needed for multiplication and summation to compute the final answer. Starting at the summation which is moved most inward the different factors are multiplied with each other using a special multiplication

technique called a pointwise product. The result of a pointwise product of two factors is another factor whose variables are the union of the variables of the two factors. The probability entries of the new factor are the product of the probability entries of the two input factors (for a more in-depth explanation see [4]). After the factors have been multiplied, the variable of the summation is summed out, giving a new factor containing the remaining variables. The process is repeated until the last variable is summed out. Now only one factor remains and after normalization of the entries of the factor the answer of the query is found.

Approximate inference

Even though the variable elimination algorithm is quite efficient, when the size of a Bayesian network grows, it still takes an exponential amount of time to calculate the desired query. Exact inference is a difficult problem; it is #P-hard, meaning that is even more difficult to solve than NP-complete problems [4]. To be able to calculate queries in a reasonable amount of time approximate inference algorithms have to be used. Randomized sampling algorithms are used for approximate inference. There a few different types of these algorithms, also called Monte Carlo algorithms, two approaches are direct sampling and Markov chain sampling.

Direct sampling methods are relatively easy; they use a very easy-to-sample distribution to produce samples from a hard-to-sample distribution. In this case the hard-to-sample distribution is the joint probability distribution represented by the Bayesian network. There are some different variations of direct sampling methods, differing in complexity and quality. The simplest one samples from the prior distribution, beginning at the root variables of the network and moving down to the leaves. At every variable it generates a random number and using the prior distribution of the variables it assigns a value to the variable. If the variable has parents, the conditional distribution is conditioned on the values of the parents. After all variables have been assigned a value a sample has been generated. This sample can be assigned a probability by multiplying the prior probabilities of the values of all the variables. This probability can be used as an estimate of query. If the sampling process is repeated a large amount of times more accurate estimates of the real probability of the query can be acquired. To get an estimate of a query, probabilities necessary for the query have to be sampled. For a query with query variable X and evidence variables **e** this means estimating the probability distribution **P**(X,**e**). Since sampling is a random process samples will be generated that will have evidence that is not consistent with the desired query. These samples are not very useful for the approximation of the query and to get a better estimate these samples have to be dealt with.

Rejection Sampling

One way of dealing with this problem is a method called rejection sampling. The prior distribution is sampled N times and every sample which is not consistent with the evidence of the query is discarded. The remaining samples are used to obtain the estimate of the probability distribution **P**(X,**e**). A frequency table is created where the number of times X takes a certain value is counted. This table is normalized to get the estimate of the probability distribution. The problem with rejection sampling is that when there are a lot of evidence variables the amount of samples with consistent evidence will drop exponentially.

The exponential drop is because of the exponential growth of the possibilities of the evidence. This means that there will be only a very small amount of samples with consistent evidence for the calculation of the estimate of the probability distribution of the query. The result is that the rejection sampling method is simply unusable for large, complex problems [4].

A method that gives better results and does not throw away any of the samples is likelihood weighting. It calculates the likelihood of a sample and weights the sample with this probability. Now samples with inconsistent evidence will only have a small influence on the final result. When the amount of evidence variables increases likelihood weighting will also perform less, again because of the exponential growth of evidence possibilities.

Markov Chain Simulation

Totally different from direct sampling algorithms is Markov chain simulation. Events created by direct sampling algorithms are independent from each other. For every event the algorithm starts again with the prior probability distribution. The Markov chain Monte Carlo (MCMC) algorithm generates a new event by making a random change to the current event [4]. More accurately "The next state is generated by randomly sampling a value for one of the non-evidence variables X_i, conditioned on the current values of the variables in the Markov blanket of X_i." [4] In this context a state is a complete assignment of values to all variables. The MCMC algorithm walks through the state space by randomly assigning values to non-evidence variables, the evidence variables are kept constant during the process. Every state visited represents a sample for the estimation of the query probability. The number of times a query variable has a certain value is used in computing the probability estimate. The values are normalized by the total amount of states visited. The MCMC algorithm works because of properties of Markov Chains. When the process is run for a long time the process stabilizes into a dynamic equilibrium. When this equilibrium is reached the time spent in each state is proportional to the posterior probability of being in this state. So as with the other sampling algorithms: the longer the algorithm is run, the better the estimate of the query probability gets.

2.2.4 Dynamic Bayesian Networks

Until now the assumption has been made that the modeled processes are static. This means that values of variables do not change when time passes. For many problems this is not a problem. For example, modeling car diagnosis statically is not a problem because the defect will normally not change until the mechanic takes action. The evidence variables will not change during the diagnosis. In other situations where time plays an important role, a static model will not be sufficient. If variables change over time a dynamic or temporal model is necessary. Examples could be the medical domain, robotics or other areas where information could change over time. There are a few ways to implement a temporal model. The one discussed here is dynamic Bayesian networks.

A dynamic Bayesian network is a Bayesian network that models a temporal process. The main principle of temporal modeling is that the system is seen as a static process viewed at different times. The model consists out of several time slices. Each time slice represents the static model at a certain point in time. If the time slices are independent of each other there is no temporal model and the model is simple static. When time slices are dependent of each other, some of the variables in a time slice t are conditionally dependent of the same variables in time slice t-1. Variables could depend on more than one time slice, but it is very common to limit the temporal dependence to one time slice. A process which is limited in this way is known as a first-order Markov process. This type of process makes use of the Markov assumption: "the current state only depends on a finite amount of previous states." In this case the current state only depends on the previous state. Another necessary assumption to make temporal modeling work is that the process to be modeled is a stationary process. When a process is a stationary process the "laws" of the process do not change in time. This means that the dynamic behavior between time slices does not change. This assumption is useful because it limits the amount of conditional probabilities between time slices.

To put it simply: the conditional probability for a variable in time slice t+1 with a parent in time slice t is the same for every pair of time slices t and t+1.

Network Structure

Basically the variables in a dynamic Bayesian network can be divided into three groups: Context, state and evidence. Context variables can be used to model static aspects of the process, parts which do not change over time. State variables usually model a hidden variable which is to be inferred. This could be the weather, the emotional state of a subject, battery capacity or other features which cannot be detected directly. Evidence variables are the sensors of the network. These variables are used to model features of the process that can be detected or measured directly.

A dynamic Bayesian network can consist out of an arbitrary amount of time slices. When modeling the temporal process one is free to choose the amount of time slices. The amount of time slices has of course an effect on the accuracy of the model and the time necessary to compute the desired probabilities. A general dynamic Bayesian network is presented in Figure 2:

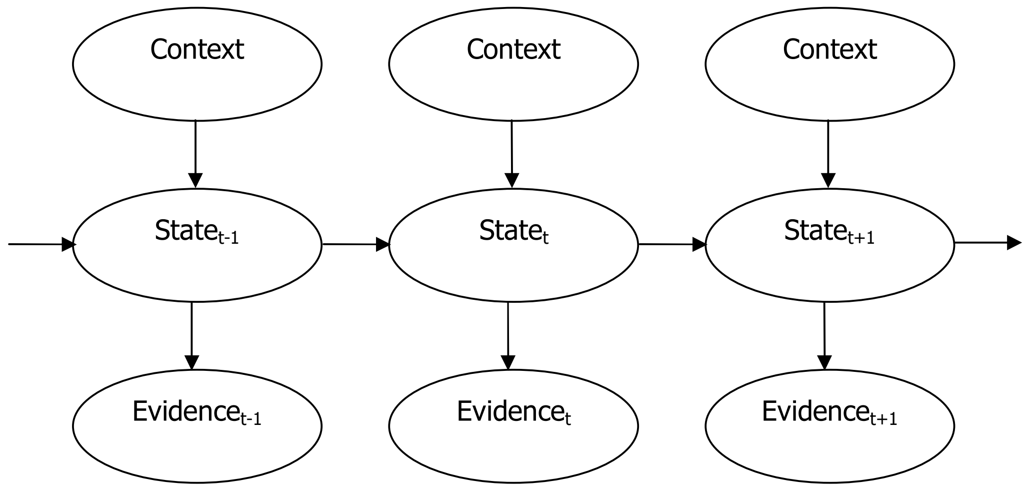


Figure 2: Dynamic Bayesian Network

The necessary CPTs are omitted from the figure; context variables have a prior probability distribution, state variables have conditional distributions conditioned on the context and the previous state variable and the evidence variables have conditional distributions that depend on the state variable from the same time slice. The context is a static variable; the values are assigned once to all the context variables of all time slices. Context variables are not a necessary requirement; they can be removed if there are no relevant, static factors in the problem to be modeled. Since they are not meant to change over a short period of time, they have a constant or deterministic influence on the state variables. An example of a context variable could be the location of a weather prediction system. There is a very big difference between predicting the weather in the Sahara desert or in the African rainforest. But when the system is in place the context will not change and will only cause a constant, deterministic influence on the model.

The system can now be simplified by removing the CPT entries that are conditioned on values of the context variables which now cannot occur, effectively removing the context variables from the model and the inference process.

To construct a DBN, three kinds of information have to be specified [4]: the prior distribution over the state variables, $\mathbf{P}(\mathbf{S}_0)$; the transition model $\mathbf{P}(\mathbf{S}_{t+1}|\mathbf{S}_t)$; and the sensor model $\mathbf{P}(\mathbf{E}_t|\mathbf{S}_t)$. Also the graphical structure of the model has to be specified. To begin with the topology of the prior distribution and the first time slice is enough, when more time slices are necessary they are simply copies of the first time slice.

Inference in Dynamic Bayesian Networks

Inference in dynamic Bayesian networks and temporal models in general can be divided into four different tasks [4]:

- Filtering: calculating $\mathbf{P}(\mathbf{S}_t|\mathbf{e}_{1:t})$, i.e. computing the posterior distribution of the current state of the model given all evidence up to the current time slice.
- Prediction: calculating $\mathbf{P}(\mathbf{S}_{t+k}|\mathbf{e}_{1:t})$ for a k that k > 0, i.e. computing the posterior distribution of a future state S_{t+k} given all evidence up to the current time slice.
- Smoothing: calculating $\mathbf{P}(\mathbf{S}_k|\mathbf{e}_{1:t})$ for a k that $0 \leq k < t$, i.e. computing the posterior distribution of a past state k given all evidence up to the current time slice.
- Most likely explanation: calculating $\text{argmax}_{\mathbf{s}_{1:t}} P(\mathbf{s}_{1:t}|\mathbf{e}_{1:t})$, i.e. computing the sequence of states **s** which are most likely to have generated the observed evidence **e**.

These inference methods are explained in more detail in [4]; these methods are general temporal inference methods and are used in different kinds of temporal models. In the case of dynamic Bayesian networks it is important to realize that DBNs are Bayesian networks, so all BN inference algorithms can be used for DBNs. A process called unrolling is used to add time slices to the network to accommodate the amount of available evidence. Once the network is unrolled, the standard algorithms can be used to calculate the desired query. One has to be cautious when implementing unrolling. A simple approach would be to first unroll the network and then to perform inference, but this leads to linear increase of the necessary space and time. The process is linear in the amount of time slices O(t). Using a recursive process which only requires two time slice to be in memory the necessary amount of time and space can be made constant O(1) of the amount of time slices. It works by summing out a time slice and then unrolling a new one into the model. This process repeats itself until all evidence has been used in the calculation. The variable elimination algorithm discussed earlier can be used to implement this process. Sadly the total complexity for exact inference is still exponential and this means that when the model contains a large amount of state variables it is infeasible to use exact inference for DBNs. The only viable option at this moment in time to perform inference in DBNs efficiently is to use approximate inference algorithms.

Approximate Inference

As with exact inference, because DBNs are Bayesian networks, standard approximate inference algorithms like likelihood weighting and MCMC can be used to compute queries for the network. For the standard algorithm to be efficient they will have to be adapted for DBNs. The problem of unrolling the network causes the same problems.

An effective approximate algorithm family for inference in DBNs is called particle filtering. These algorithms are similar to likelihood weighting algorithms, but because of two features they are better suited for inference in DBNs. These features are [4]:

- *The algorithms use the samples themselves as an approximate representation of the current state distribution.* This assures that updating only needs "constant" time per update.
- *The algorithms focus the set of samples on the high-probability regions of the state space.* This makes sure that the state variables actually depend on the evidence variables. With likelihood weighting algorithms the problem is that because of the structure of the network the generated samples become independent of the evidence. This is what makes likelihood weighting algorithms totally unusable for DBNs.

Particle filtering works as following:

1. A population of N samples is created by sampling from the prior distribution of time slice 0, the initial state of the system.
2. Every sample is propagated forward through the network by sampling the new state value s_{t+1}, using $P(\mathbf{S}_{t+1}|\mathbf{s}_t)$, given the current state value of the sample.
3. Each sample is weighted by the likelihood of being in the sampled state given the new evidence $P(\mathbf{e}_{t+1}|\mathbf{s}_{t+1})$.
4. The whole population is resampled to generate a new population of N samples. The sampling process samples from the current population. The probability that a sample is selected depends on the weight calculated at step 3. The larger the weight the larger the probability of selection.
5. The process now jumps back to step 2 to start on the next time slice. The process is repeated until all time slices have been visited.

Particle filtering algorithms are efficient for approximating inference in dynamic Bayesian networks. Because they sample values from the distributions of the random variables they can also easily be used for inference approximation in continuous or hybrid DBNs.

3 Paper Review

For the literature survey 17 papers have been reviewed. These papers all have subjects related to affective computing and (dynamic) Bayesian networks. The papers have been divided into eight different topic areas:

- Active affective state detection
- User modeling
- Education
- Mental state detection
- Empathic and emotional agents
- Cognitive workload detection
- Facial recognition
- Natural communication

3.1 Active Affective State Detection

In this section papers are reviewed regarding active affective state detection. Interesting is the "active" part, which has to do with dynamically selecting the number of sensors used in the model when inferring the affective state.

3.1.1 Active Affective State Detection and User Assistance with Dynamic Bayesian Networks

X. Li, Q. Ji, [6]

Abstract

With the rapid development of pervasive and ubiquitous computing applications, intelligent user-assistance systems face challenges of ambiguous, uncertain, and multimodal sensory observations, user's changing state, and various constraints on available resources and costs in making decisions. This paper introduces a new probabilistic framework based on the dynamic Bayesian networks (DBNs) to dynamically model and recognize user's affective states and to provide the appropriate assistance in order to keep user in a productive state. An active sensing mechanism has been incorporated into the DBN framework to perform purposive and sufficing information integration in order to infer user's affective state and to provide correct assistance in a timely and efficient manner. Experiments involving both synthetic and real data demonstrate the feasibility of the proposed framework as well as the effectiveness of the proposed active sensing strategy.

Comment

The authors have created a generic framework for applying Bayesian networks to user modeling problems. They call their model the Context-Affective State-Profile-Observation model. The model can be used to infer a user's affective state from observations with sensors given the context and the user's personal profile. Interesting about their approach is that they use techniques from information theory and utility theory to dynamically select sensors for the inference procedure and to decide when to provide assistance to the user.

They use Shannon's measure of entropy to calculate how much the addition of a piece of evidence would reduce the uncertainty of the hypothesis of the affective state. To be more exact they use Shannon's entropy measure and the mutual information measure to calculate the entropy over the affective state variable given the previous affective state and the piece(s) of evidence.

$$ENT(H) = -\sum_i p(h_i)\log p(h_i)$$
$$I(H^t; E_j) = ENT\left(H^t \middle| H^{t-1}\right) - \sum_j p(e_j) ENT\left(H^t \middle| H^{t-1}, e_j\right)$$

Equation 11: Shannon's Information Measure and the Mutual Information Measure

The equations can be change to be able to use them for multiple evidence sources. In this case the distribution of the evidence variable is replaced by a joint distribution of multiple variables. They have combined this information measure with a cost function for the sensors to create a utility function they maximize to decide how many and which sensors should be used for the inference process. The cost function is defined by the authors and gives each sensor a numerical value. To calculate the total cost of a sensor subset relative to the maximum possible cost the following equation is used:

$$C(E) = \frac{\sum_{i=1}^{n} C_i}{\sum_{j=1}^{m} C_j}$$

Equation 12: Cost function

Using the cost function and the mutual information measure the authors have defined a utility function:

$$EU(H^t, E) = \alpha I(H^t; E) - (1-\alpha)C(E)$$

Equation 13: Utility Function

The factor α is used to balance the two terms.
To choose the optimal sensor action for a time slice, the authors examine every possible sensor configuration and choose the configuration with the highest utility.

To decide if the system should assist the user, the authors have created a similar utility rule to decide on the best possible action. First a weighted sum is calculated of the posterior probabilities of the different affective states. When this sum exceeds a specified threshold the system will assist the user, and now the best possible action is calculated using a utility function. This function consists out of a cost and a benefit function and the best possible action is calculated by again maximizing the utility function.

Using a combination of utility and information theory to determine the amount of sensors for information gathering and to determine the best assistance action is interesting. The authors state that using a sensor costs something. It could be processing time, energy or the user's patience. Using more sensors does not have to imply that the result will automatically be better.

The authors have conducted experiments with and without active information fusion and when the utility rules are used the system does perform better and faster. It now takes less time slices to reach the assistance threshold.

On a more critical note; the computation of the mutual information looks quite computationally expensive. The authors claim that the necessary probabilities are available after an iteration of the forward and backward inference propagation of the DBN, but it looks like this is not entirely correct. After the inference propagation is completed the posterior probabilities of the affective state are available but to be able to compute the mutual information measure every possible combination of sensors has to be considered. This does not happen when updating the DBN, only the measured values of the sensors are used for this procedure. Also to compute the conditional information measure (which is used for computing the mutual information) a weighted average is computed which needs every probability of the joint probability distribution of the subset of sensors. In short the process is exponential in the amount of sensors, and at the moment the fact that to find the sensor configuration with the highest utility every possible combination has to be calculated hasn't even been taken into account yet. The authors have experimented with a relatively small DBN. It contained two time slices and thirteen nodes per time slice. If the network size would increase the necessary computation time would increase fast and the benefit of saving processing time by not using every sensor would eventually disappear. Other benefits would stay; direct communication with the user by asking a question, instead of non-intrusive sensors, can still be postponed until there is no other viable option. Active Affective State Detection is interesting and useful but will be hard to implement in real world application because of the computational complexity of the system.

3.1.2 A Probabilistic Framework for Modeling and Real-Time Monitoring Human Fatigue

Q. Ji, P. Lan, C. Looney, [7]

Abstract

The authors introduce a probabilistic framework based on the Bayesian Networks (BNs) for modeling and real-time inferring human fatigue by integrating information from various sensory data and certain relevant contextual information. They first present a static fatigue model that captures the static relationships between fatigue, significant factors that cause fatigue and various sensory observations that typically result from fatigue. Such a model provides mathematically coherent and sound basis for systematically aggregating uncertain evidences from different sources, augmented with relevant contextual information. The static model, however, fails to capture the dynamic aspect of fatigue. Fatigue is a cognitive state that is developed over time. To account for the temporal aspect of human fatigue, the static fatigue model is extended based on the Dynamic Bayesian Networks (DBNs). The dynamic fatigue model allows integrating fatigue evidences not only spatially but also temporally, therefore leading to more robust and accurate
fatigue modeling and inference. A real time non-intrusive fatigue monitor was built based on integrating the proposed fatigue model with a computer vision system the authors developed for extracting various visual cues typically related to fatigue.

Performance evaluation of the fatigue monitor using both synthetic and real data demonstrates the validity of the proposed fatigue model in both modeling and real time inference of fatigue.

Comment

The authors have created a system for monitoring fatigue using non-intrusive sensors. From their literature survey they have concluded that fatigue is a significant factor in transportation accidents. Systems that are able to detect fatigue can be used to improve driver vigilance and prevent fatigue related accidents. To get better result they measure several different factors which can contribute to fatigue and measure different symptoms which could be signs of fatigue. Their complete model is shown in Figure 3:

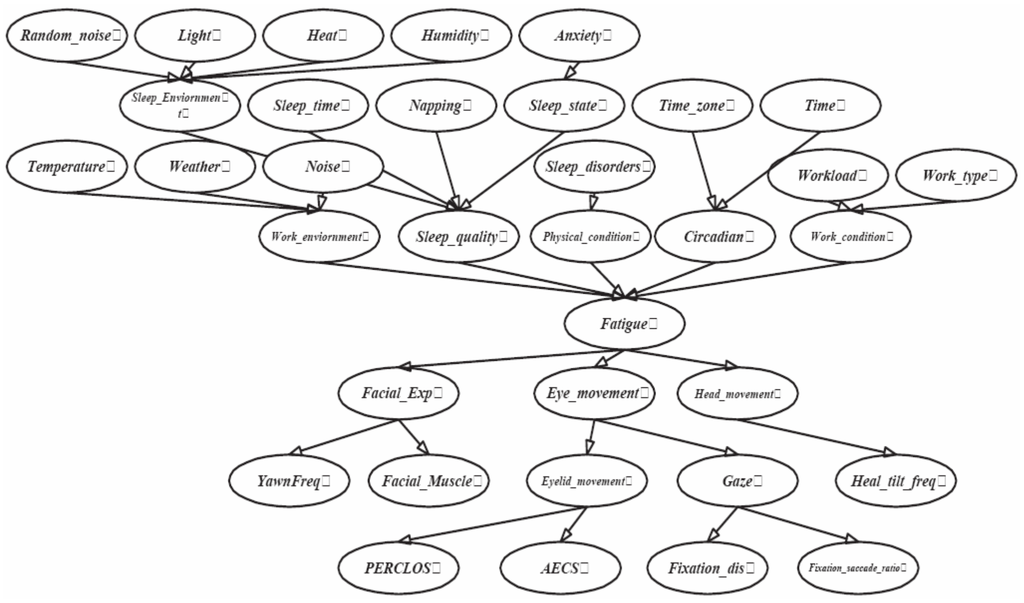


Figure 3: Bayesian Network for Modeling Fatigue

In the middle is the fatigue variable which is used to infer the fatigue state of the subject, variables above are context variables and variables below are evidence variables. This Bayesian network is the static variant of their system. To be able to deal with temporal influences, they have also created a dynamic Bayesian network. This network consists out of two time slices. The time slices are identical to the BN in Figure 3, but are connected with each other by conditional links to model temporal effects. Connected in this way are the fatigue variable and several of the evidence variables. To construct the necessary CPTs they have used data from several large-scale subjective surveys as a basis for determining the necessary probabilities. To decrease the amount of necessary probabilities they have used the noisy-or principle. The authors note that it still has to be validated if the use of noisy-or principle is appropriate for this model. The authors have done two experiments: one with synthetic and one with real data. The experiment with synthetic data was performed to demonstrate that the model can fuse different types of sensor data to infer the fatigue level of a subject. The authors have generated likely situations of sensor values where the subjects would be fatigued. The experiment with real data was done with 8 human subjects.

The subjects were asked to perform two tests regarding their psychomotor skills. The objective of the test was to push a button when a light flashed on the screen. The first test was performed at 9 pm, the second at 7 am after 25 hours of sleep deprivation. To quantify a subject's performance their response time was measured. The average response times of subjects were plotted against their respective fatigue predictions and a correlation coefficient of about 0.95 was determined. These results have given the authors a strong indication that their model is capable of modeling and predicting fatigue.

Their research was focused on developing a theoretical framework for fatigue modeling. The authors have defined a general structure for the fatigue model; to get better results they have stated that they will have to conduct more research to improve the model structure and the parameterization of the model.

The structure of the model is very straight forward, as mentioned before: context variables influence the hypothesis, and the hypothesis influences evidence variables which can be measured. Since they used results from several related studies for determining the relevant factors of fatigue, it is likely that structure of their model is good enough to model fatigue accurately. The biggest problem isn't the structure, but the necessary CPTs for the model. Getting all the probabilities requires a massive amount of data which, considering the size of the DBN is virtually impossible to collect. Using the noisy-or principle the authors have reduced the necessary amount of data, but this decision will influence the accuracy of the model.

The experiments were conducted on a quite small scale. This isn't a big problem because their work is more theoretical than directly aimed at an application. Their experiment with synthetic data covered 26 different situations and their human experiment had 8 participants. According to the authors theoretically there are 2^{22} different possible inference results, so when the model is actually implemented in an application it is a good idea to conduct larger experiments with more situations and more participants to get a more detailed view of the performance of the application.

3.2 User Modeling

The goal of user modeling is to be able to predict the mental or affective state of the user. Using this information, systems are able to adapt to the user to improve the human-computer communication.

3.2.1 Modeling the Emotional State of Computer Users

G. Ball, J. Breese, [8]

Abstract

The authors describe the structure of a Bayesian network designed to monitor the behavior of a user interacting with a conversational computer and use that information to estimate the user's emotional state. Their model of emotional state uses two discrete dimensions, valence (bad to good) and arousal (calm to excited), to capture the dominant aspects of physical emotional response. Emotion is expressed behaviorally in a variety of ways, including by linguistic choices, qualities of vocal expression, and movement.

In this paper, the authors focus on vocal expression, and its correlation with emotional state using the psychological literature to suggest the appropriate parameterization of their Bayesian model.

Comment

The authors have developed a Bayesian network that models the causal links between the user's emotional state, represented by valence and arousal, and 4 different parameters of speech:

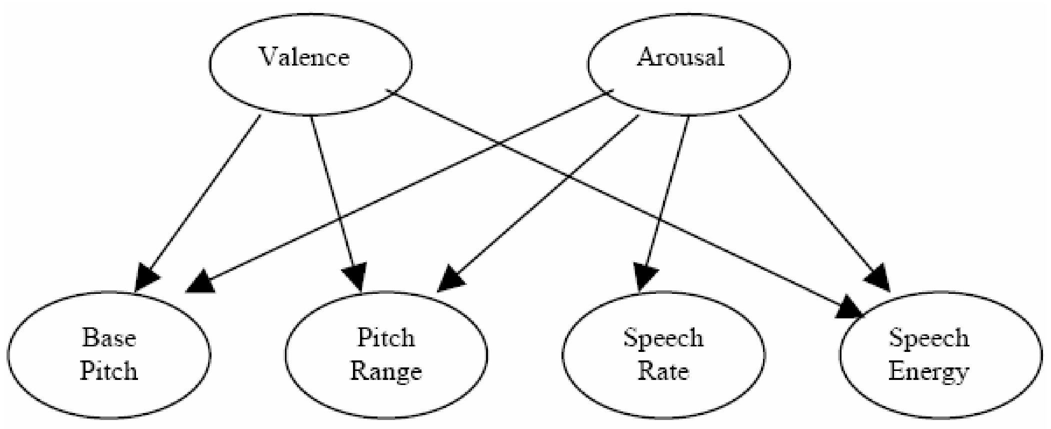


Figure 4: Bayesian network for emotion detection with speech parameters

In its current form the model is not very useful. The authors note that according to relevant psychology literature, emotion detection using these speech parameters is still very unreliable. There is a lot of ambiguous information in speech and at the moment the only way to get better results is to use more information sources like facial expressions and word choices.

Once there is a better understanding of the effect of emotions on human speech, it might be possible to create models and systems that only require speech to be able to predict a subject's affective state.

3.2.2 Harnessing Models of Users' Goals to Mediate Clarification Dialog in Spoken Language Systems

E. Horvitz, T. Paek, [9]

Abstract

Speaker-independent speech recognition systems are being used with increasing frequency for command and control applications. To date, users of such systems must contend with the fragility of recognition with subtle changes in language usage and environmental acoustics. The authors describe work on coupling speech recognition systems with temporal probabilistic user models that provide inferences about the intentions associated with utterances. The methods can be employed to enhance the robustness of speech recognition systems by endowing the systems with the ability to reason about the costs and benefits of action in a setting and to make decisions about the best action to take given uncertainty about the meaning behind acoustic signals.

The methods have been implemented in the form of a dialog clarification module that can be integrated with legacy spoken command and control systems. The authors describe representation and inference procedures and present details on the operation of an implemented spoken command and control development environment named DeepListener.

Comment

The authors have designed a system that increases the recognition rate of a command-and-control system by modeling the user's goals. When recognition fails the system uses the current goal hypothesis and a utility function to determine the best course of action. This could be asking for clarification or just executing the most likely given command. Their system consists out of different modules developed at Microsoft (the authors are Microsoft researchers). They integrated a speech recognizer (ASR), a Bayesian network tool and a command-and-control system to create their system DeepListener.
Their user model is a dynamic Bayesian network, using multiple time slices to infer the goal of a user of the system. Two time slices of the network are shown in Figure 5, note that only the variables "Speaker's Goal" and "User's Spoken Intention" have temporal links between time slices. This is a logical choice because these variables represent what the user (possible) wants from the system.

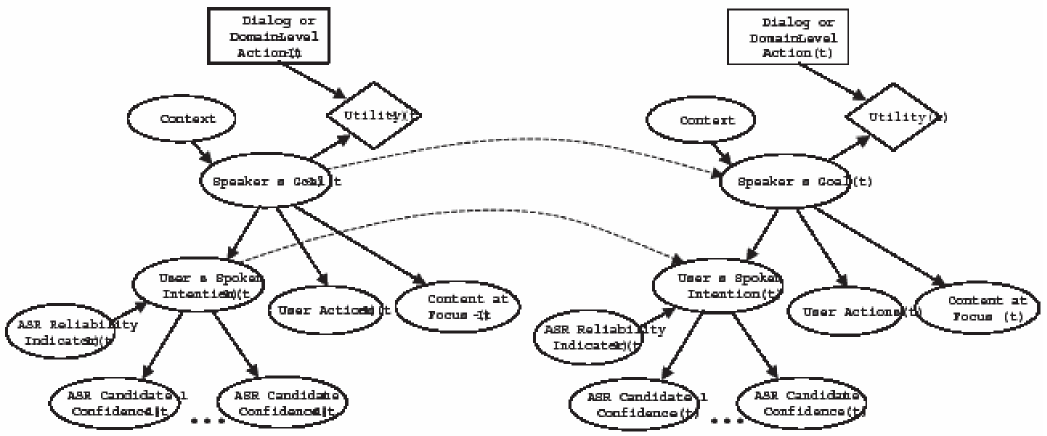

Dialog or DomainLevel Action(t-1)
Context
Utility(t-1)
Speaker s Goal(t-1)
User s Spoken Intention(t-1)
User Action(t-1)
Content at Focus (t-1)
ASR Reliability Indicator(t-1)
ASR Candidate 1 Confidence(t-1)
ASR Candidate Confidence(t-1)
...
Dialog or DomainLevel Action(t)
Context
Utility(t)
Speaker s Goal(t)
User s Spoken Intention(t)
User Actions(t)
Content at Focus (t)
ASR Reliability Indicator(t)
ASR Candidate 1 Confidence(t)
ASR Candidate Confidence(t)
...

Figure 5: DeepListener dynamic Bayesian network

Every time the ASR picks up an utterance the data is inputted in a new time slice and inference is performed on the network. Next the utility is calculated to determine the next action.

The paper reports about the progress the authors have made with the Deeplistener system. They did not have any experimental result at the moment of writing the paper, so no real verdict can be given of the effectiveness of the developed system. It is to be expected however that the addition of user modeling to a command-and-control system will improve the interaction between user and system. If the system doesn't have a clue what the user wants, it will now tell the user so a solution can be found. Standard systems do not have this ability and can only report an error or execute the wrong action.

3.2.3 Modeling Patient Responses to Surgical Procedures during Endoscopic Sinus Surgery using Local Anesthesia

K. Sakai, M. Masaaki, K. Yokoyama, [10]

Abstract

The authors have constructed a patients' response model during endoscopic sinus surgery using local anesthesia. During this surgery, the patient keeps a high level of consciousness, and feels pressure, vibration and pain from the surgical procedures. These feelings affect the patient's mental and physiological condition, and the excessive fluctuation of patient's condition interrupts the surgery. Therefore, awareness of the patient's condition and control of them are important to the surgeon. With this in view, the authors are constructing a model that can represent the patient's mental and physiological responses. To develop the model, the causal relationship between surgical procedures and patient's responses was investigated. It becomes clear that the patient's response is dependant on the pattern of the procedure sequence, the length of the procedure and the effects resulting from the procedure such as vibration and pain. The authors constructed a model of these behaviors using the Bayesian network.

Comment

The authors have recorded data from ten surgical procedures. They have divided the procedure into four surgical phases. For every phase they have measured the heart rate and breathing rate of the patient and recorded pain complaints from the patient. Also, they recorded all the actions of the surgeon. Using this data they created causal models describing the relationship between surgical procedures and patient responses:

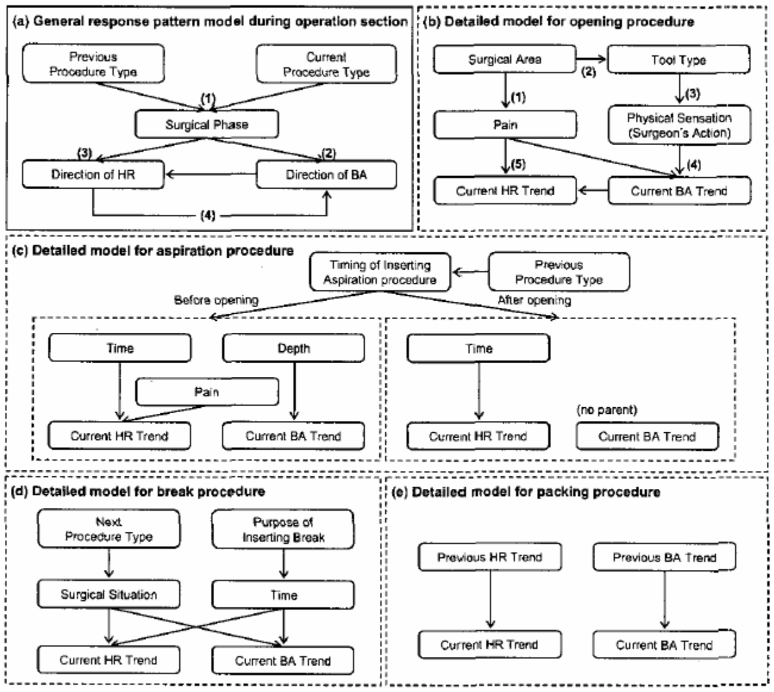


Figure 6: Structure of causal relationships between surgical procedures and patient responses

The authors have defined variables and have determined the casual relationship between the different variables by focusing on patterns in the patient's response and the surgeon's actions. Different tree structure with the variables where then generated and using the Minimum Description Length (MDL) principle the authors calculated which structure was optimal for the model. They have developed a general model and several smaller models specific for a phase of the operation.

The authors noted that there were significant differences in the heart rate and breathing rate data between patients, but extra statistical analysis showed that this was manageable and in the case of the breathing rate data the differences were probably caused by tension differences of the respiratory belt sensor.

The amount of test subjects seems a bit small, only ten operations were recorded. Since they use learning techniques to determine the model structure a larger dataset might lead to better results. The paper also only mentions that the subjects were Japanese adults. Perhaps age and gender have an influence on the patient's behavior and are worth of researching. Other psychological factors could also have an influence. Having refined models for every phase of the operation will improve the overall model. Using only the general pattern model would have caused specific patient responses only occurring at certain phases to be lost. The authors are still working on refining and validating the model, once the results of the model improve the want to integrate it into their surgical simulation software.

3.2.4 Bayesian Network Modeling of Offender Behavior for Criminal Profiling

K. Crews Baumgartner, S. Ferrari, C.G. Salfati, [11]

Abstract

A Bayesian network (BN) model of criminal behavior is obtained linking the action of an offender on the scene of the crime to his or her psychological profile. Structural and parameter learning algorithms are employed to discover inherent relationships that are embedded in a database containing crime scene and offender characteristics from homicide cases solved by the British police from the 1970s to the early 1990s. A technique has been developed to reduce the search space of possible BN structures by modifying the greedy search K2 learning algorithm to include a-priori conditional independence relations among nodes. The new algorithm requires fewer training cases to build a satisfactory model that avoids zero-marginal-probability (ZMP) nodes. This can be of great benefit in applications where additional data may not be readily available, such as criminal profiling. Once the BN model is constructed, an inference algorithm is used to predict the offender profile from the behaviors observed on the crime scene. The overall model predictive accuracy of the model obtained by the modified K2 algorithm is found to be 79%, showing a 15% improvement with respect to a model obtained from the same data by the original K2 algorithm. This method quantifies the uncertainty associated with its predictions based on the evidence used for inference. In fact, the predictive accuracy is found to increase with the confidence level provided by the BN. Thus, the confidence level provides the user with a measure of reliability for each variable predicted in any given case.

These results show that a BN model of criminal behavior could provide a valuable decision tool for reducing the number of suspects in a homicide case, based on the evidence at the crime scene.

Comment

The authors have created a system for inferring psychological factors from crime scene evidence. Using psychological profiles and crime scene data they defined 36 crime scene variables and 21 psychological variables, which are called offender variables. They created a database from 247 sample cases collected from British police forces regarding single offender/single victim homicides in the time period from the 1970s to the early 1990s.

The structure of the Bayesian network was learned from the data using greedy search algorithms K2 and K2′. In the paper the authors compared the performance of the two algorithms. K2′ performed better than K2 because it uses conditional independence assumptions to decrease the size of the search space of all possible Bayesian network structures. Another advantage of using these independence assumptions is that the K2′ has a lower computational complexity than the K2 algorithm.

After learning the structure of the network the authors used the EM learning algorithm to determine the CPT values. Inference is done by computing the posterior probability of every offender variable given the instantiated crime scene variables. The authors call this the marginal probability of a variable. The state with the largest probability of all states of an offender variable is chosen as the prediction value for that offender variable.

To be able to test the Bayesian networks created by the K2 and the K2′ algorithms the authors divided the database into a training set and a validation set. The networks were trained with the cases from the training set and validated with the cases from the validation set. Their results show that the K2′ algorithm outperforms the K2 algorithm.

There were some interesting observations in the data. For validation of the Bayesian networks the authors used 47 of the 247 cases. Because there are 21 offender variables which have to be predicted there are 21 predictions per validation case and 987 predictions in total. A value of an offender variable is chosen as the prediction value when it has a probability larger or equal to 0.5. All the variables are binary so it's either true or false. Using the Bayesian network created by the K2′ algorithm only 798 of the total 987 predictions had a marginal probability ≥ 0.5. In the other 189 cases the probability of both states was 0. This is very strange because now the marginal probabilities do not sum up to 1. This is a violation of one of Kolmogorov's axioms. The authors call these nodes "Zero Marginal Probability" (ZMP) nodes. They have observed them when performing inference on a Bayesian network with an inadequate number of training cases. The authors added the amount of ZMP predictions as a factor to the algorithm comparison and note that this makes the comparison more accurate. A result is that BNs trained with K2′ contain less ZMP nodes than BNs trained with K2. This translates to requiring less training data when using K2′ instead of K2.

The occurrence of ZMP nodes is very strange, as noted before they break one of Kolmogorov's axioms. All probabilities of a distribution should sum to one.
It looks like the authors have made some sort of mistake somewhere. They use the EM algorithm to learn the CPT entries from their data; perhaps they have made a mistake here. The authors only mention the EM algorithm twice and do not give any details. The algorithm itself is correct and cannot break Kolmogorov's axioms; it uses standard inference algorithms for BNs. To check what happens when a ZMP node occurs a joint probability distribution and a Bayesian network have been created which should give the ZMP probabilities.

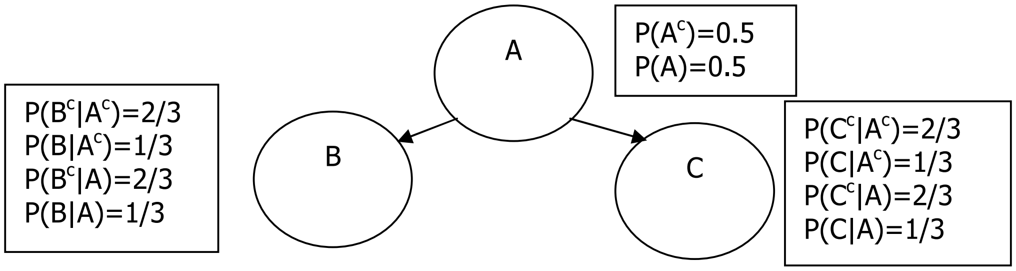


Figure 7: Bayesian network for testing ZMP nodes

Table 2: joint probability distribution for testing ZMP nodes

A	B	C	P(ABC)
F	F	F	1/6
F	F	T	1/6
F	T	F	1/6
F	T	T	0
T	F	F	1/6
T	F	T	1/6
T	T	F	1/6
T	T	T	0

To create a ZMP prediction, the query P(A|BC) is chosen to be 0 for A and A^c.
So:

$$P(A|BC) = P(A^c|BC) = 0 = \frac{P(ABC)}{P(BC)} = \frac{P(A^cBC)}{P(BC)} = 0$$

$$\frac{P(ABC)}{P(BC)} = 0 \Rightarrow P(ABC) = 0, \frac{P(A^cBC)}{P(BC)} = 0 \Rightarrow P(A^cBC) = 0$$

$$P(ABC) = 0 \wedge P(A^cBC) = 0 \Rightarrow P(ABC) + P(A^cBC) = 0$$

$$P(ABC) + P(A^cBC) = P(BC) = 0 \Rightarrow \frac{P(ABC)}{P(BC)} = \frac{P(A^cBC)}{P(BC)} = \frac{0}{0}$$

There is a contradiction here; if both P(A|BC) and P(A^c|BC) are set to 0, they become 0/0 and thereby become undefined. The authors have used Matlab for implementing their system. When matlab is presented with a 0/0 it returns the IEEE standard: NaN (Not a Number). Perhaps that the Matlab Bayes Net Toolbox changes this to a 0, this has not been tested so no further conclusion can be given about what exactly causes the appearance of the ZMP nodes.

3.3 Education

The applications of affective computing and (D)BNs in education are quite similar to those in user modeling. Students are modeled to be able to infer their affective state or their progress in solving a problem.

3.3.1 DT Tutor: A Decision-Theoretic, Dynamic Approach for Optimal Selection of Tutorial Actions

R.C. Murray, K. VanLehn, [12]

Abstract

DT Tutor uses a decision-theoretic approach to select tutorial actions for coached problem solving that are optimal given the tutor's beliefs and objectives. It employs a model of learning to predict the possible outcomes of each action, weighs the utility of each outcome by the tutor's belief that it will occur, and selects the action with highest expected utility. For each tutor and student action, an updated student model is added to a dynamic decision network to reflect the changing student state. The tutor considers multiple objectives, including the student's problem-related knowledge, focus of attention, independence, and morale, as well as action relevance and dialog coherence. Evaluation in a calculus domain shows that DT Tutor can select rational and interesting tutorial actions for real-world-sized problems in satisfactory response time. The tutor does not yet have a suitable user interface, so it has not been evaluated with human students.

Comment

The authors have created a tutoring system using a Dynamic Decision Network (DDN). A DDN consists out of a DBN expanded with utility and decision nodes. The system can be used to tutor almost any type of problem, but the authors have chosen to only implement some calculus problems.
Their reasons for doing so were that calculus problems are not trivial, but not too large and an interface for the calculus domain had already been developed by another researcher. The authors modified this interface to be able to observe every step in the problem solving process. The model created by the authors consists out of three time slices:

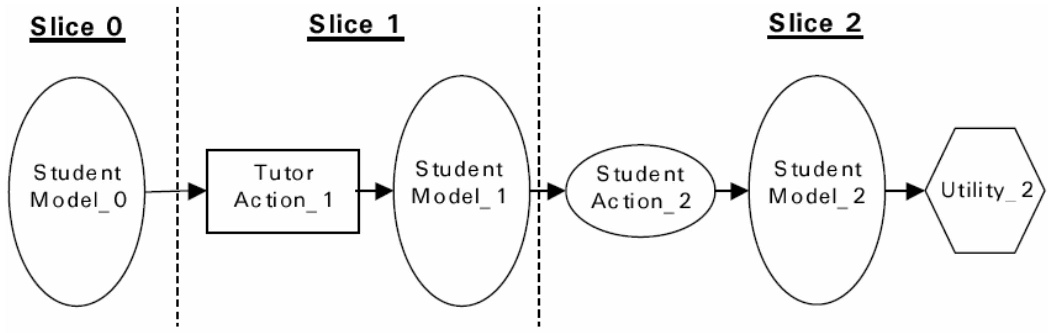


Figure 8: Tutor action cycle network overview

The authors call their network a "tutor action cycle network". The first slice contains the tutor's beliefs about the current student state. This model consists out of more variables than shown in the figure. There are sub variables that address the student's affective state (morale, independence), problem solving progress and the student's problem-related knowledge.
The next slice contains tutor action nodes which contain the possible actions a tutor can do and a student model which represent how the student could react on the performed tutor action. Two different tutor action nodes are defined: Tutor type nodes model in which way the tutor action should be executed, i.e. with a positive or a negative tone. Tutor topic nodes decide which part of the problem should be tutored. Both nodes are decision nodes.
The third slice contains student action nodes which model the possible actions a student can do. They are roughly equivalent to the tutor action nodes. There are student topic and student type nodes. The student topic nodes model which part of the problem the student is working on and the student type nodes model how the student is performing on that part of the problem. Possible values here are: correct, error, impasse and null (do nothing). Also the slice contains again a student model, which now models how the state of the student is affected by the student's actions.
To be able to make a decision for a tutor action the third slice contains a utility model. This model, consisting out of several utility nodes is used to calculate the best possible action for the tutor program. When a tutor action has been performed the slices are emptied and the new state of the student is placed into slice 0 and then the entire process repeats itself.

To evaluate their system, the authors have focused on two different aspects: the tractability of the system and the tutorial action selection ability of the system. Bayesian model for real world problems require a lot of space and time to run. The authors have worked on keeping the response time of the system short. They have evaluated different inference algorithm to determine their influence on the systems response time. The authors have compared an exact inference algorithm with different approximate algorithms with different accuracy levels.

The authors have evaluated the performance of their system by investigating if the system could select rational tutoring actions which are similar to actions performed by human tutors. They mention that the results of the system are promising: the system does show behavior which resembles the approach of human tutors.

The DT Tutor system, developed by the authors, looks promising. It is capable of intelligent tutor behavior, but it is a general framework. According to the paper the probabilities and the utilities for the different models are generated by the system from the problem solution graph. The system does have the option to load probabilities and utilities from an input file, so other data can be used as input for the models. To improve the performance of the system it might be a good idea to use data from real tutor/student sessions. Then the system could adapt to specific student reactions on a specific problem domain. With this ability the system might be able to react faster on common errors made by a large amount of the students and provide better support.

3.3.2 A Probabilistic Framework for Recognizing and Affecting Emotions

C. Conati, X. Zhou, [3]

Abstract

The authors present a framework for affective user modeling that deals with the high level of uncertainty involved in recognizing a variety of user emotions by relying on a Dynamic Bayesian Network. The authors summarize how they used this framework to build a model of player affect to be used by a socially intelligent agent during the interaction with an educational game.

Comment

The authors have created a probabilistic framework for recognizing emotions. They have implemented this framework into an educational game. The game, called "Prime Climb", is designed to teach number factorization. The players must climb mountains which are divided into sectors that have numbers. To climb the players have to choose numbers which do not share any factors with the number of the sector their partner is on. The framework is used to model the player's emotions and personality, to asses the player's goals and to adjust the educational agent's response to the player.

The basis for the probabilistic framework is a Dynamic Decision Network. The authors have developed this network by combining results from psychology and empirical data, gathered from experiments and user studies.

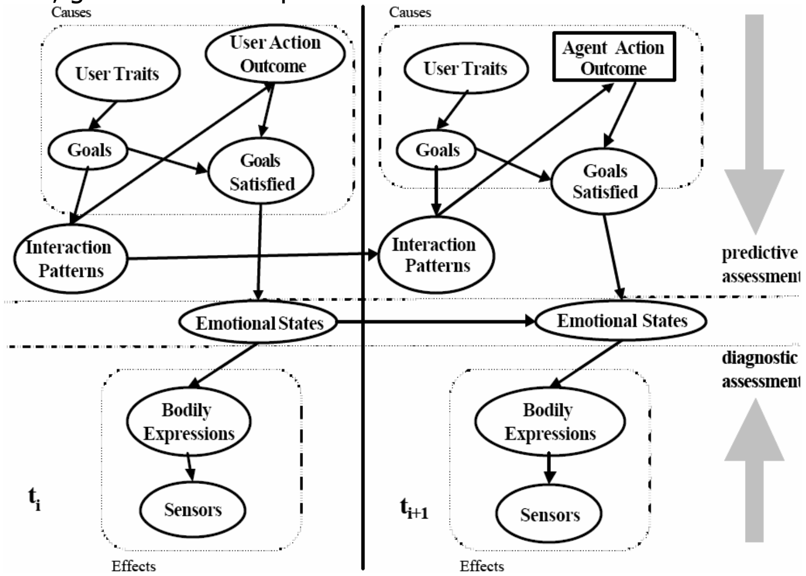


Figure 9: probabilistic framework

The figure shows two time slices of the DDN, a time slice is added to the framework when the user or the agent performs an action. This is a general overview of the DDN. There are several sub-networks which model in more detail different personality traits, goals, interaction patterns and individual actions.

The data for these sub-networks was acquired by conducting Wizard of Oz studies. The authors still have to validate the current model and have to refine some of the links in the DDN. They are running user studies to acquire more data.

3.3.3 Exploiting Emotions to Disambiguate Dialogue Acts

W. Bosma, E. André, [13]

Abstract

This paper describes an attempt to reveal the user's intention from dialogue acts, thereby improving the effectiveness of natural interfaces to pedagogical agents. It focuses on cases where the intention is unclear from the dialogue context or utterance structure, but where the intention may still be identified using the emotional state of the user. The recognition of emotions is based on physiological user input. The authors' initial user study gave promising results that support their hypothesis that physiological evidence of emotions could be used to disambiguate dialogue acts. This paper presents the authors' approach to the integration of natural language and emotions as well as their first empirical results, which may be used to endow interactive agents with emotional capabilities.

Comment

The authors have worked on models which use emotions to clarify dialogue between computers and users. Knowledge of the affective state of a user can prevent misinterpretation of the user. The authors' state: "Even when two utterances are textually identical, the user's emotional state may convey information that reveals entirely different meanings". The authors have created an experimental setup to test this. They have combined an educational game with a pedagogical agent, which gives advice to the player. For their first experiment they used a Wizard of Oz setup to emulate the pedagogical agent with a human tutor. To measure the affective state they use several bio sensors. They measure electrocardiography (ECG), electromyography (EMG), skin conductivity (SC) and respiration (RSP). They have used the acquired data from their experiment to create their model.

Their model consist out of two parts: a Bayesian network for the inference of the affective state and the fusion of the different sensor signals and a language model for determining the best interpretation of an utterance.
The language model is implemented using a weighted finite-state transducer (FST). The model can recognize a fixed set of dialogue acts. It takes natural language utterances as input. The two parts are combined through the set of dialogue acts. The Bayesian network has a node that represents the probability distribution of all dialogue acts. These probabilities are used a weights in the FST. The result of this is that when the affective state of the user changes, this influences the probability distribution of the set of dialogue acts. This changes the weights of the FST of the language model and, this in turn has an influence on the final interpretation of the utterance.

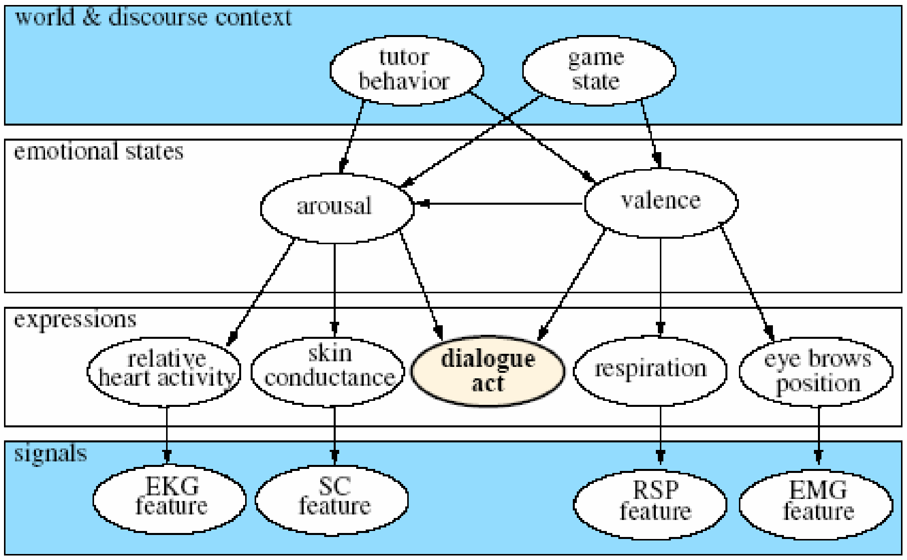


Figure 10: Bayesian network for detection of affective state

The model determines the best interpretation of an utterance by finding the cheapest path in the FST. For clarification the authors have presented a short example with a FST that can only detect the utterances "oh" and "okay" and can determine if the utterance means acceptance or rejection.

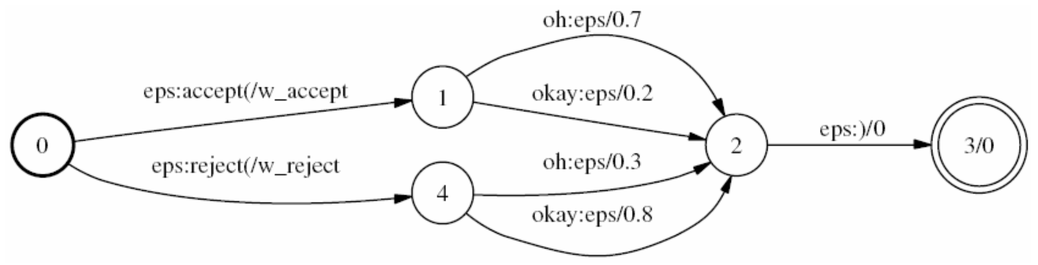


Figure 11: example Finite-State Transducer

The first two weights: w_accept and w_reject are extracted from the dialogue act node from the Bayesian network. The other weights are static and are derived from the language model. In this case in 80 % of the cases okay is used as an acceptance and in 20 % of the cases as a rejection (i.e. sarcasm). Now to determine the meaning of an utterance the Bayesian network is updated with the evidence and the updated probabilities from the dialogue act node are put into the FST. Now the cheapest path in the updated FST can be determined. This cheapest path now forms the most likely interpretation of the utterance.

For the experiment the responses of four test subjects were measured. Every session took about 50 minutes. Features used in the Bayesian network were extracted from the data generated by the sessions. The authors found that there were differences between test subjects regarding physiological behavior. "One subject was found relatively expressive in his heart rate and respiration whereas another had a more expressive skin conductivity." Having more test subjects for the experiment may prevent that these individual differences have a large influence on the total model.

The authors have shown that the affective state does influence the meaning of utterances, being able to detect the affective state can therefore improve the interpretation ability of software agents. But implementing their current system in a real world situation is unlikely because of the invasive sensors they use. These sensors have to be attached to the user for them to work. Systems like these are only viable when the do not give their users discomfort. Use of non-invasive sensors like camera's (assuming no privacy issues) are much more comfortable for the users. It is not a problem to integrate different sensors into the authors' framework. The Bayesian network is naturally suited for integration of different types of sensors. One comment about the implemented Bayesian network is that it is a static one. Using a dynamic Bayesian network may lead to better prediction of the affective state of the user and give better predictions for the correct dialogue act.

3.3.4 A Bayesian Approach to Predict Performance of a Student (BAPPS): A Case with Ethiopian Students

R. Bekele, W. Menzel, [14]

Abstract

The importance of accurate estimation of student's future performance is essential in order to provide the student with adequate assistance in the learning process. To this end, the authors' research aimed at investigating the use of Bayesian networks for predicting performance of a student, based on values of some identified attributes. The authors presented empirical experiments on the prediction of performance with a data set of high school students containing 8 attributes. The paper demonstrates an application of the Bayesian approach in the field of education and shows that the Bayesian network classifier has a potential to be used as a tool for prediction of student performance.

Comment

The authors believe that a student's math performance can be predicted. For this prediction process they have used a Bayesian network. The nodes in the network were chosen by researching literature and discussing the findings with colleagues. Eventually the authors selected eight different attributes.

Questionnaires for collecting data were developed to be able to gather data for the network. Data was acquired from 571 students from an Ethiopian High school. To train the Bayesian network the dataset was split into a training set and a test set using a percentage split. To achieve better results different partition sizes were used for training and testing multiple BNs.

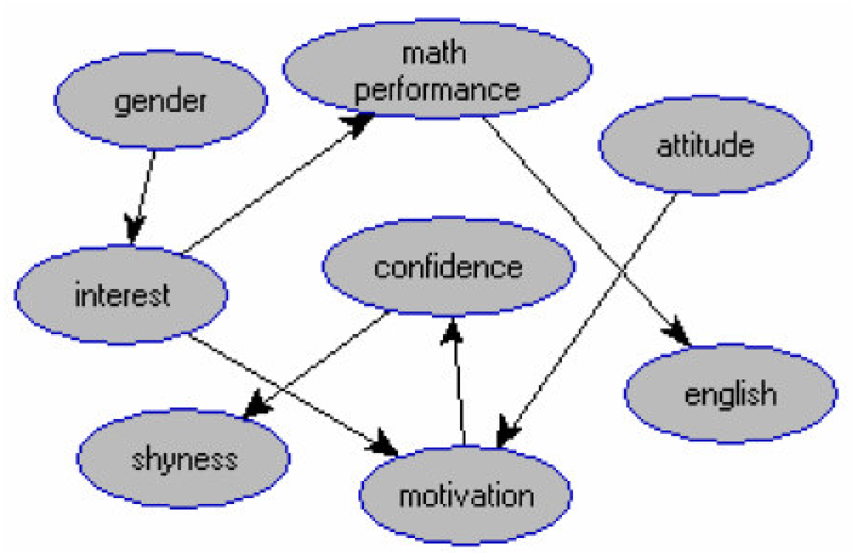


Figure 12: One of the BNs learned during training

After training the networks were tested with the test set and the result were put into a confusion matrix to asses the accuracy of the prediction. For their implementation they chose the network with the lowest prediction error.
The authors note that prediction errors can occur because of students that do not completely tell the truth in the questionnaire, thereby making the test and training data erroneous. To reduce bad data records the authors had added lie detector statements to the questionnaire. When too many of these statements were true the questionnaire would be disregarded. In this way 57 questionnaires were excluded from the dataset leaving 514 data records for the experiment.

The Bayesian network was implemented using an open source software package; the Bayesian Network in Java software package (BNJ). When predicting the student's math performance the probability
P(MathPerformance|Gender,Attitude,Interest,Confidence,Shyness,Motivation,English) was computed using the Logic Sampling algorithm. This algorithm is an example of approximate inference. This means that the algorithm approximates the probability; the longer it runs, the better the approximation gets. The trained network isn't very big. It only has 8 nodes. With the exception of the gender node every node has three possible values. The worst case scenario, using the complete joint probability distribution, needs $2*3^7$ = 4374 probabilities. The Bayesian network shown in Figure 12 on the other hands needs only 71 probabilities. Although the authors did not state that this Bayesian network was the network with the lowest prediction error,

it is reasonable to assume that the network with the lowest prediction error will need a similar amount of probabilities. The authors do not show the final network in the paper, so for the rest of the review it is assumed that the network in Figure 12 is the final network for the model. Approximate inference algorithms are used instead of exact inference algorithms because they have a lower computational complexity, i.e. they are faster. Taking the size of the final network into account, 8 nodes and only 71 probabilities, there may only be a small advantage when using approximate algorithms. In this situation using exact algorithms is feasible and these algorithms will give the exact probability and not an approximation, so using an exact algorithm might yield more accurate predictions.

After training the BN the authors could have implemented an exact algorithm for the inference, but because one can never be sure about the amount of probabilities in the network, using an approximate algorithm may still be the best choice.

When predicting a student's math performance, the use the math performance node as query variable and the other nodes as evidence variables. In 2.2.3 - Definition of Bayesian Networks – it was mentioned that a node is conditionally independent of every other node given its Markov Blanket (parents, children and children's parents). All evidence nodes are used when predicting, so the nodes in the Markov Blanket are also used. Now the prediction can be reduced to:

P(MathPerformance|MarkovBlanket(MathPerformance))

According to [4] this can be rewritten as:

$$P(mp|\mathbf{e}) = P(mp|MB(MP)) = \alpha P(MP|Parents(MP)) \times \prod_{Y_j \in Children(MP)} P(y_j|Parents(Y_j))$$

Equation 14: reduced probability calculation when using Markov Blanket

In the case of the authors' Bayesian network the Markov Blanket of the Math Performance node consists out of the Interest node and the English node. Now the total computation is reduced to:

$$P(mp|i,e) = \alpha P(mp|i) \times P(e|mp)$$

Both of these probabilities can be extracted directly from the Bayesian network. α is a normalizing factor that makes sure that all probabilities P(mp|i,e) sum up to one. It can be calculated by calculating P(mp|i,e) for all different values of mp, summing these probabilities and then take the reciprocal of the sum.

Calculating predictions using the Markov Blanket is much more efficient and can speed up exact and approximate algorithms. Because the authors only calculate one specific query, which uses all nodes, everything necessary for the improved calculation is always available. Only the Markov Blanket needs to be determined after the training phase to be able to use the improved query for predicting the student's performance.

3.4 Mental State Detection

Mental state detection is a more general form of affective state detection. Besides modeling emotion also other (mental) states like thinking or concentrating are being modeled.

3.4.1 Mind Reading Machine: Automated Inference of Cognitive Mental States from Video

R. El Kaliouby, P. Robinson, [15]

Abstract

Mind reading encompasses our ability to attribute mental states to others, and is essential for operating in a complex social environment. The goal in building mind reading machines is to enable computer technologies to understand and react to people's emotions and mental states. This paper describes a system for the automated inference of cognitive mental states from observed facial expressions and head gestures in video. The system is based on a multilevel dynamic Bayesian network classifier which models cognitive mental states as a number of interacting facial and head displays. Experimental results yield an average recognition rate of 87.4% for 6 mental states groups: agreement, concentrating, disagreement, interested, thinking and unsure. Real time performance, unobtrusiveness and lack of preprocessing make the authors' system particularly suitable for user-independent human computer interaction.

Comment

The authors have created a "mind reading" system. Using multiple techniques they infer the mental state of a person from video images. They start by tracking 24 points on the face, which are used for head pose estimation and facial feature extraction. For the head pose estimation pitch, yaw and roll are estimated. Estimates from consecutive frames are used to estimate action units from the FACS coding system for the head pose.
The facial features are extracted in a similar matter. For instance the shape of the mouth is determined using the points on the face. Using Color-based analysis the amount of teeth showing is determined. Finally the result is classified into three different categories: closed, jaw drop and mouth stretch.
From these classifications action units are again generated which are used in the next part of the recognition.

The facial and head actions determined earlier are used as input for Hidden Markov Model (HMM) classifier, which are used to identify facial expressions and head gestures. The expressions and gestures are modeled as a temporal sequence of the determined action units. For every possible expression and gesture a HMM model is modeled and trained. To select the most probable gesture or expression the forward-backward algorithm is used to determine the likelihoods that the models would have generated the sequence of action units. The likelihoods of all models are used as input for the dynamic Bayesian networks.

Dynamic Bayesian networks are used to infer the cognitive mental state. For every mental state a DBN has been defined and trained. As inputs the DBNs get the facial expressions and the head gestures from the respective HMMs.
These inputs and the previous recognized cognitive state are used to predict the current state. The DBNs consist out of several time slices to get better predictions.

The system is created in a modular fashion. To add more mental states to the system means adding HMMs for the necessary facial actions and head actions and adding a DBN for the inference of the mental state. Also because the FACS coding system is being used it is easy to change the lower level face tracking and feature estimation algorithms of necessary. As long as the new algorithms output Action Units the higher, more abstract algorithms do not have to be changed.

The system looks very user friendly. When trained, it is completely automated. Also because the system works with one camera it is completely non-invasive. A camera connected to the system only has to be pointed at a subject and the system does the rest without any user intervention. The authors are working on a prototype of what they call an "emotional hearing aid" for people with Asperger's Syndrome. The system should aid users in conversations with people to give them a clue about the other's mental state.

3.5 Empathic and Emotional Agents

The ability to recognize the affective state of a subject can be used in many different ways. Emphatic and emotional agents use this information to be able to display an appropriate emotional response to the subject. The messages from the agents are now given an emotional content and may have a bigger impact on the subject. This way the desired goal the agent is trying to accomplish may be reached faster and more effective.

3.5.1 Affective Advice Giving Dialogs

A. Cavalluzzi, V. Carofiglio, F. de Rosis, [16]

Abstract

In affective dialog simulation, recognition and interpretation of the affective state of the user should be integrated with display of empathy by the system and with dialog planning and execution. Cognitive models dealing with the inherent uncertainty of this interpretation are the method the authors propose to adopt. The authors describe how they integrated these models in an information-state approach to dialog modeling by illustrating, in particular, their application to a decision support system which is tailored to the 'state of change' of the user.

Comment

Nasty habits are hard to get rid of. Smoking and unhealthy eating patterns are examples of habits where help may be needed to successfully stop with these undesirable behavioral patterns. Interactions between a "patient" and an expert help the patient with changing his behavior.

The authors have worked on a computer-based advice-giving dialog system. This system should act like a computerized counselor and is capable of detecting the affective state of the patient. The system uses this information to adjust its dialog capabilities to "talk" to the patient with an appropriate emotional tone, i.e. the system could show sympathy towards the patient. The system uses this ability to persuade the patient to take a certain course of action.

The system uses the so called State Of Change Model; this model describes the transition states which a patient goes through when he tries to quit a bad habit. The model suggests signs of the different states and actions which can be used to promote the correct behavior. The system infers the state of change and the affective state by using two dynamic Bayesian networks and one static Bayesian network.

The authors have defined three cognitive and affective models:

1. Manipulation: this model calculates how the patient will respond on an agent's action. The model predicts the patient's emotional state. This model is implemented using a DBN.
2. Empathy: this model, similar to manipulation, describes the agent's emotions after an action of the patient. This model is also implemented using a DBN.
3. Interpretation: Models the patient's state of mind from the analysis of the patient's actions. This model is implemented using a static Bayesian network.

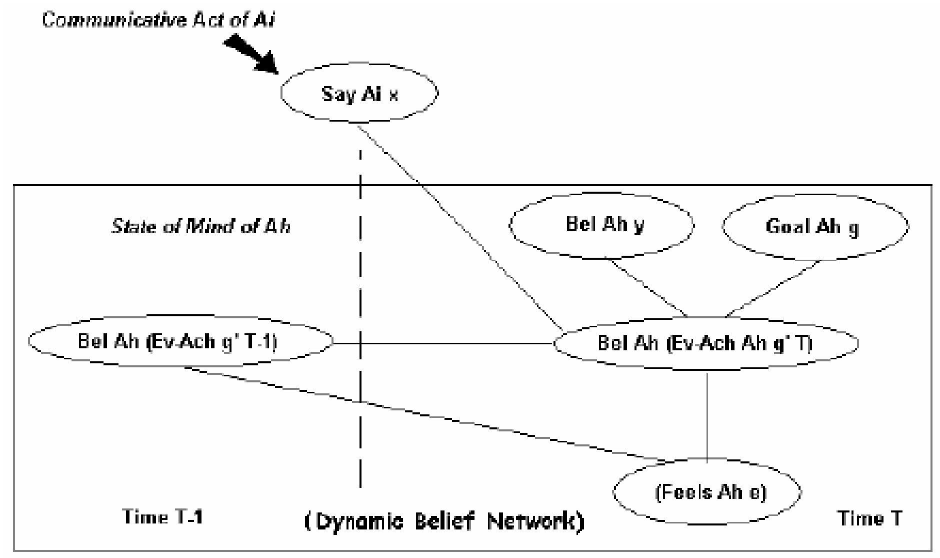


Figure 13: DBN used for the Manipulation and Empathy models

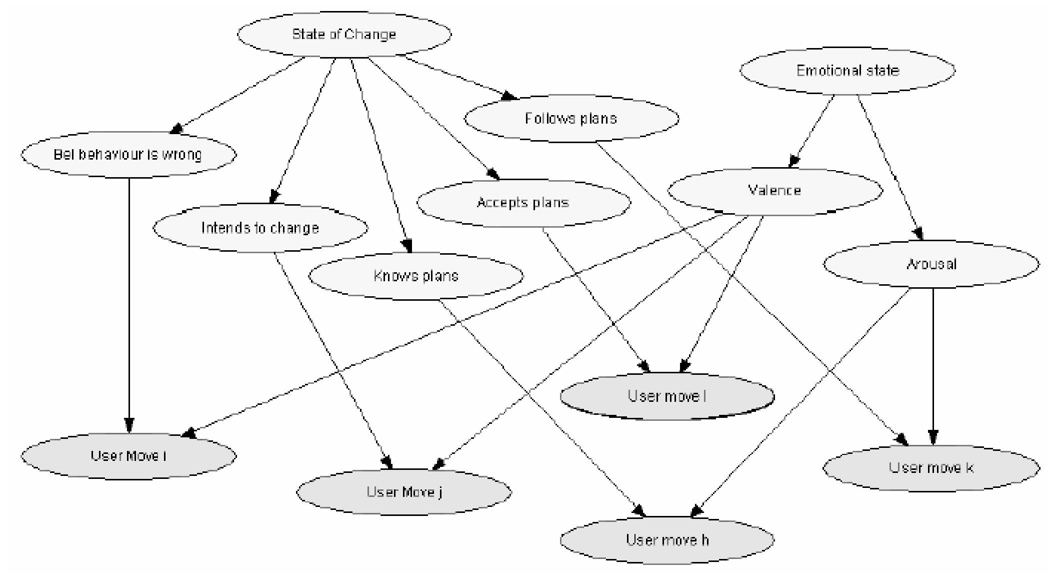


Figure 14: The static Bayesian network used for the interpretation model

The three different models are used to update the agent's and the patient's affective state and the patient's state of change. The current states are stored on the so called Information State Model. This model was developed as a method to formalize modular dialog simulation systems with a plan-based approach. The information state is a blackboard where all data necessary for the development of the dialog is stored. The data can be updated, tasks and plans for the dialog are represented by a logical formalism, select rules are used to determine the next plan and finally there is a control strategy to drive the activation of the rules.

To update the information state, the three models are run on the data in the blackboard. The authors have used Bayesian network API's developed by Hugin to implement the necessary (D)BNs.

The system manages the goals and plans for the dialog on a priority basis. Multiple plans can be used to accomplish a goal and plans can be linked causally, meaning that before plan B can be executed plan A has to be finished first. During the dialog plans can be changed to react on changes in the affective state of the patient.

The interaction between agent and patient is handled by an Interaction Manager. This module is responsible for handling the user interface. Currently the patient has select a sentence from a list which best represents what he would like to say. The agent is represented by a body model and text appears on screen or can be pronounced.

The concept of the system is interesting, but it completely depends on the willingness of the patient to talk truthfully to a computer. The current system uses a fixed dialog which the patient has to choose from. The authors are working on implementing a text parser, but they do not mention capturing the affective state using other sensors like a camera. One way the system could be useful is as an aid for counselors to give advice on actions they could suggest to the patient. The system itself looks solid and the implemented models originate from relevant psychological literature.

The authors do note that the current parameters for their BNs were determined subjectively. They are looking into gathering data from health services and to calibrate their system by using learning algorithms.

3.5.2 Physiologically Interactive Gaming with the 3D Agent Max

C. Becker, A. Nakasone, H. Prendinger, M. Ishizuka, I. Wachsmuth, [17]

Abstract

Physiologically interactive (or affective) gaming refers to research on the evocation and detection of emotion during game play. In this paper, the authors first describe the two building blocks of their approach to affective gaming. The building blocks correspond to two independently conducted research strands on affective human–computer interaction: one on an emotion simulation system for an expressive 3D humanoid agent called Max, which was designed at the University of Bielefeld; the other one on a real-time system for empathic (agent) feedback that is based on human emotional states derived from physiological information, and developed at the University of Tokyo and the National Institute of Informatics [19]. Then, the integration of both systems is motivated in the setting of a cards game called Skip-Bo that is played by a human game partner and Max. Physiological user information is used to enable empathic feedback through non-verbal behaviors of the humanoid agent Max. With regard to the new area of Conversational Informatics the authors discuss the measurement of human physiological activity in game interactions and non-verbal agent behavior.

Comment

To make games more interesting it may be useful to know the affective state of the player and to exploit it. This is the topic of the Affective Gaming research field. The authors have created an interactive game with a 3D agent, Max, which can detect the players affective state and can react on this state.
The game is divided into two parts, the Max Agent and the emotion recognition component. Both parts consist out of different layers, each responsible for a part of the operation of the game. The system architecture is shown in Figure 15:

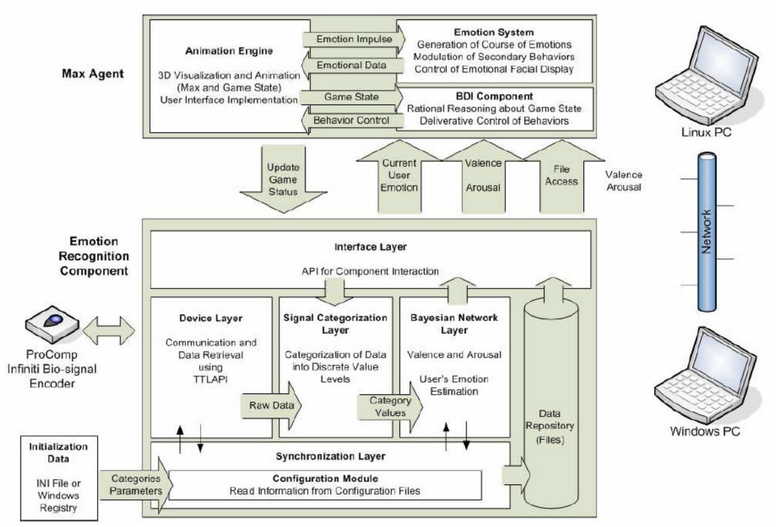

Max Agent
Animation Engine
3D Visualization and Animation (Max and Game State) User Interface Implementation
Emotion Impulse
Emotional Data
Game State
Behavior Control
Emotion System
Generation of Course of Emotions
Modulation of Secondary Behaviors
Control of Emotional Facial Display
BDI Component
Rational Reasoning about Game State
Deliverative Control of Behaviors
Linux PC
Update Game Status
Current User Emotion
Valence
Arousal
File Access
Valence Arousal
Network
Emotion Recognition Component
Interface Layer
API for Component Interaction
ProComp Infiniti Bio-signal Encoder
Device Layer
Communication and Data Retrieval using TTLAPI
Signal Categorization Layer
Categorization of Data into Discrete Value Levels
Bayesian Network Layer
Valence and Arousal
User's Emotion Estimation
Raw Data
Category Values
Data Repository (Files)
Windows PC
Initialization Data
INI File or Windows Registry
Categories Parameters
Synchronization Layer
Configuration Module
Read Information from Configuration Files

Figure 15: System architecture of the game

The system uses an external device that measures skin conductivity and electromyography to determine the player's affective state. The authors have used the valence-arousal model for modeling emotions. To infer the affective state a Bayesian network is used. This network is shown in Figure 16. What is interesting about this model is that its implementation differs from other models using the same valence-arousal model. The other models see the emotion variable as the hidden source of the valance and arousal, which in turn influence the measurements. This type of modeling is called causal modeling. The current model is an example of diagnostic modeling, here the measurements influence valence and arousal probabilities, which in turn influence the probabilities of the different possible emotions. The result is the same, but the probabilities in the different network represent totally different events.

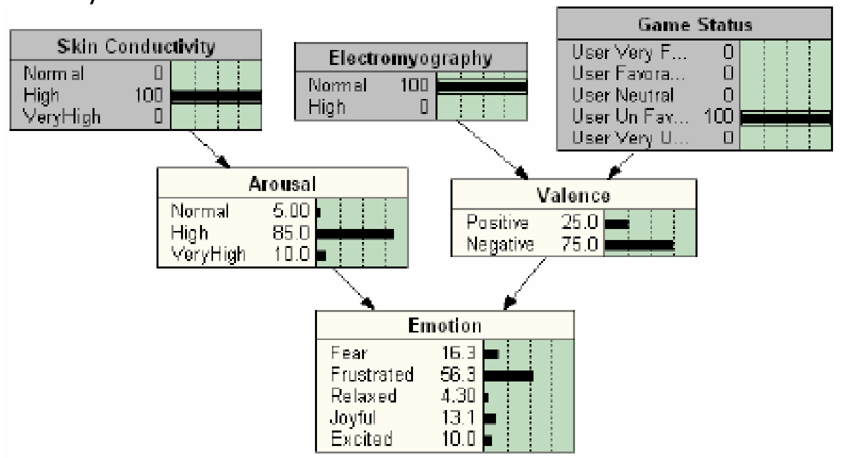


Figure 16: Bayesian network used for inference of the player's affective state

According to Russell and Norvig [4] diagnostic knowledge is often more fragile than causal knowledge. If the unconditional probabilities of the emotion node would suddenly change, it can be problematic to change the node's conditional probabilities P(emotion|arousal,valence) to accurately represent this change. A causal model on the other hand wouldn't have this problem because although the probabilities of the different emotions would change, every emotion still causes the same arousal and valence response. Meaning that the probabilities P(arousal|emotion) and P(valence|emotion) wouldn't change. Using a causal model may be a better choice because it is more robust than a diagnostic model. Diagnostic models depend more on the data collected by the sensors. If these probabilities change the whole result is influenced. Currently the authors have chosen the probabilities for the network themselves; no machine learning has been applied.

The authors have created a scenario to experiment with Max and a human player. They have implemented the game Skip-Bo, a card game, which Max and a human play against each other. The authors are preparing a study in which they want to evaluate a player's game experience. They want to experiment with four different conditions for Max:

1. Non-affective condition: Max doesn't show or reacts to emotions
2. Affective Self-emotional condition: Max only shows emotions evoked only by his own actions
3. Negative Emphatic condition: Max plays competitively and wants to win. When the user gets frustrated he will laugh at him.
4. Positive Emphatic condition: Max will act like a teacher and will encourage a player when the player is winning.

Until the experimental results are available the effectiveness of the system cannot be properly evaluated. The structure of the Bayesian network may have a negative influence on the result, but this totally depends on the dataset.

3.5.3 A Tool for Animated Agents in Network-Based Negotiation

M. Yuasa, Y. Yasumura, K. Nitta, [18]

Abstract

In this paper, the authors describe a tool for developing animated agents with facial expressions in negotiation through a computer network. The tool learns a user's tendency to select facial expressions of the animated agent, and generates facial expressions instead of human. In order to estimate facial expressions, the tool has an emotional model constructed by Bayesian Network. The authors can easily develop animated agents if they use this tool as a component. And they describe the estimation of an opponent's emotional state, based on observed data, by using the Bayesian Network.

Comment

The agent developed by the authors can be used for negotiations over a network. Two people can negotiate with each other by sending proposals and an animated facial expression. The agent has the possibility to take over negotiations from the user and send proposals and facial expressions instead of the user. The agent learns its emotional model by watching the user negotiate. With this model the agent can generate the unintentional facial expressions. These are the spontaneous expression the user shows when reading the opponents proposal. There are also intentional facial expressions; these are sent with the user's proposal. The agent's architecture can be divided into three sections (Figure 17): the user interface (actually outside the agent), the TAA (Tool for Animated Agent) and the agent application. The agent application implements the utility decision making part of the agent. It generates proposals and generates the next move and the intentional facial expression. The TAA consists out of a learning module, an emotional model and a module that generates the unintentional facial expressions. The learning module determines the probabilities for the emotional model, which is implemented as a Bayesian network.

The structure of the network was defined by the authors, see Figure 18. The probabilities were learned from experiments with eleven students. The emotional model used by the authors was based on the ABX model, developed by Newcomb. This model describes emotional states among two participants and a proposal [18]. Participants can feel positive, negative or intermediate about the other participant or the proposal. Using the ABX model the authors have defined nine different emotional states for the user, taking into account the user's view toward his opponent and the opponent's proposal. The different emotional states are shown in Figure 19, the arrows show the direction of the emotion. The figure shows how participant A feels toward participant B and proposal X.

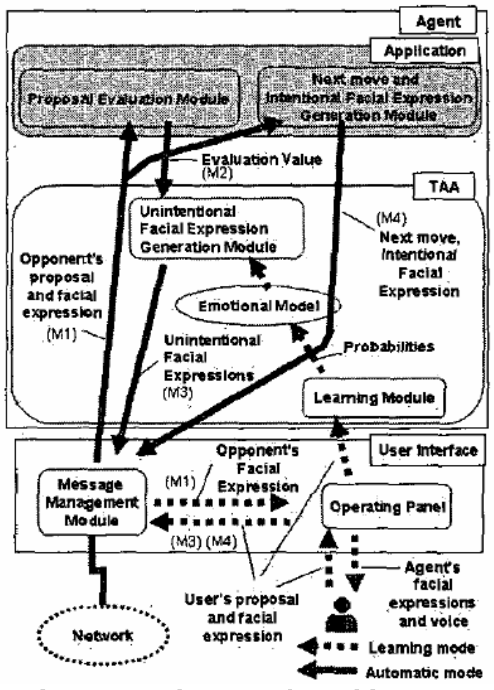

Agent
Application
Proposal Evaluation Module
Next move and Intentional Facial Expression Generation Module
Evaluation Value (M2)
TAA
Unintentional Facial Expression Generation Module
(M4) Next move, Intentional Facial Expression
Opponent's proposal and facial expression (M1)
Emotional Model
Unintentional Facial Expressions (M3)
Probabilities
Learning Module
User Interface
Opponent's Facial Expression
(M1)
Message Management Module
Operating Panel
(M3) (M4)
Agent's facial expressions and voice
User's proposal and facial expression
Network
Learning mode
Automatic mode

Figure 17: The agent's architecture

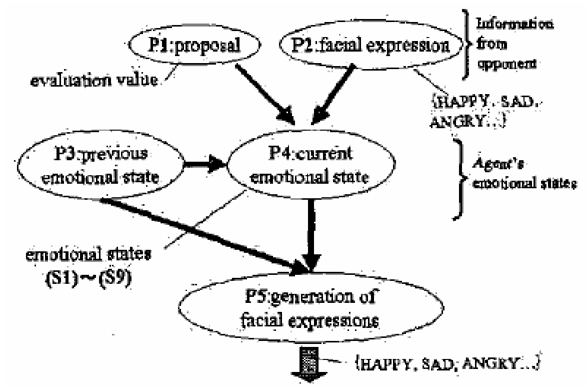

P1:proposal
P2:facial expression
Information from opponent
evaluation value
{HAPPY, SAD, ANGRY...}
P3:previous emotional state
P4:current emotional state
Agent's emotional states
emotional states (S1)~(S9)
P5:generation of facial expressions
{HAPPY, SAD, ANGRY...}

Figure 18: The Bayesian network for generation of facial expressions

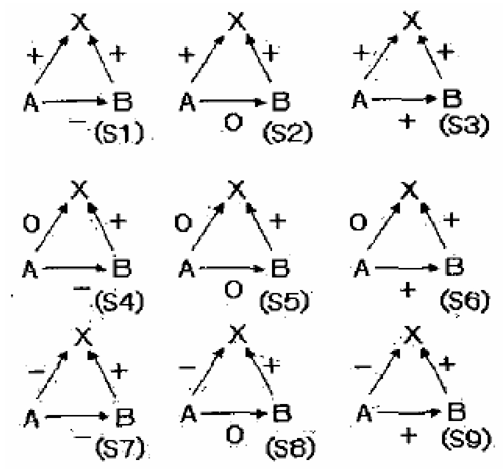


Figure 19: The nine possible emotional states

Using a similar Bayesian network the agent can also estimate the emotional state of the opponent. However the probabilities for this network must be estimated from the facial expression and the actions of this opponent.

The authors have done two different experiments: one to determine if the agent can accurately predict the user's emotional state and one to determine the opponent's emotional state. The first experiment showed a recognition rate of roughly 70% for recognizing the user's emotional state. This result was reached by taking the data from two students X and Y and dividing both datasets into a training set and a test set. The network was trained and tested two times giving four results. Two results from using test and training set from the same student and two results from using test set and training set from different students. The recognition rates from results where test and training set were from the same student were around 70% and the other result were 20% (Y-X) and 50% (X-Y).

From this result the authors conclude that the agent can recognize the user's emotional state. But this result also shows that the learned network is only valid for recognizing the user's emotional state and not for recognizing the emotional state of an opponent. Knowledge about the opponent is absolutely necessary to get an accurate prediction of the emotional state.

The authors have done an experiment to test the accuracy of the prediction of the opponent's emotional state. The authors have defined four different situations: The agent knows / doesn't know the opponent's Bayesian network and facial expressions from the opponent are / aren't used. In the case that the opponent's Bayesian network wasn't known a network generated by the average data of all eleven students was used for the prediction. The result of this experiment was that when the opponent's BN was known and facial expressions were used that the recognition rate was 70% in the other cases the recognition rate was between 40% and 50%. These results aren't very good, less than half of the time the agent will make a correct prediction. Sadly, unless the opponent's BN can be accurately predicted, a 45% prediction rate will be the best the agent can achieve. The authors have shown that the agent can predict the opponent's emotional state, but most of the time the prediction will be wrong.

The total system doesn't look very useful. Users have to choose their facial expression from only a few possibilities in the user interface. The authors have not implemented automatic facial recognition with cameras, so it's very easy to fool the opponent and the agent by selecting a facial expression that does not represent the user's true expression. If automatic recognition was implemented it would be a lot harder to fool the agent and the opponent.
Another drawback is the poor recognition rates for the emotional state of the opponent. This is the function that makes the agent actually interesting and the disappointing results make the agent less useful. The agent's ability to take over negotiations is interesting, but when something important is being negotiated the user will probably want to handle the negotiations himself.

3.6 Cognitive Workload Detection

People only have a limited amount of resources available for solving a problem. The available time and attention may vary during the process. If a system can detect if the user is "busy" it may suggest other methods to speedup the progress or to lighten the load.

3.6.1 Making Systems Sensitive to the User's Time and Working Memory Constraints

A. Jameson, R. Schäfer, T. Weis, A. Berthold, T. Weyrath, [19]

Abstract

Recent advances in user modeling technology have brought within reach the goal of having systems adapt to temporary limitations of the user's available time and working memory capacity. The authors first summarize empirical research conducted by themselves and others that sheds light on the causes and consequences of these (continually changing) resource limitations. They then present a decision-theoretic approach that allows a system to assess a user's resource limitations and to adapt its behavior accordingly. This approach is illustrated with reference to the performance of the prototype assistance system READY.

Comment

The authors have created a prototype assistance system for fixing cars. Fixing cars is just a scenario the authors chose to implement for the system; it can be used for other tasks. The author's goal was to model the time and working memory constraints of the user of the prototype system. When people run out of time, their performance will drop to get the job done in time. The READY system is designed to detect the user's time limitation and change the way it gives advice to speed up the dialog. The system also models the user's working memory capacity. When a task is performed by the user, the system reduces the amount of expected working memory. When the user starts to perform more tasks simultaneously the amount of available memory may no longer be sufficient and the user's overall performance may start to decrease.
The system needs to detect these events and change its dialog with the user to counter the negative effects.

READY's current architecture is as following:

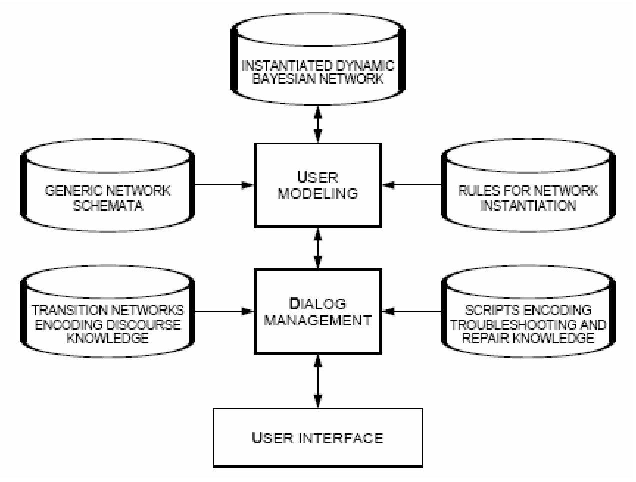

INSTANTIATED DYNAMIC BAYESIAN NETWORK
GENERIC NETWORK SCHEMATA
USER MODELING
RULES FOR NETWORK INSTANTIATION
TRANSITION NETWORKS ENCODING DISCOURSE KNOWLEDGE
DIALOG MANAGEMENT
SCRIPTS ENCODING TROUBLESHOOTING AND REPAIR KNOWLEDGE
USER INTERFACE

Figure 20: READY's architecture

Spoken utterances from the user are input into the system through a user interface where they can be composed from standard utterances and some style aspects like length of pauses in the speech can be added as extra information. The utterances are then processed so that the dialog management module can update the user model and determine the systems next response to the user. It has access to domain specific knowledge, in this case automobile knowledge, which it uses to determine the next action. The possible dialog response and repair actions are sent to the user model. The model decides using the current cognitive workload of the user what the best response and action is.

The cognitive workload is modeled by a dynamic Bayesian network. Every time the user or the system performs a dialog act or an action, time slices are added to the network.

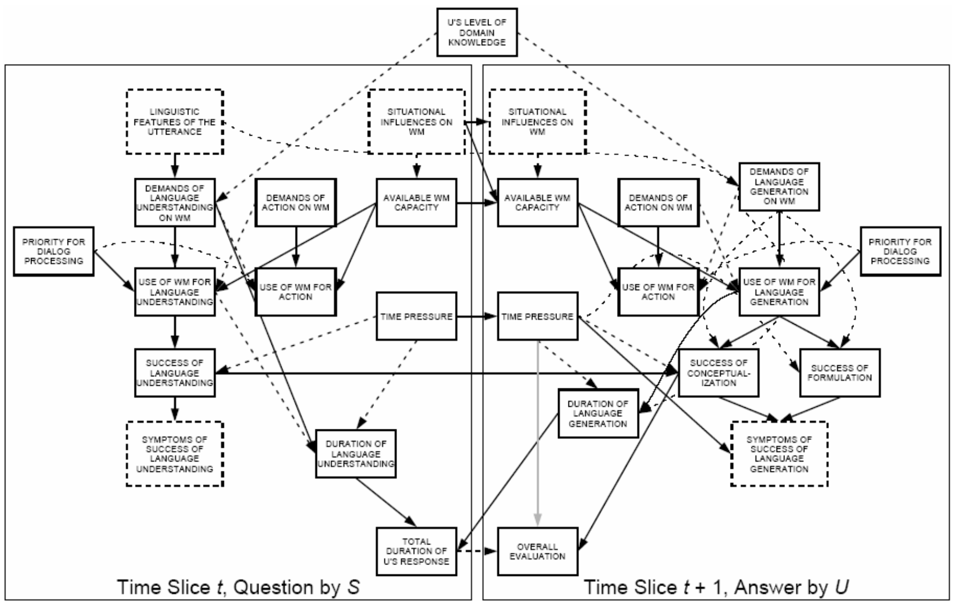

U'S LEVEL OF DOMAIN KNOWLEDGE
LINGUISTIC FEATURES OF THE UTTERANCE
SITUATIONAL INFLUENCES ON WM
SITUATIONAL INFLUENCES ON WM
DEMANDS OF LANGUAGE GENERATION ON WM
DEMANDS OF LANGUAGE UNDERSTANDING ON WM
DEMANDS OF ACTION ON WM
AVAILABLE WM CAPACITY
AVAILABLE WM CAPACITY
DEMANDS OF ACTION ON WM
PRIORITY FOR DIALOG PROCESSING
PRIORITY FOR DIALOG PROCESSING
USE OF WM FOR LANGUAGE UNDERSTANDING
USE OF WM FOR ACTION
USE OF WM FOR ACTION
USE OF WM FOR LANGUAGE GENERATION
TIME PRESSURE
TIME PRESSURE
SUCCESS OF LANGUAGE UNDERSTANDING
SUCCESS OF CONCEPTUAL-IZATION
SUCCESS OF FORMULATION
DURATION OF LANGUAGE GENERATION
SYMPTOMS OF SUCCESS OF LANGUAGE UNDERSTANDING
DURATION OF LANGUAGE UNDERSTANDING
SYMPTOMS OF SUCCESS OF LANGUAGE GENERATION
TOTAL DURATION OF U'S RESPONSE
OVERALL EVALUATION
Time Slice t, Question by S
Time Slice t + 1, Answer by U

Figure 21: READY's DBN

When the user model has received a new possible system utterance, the model adds two new time slices to the DBN. One represents the system's utterance and the other represents the modeling of the user's response on this utterance. The network with the two new time slices is evaluated with a standard inference algorithm to compute the new probability distributions for the new time slices. These are evaluated and weighted in the overall evaluation node.

To select the best possible system response the authors have tried two different approaches. The first was to evaluate every possible relevant system "move" and calculate the evaluation node for every possible situation. This was done by calculating a situation, then removing the two new time slices and to repeat the original process with two other time slices. Once every situation was calculated, the best action was chosen and performed by the system. The other approach used an influence diagram instead of a DBN.

Generally speaking influence diagrams are DBNs but with utility nodes.
Influence diagrams can reduce the search space by changing a few nodes that describe aspects of the utterance into decision nodes. The overall performance node is changed into a value node. Special algorithms for influence diagrams can be used to determine optimal values for the decision nodes given the overall performance node. System action where these nodes do not match can be excluded from the search and this accelerates the action determination process.

The authors note that at the moment of writing the paper the system is not very fast. They attribute this to the size of the DBN or influence diagram. They feel that "it is one or two orders of magnitude too slow for practical application".

The READY system looks quite solid. It has a clear architecture that can be used for many different applications. The dynamic Bayesian network that the system uses also looks good. The fact that they expand the network as the dialog continues is interesting and looks promising. The model itself might need a little more testing and validation. The authors have mentioned that parts of their system, i.e. recognizing the time constraints of a user, have never been tried before. They may need a few more empirical studies to get a better insight into the model's performance.

The natural language input isn't automated; the user utterances still have to be entered using a user interface. Perhaps a context sensitive automated speech recognizer (ASR) can be used to completely automate the system. Context sensitivity is important here to improve the recognition rate of the ASR. This can be realized by having a specialized corpus for the application containing only relevant words and phrases.

3.7 Facial Recognition

To recognize emotions one needs sensors that detect the evidence for these emotions. The emotions themselves are hidden; only the influence they have on the body can be measured. The face tells a lot about a person's affective state. The field of facial recognition tries to infer emotions or other mental states from the features of the human face.

3.7.1 Automatic Recognition of Facial Expressions Using Bayesian Belief Networks

D. Datcu, L.J.M. Rothkrantz, [20]

Abstract

The current paper addresses the aspects related to the development of an automatic probabilistic recognition system for facial expressions in video streams. The face analysis component integrates an eye tracking mechanism based on Kalman filter. The visual feature detection includes PCA oriented recognition for ranking the activity in certain facial areas. The description of the facial expressions is given according to sets of atomic Action Units (AU) from the Facial Action Coding System (FACS). The base for the expression recognition engine is supported through a BBN model that also handles the time behavior of the visual features.

Comment

The authors have created a system that can automatically recognize facial expressions. For the recognition process a Bayesian network has been used.
They use a Kalman filter like mechanism that tracks the movement of the eyes of the subject. The position of the eyes is used as a basis to acquire the other features points of the face. In total the system collects 30 different feature points. The system performs some prepossessing to always get feature points from a frontal view of the face. In total 15 feature values are calculated from these feature points. These values are input into the Bayesian Network for further classification.

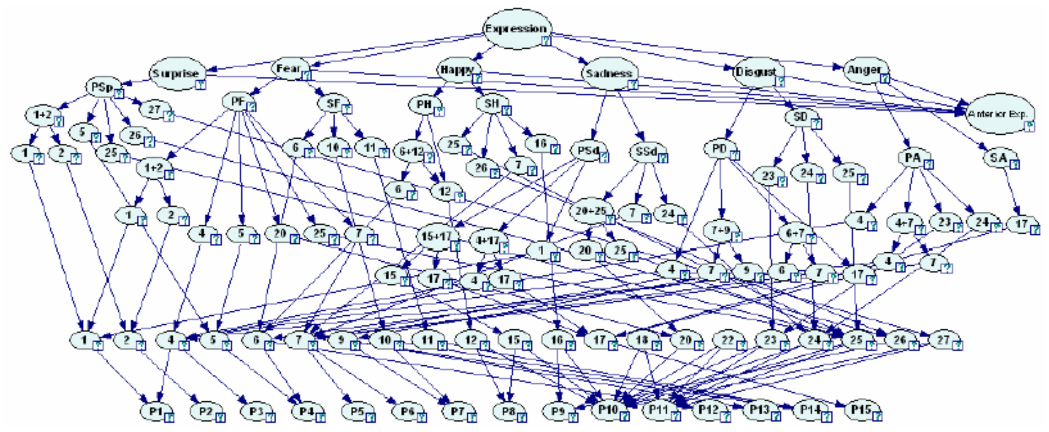


Figure 22: Implemented Bayesian network

The 15 values are quantified and the nodes P1 till P15 are set respective to the quantified values. After these nodes are given a value, the inference algorithm can be run to get the prediction percentages for the different facial expressions. The authors have used the Facial Action Coding System. This system uses action units to code certain facial features. Several action units combined give a facial expression. The other layers of the BN consist out of these AUs. At the top of the network the expression nodes are situated. Like other causal models, the idea is that the facial expressions cause the associated action units to appear and these AUs again cause the 15 measurable values to appear in the pattern corresponding to the emotion.

The system developed by the authors looks good. It doesn't look very complicated and should be easy to work with. The implementation is very straightforward and adding more facial expression to the system should not be very hard. Currently the authors have implemented their model with a static Bayesian network. To model the fact that expression develop over multiple frames they have added a node that has the previous predicted expression; the anterior expression node. Perhaps a dynamic Bayesian network can be implemented to better model the temporal dependencies between expressions, action units and feature points. This may have a negative effect on the computational complexity and the real-time performance of the system, but experiments will have to be performed to see the actual effect on system performance.

3.8 Natural Communication

When humans talk to each other a lot more communication is taking place then just the spoken words that are exchanged. Small gestures contribute to the communication process. The listener can show that he is in fact listening by looking at the sender and nodding his head, etc. To improve human-robot communication these kinds of signals can be implemented to achieve natural communication.

3.8.1 Establishing Natural Communication Environment between a Human and a Listener Robot

Y. Ogasawara, M. Okamoto, Y.I. Nakano, T. Nishida, [21]

Abstract

The progress of technology makes familiar artifacts more complicated than before. Therefore, establishing natural communication with artifacts becomes necessary in order to use such complicated artifacts effectively. The authors believe that it is effective to apply human natural communication manner between a listener and a speaker to human-robot communication. The purpose of this paper is to propose the method of establishing communication environment between a human and a listener robot. In method proposed by the authors, the common intention is formed by joint attention and redundancy of behavior.

Comment

The authors have been working on a method to establish a natural communication environment between a human and a listener robot.
They have noticed that without appropriate feedback messages become very artificial, i.e. the sender doesn't feel very comfortable when speaking to a camera. The authors propose a listening robot that mimics normal human behavior to make the user feel more comfortable when speaking to the robot's camera.

Central to their approach is the idea they call "User Involvement"; they aim to increase the user involvement in the communication progress. To achieve this they want to establish what they call joint attention, meaning that both human and robot are focusing on the same object. The authors have stated the following requirements:

- Cognitive/Communicative reality should be achieved: The user should feel the virtual object/world, or the human-to-computer interaction as "real".
- Two (or more) cognitive spaces should be linked: The user should move in and out smoothly at least two cognitive spaces such as his/her viewpoint (here) and what he/she sees (there).

Humans connect cognitive spaces by communicating verbally or nonverbally. Examples are making eye contact, matching actions and reactions, joint attention, etc. The authors focus on joint attention to establish the communication environment.

The authors have created a model for a speaker-listener communication environment. Through their behavior the speaker and the listener form the common intention.

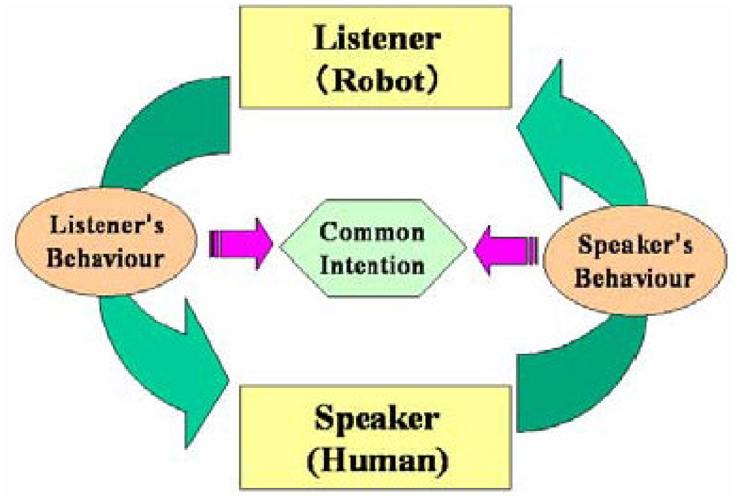


Figure 23: Speaker-Listener communication environment model

With the common intention the authors mean what is mutually aimed both by the speaker and the listener.

The authors have analyzed speaker-listener communication: they have looked at relevant psychological studies and have analyzed videos of explanatory scenes they have captured. According to a psychological study joint attention can be divided into three different types:

- Check attention
- Follow attention
- Direct attention

From their analysis of the videos of explanatory scenes they recorded, the authors conclude that humans do exert behavior to establish joint attention.
They have found that when performing an explanatory task the speaker exerts different types of attention behaviors and has different communication modes to reach a certain goal.

The authors have listed the requirements for a listener robot.

- It should be able to establish joint attention
- Attention behavior should be redundant:
    a. Multiple modalities should be implemented for attention behavior.
    b. Attention behavior should be repetitive to get a better result.
- The robot should be able to recognize and react accordingly to the following communication modes:
    a. Talking-to mode
    b. Talking-about mode
    c. Confirming mode
    d. Busy mode

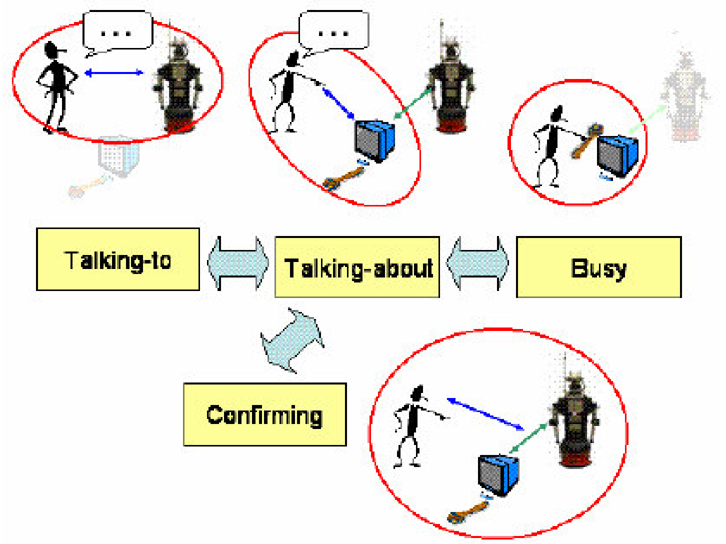


Figure 24: The communication modes

In the talking-to mode the speaker is talking to the robot and the robot should look at the speaker. When the speaker is in the talking about mode the speaker's attention is focused on an object, the robot should also focus its attention on the object. When the speaker is in the confirming mode, he wants to check if the robot is paying attention to the speaker or the object the speaker is talking about. When the speaker looks at the robot it should give some sort of confirmation signal to show that it is paying attention. A speaker is in the busy mode when he is not talking to the robot but for instance is operating the object. In this situation the robot should keep looking at the object.

The authors have constructed the listener robot by using Robovie as a platform. Robovie is a humanoid robot which has movable arms, hands, head and eyes. It has a motion caption system and a microphone for audiovisual capture possibilities. The robots architecture is as following:

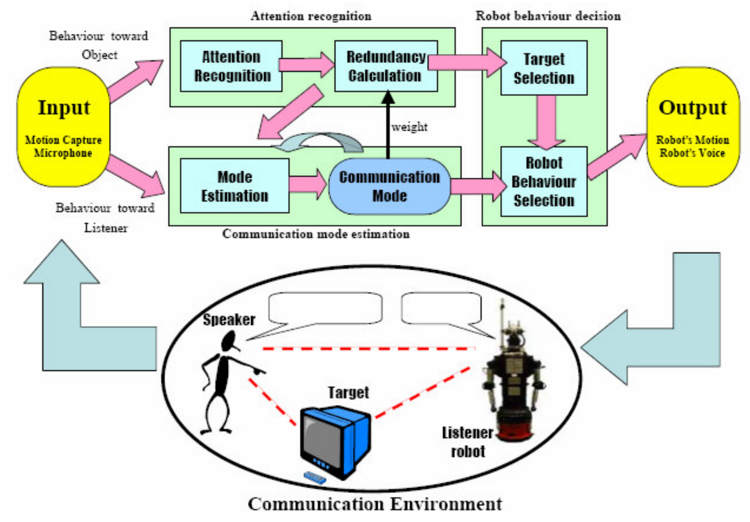


Figure 25: Listener robot architecture

The inputs from the motion capture system and the microphone are analyzed and the speaker's attention behavior and communication mode are estimated. The estimations are used to decide how the robot should act.

The attention recognition module and the communication mode estimator are implemented using Bayesian networks. The structures of the networks were constructed by using the captured video observations mentioned earlier.

The authors have chosen to implement recognition of the following attention getting behaviors:

- Eye gaze (head direction)
- Pointing
- Grasping
- Repetitive hand gestures
- Physical relationship with objects (distance, body direction)

Bayesian networks for two of these behaviors are shown in Figure 26:

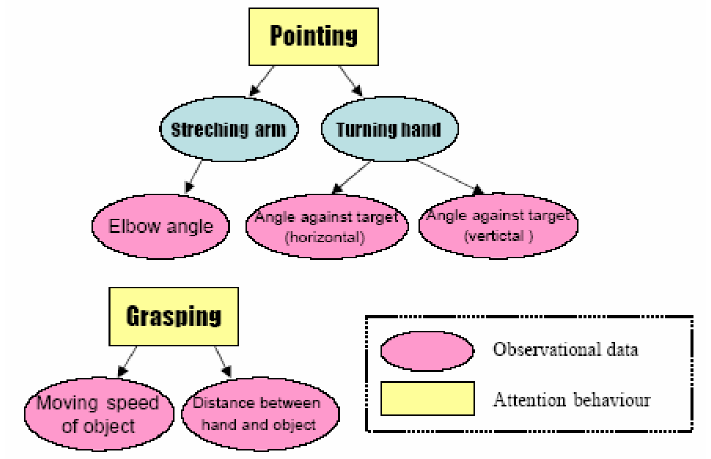


Figure 26: Bayesian networks for attention recognition

The communication mode estimation is done with the following network:

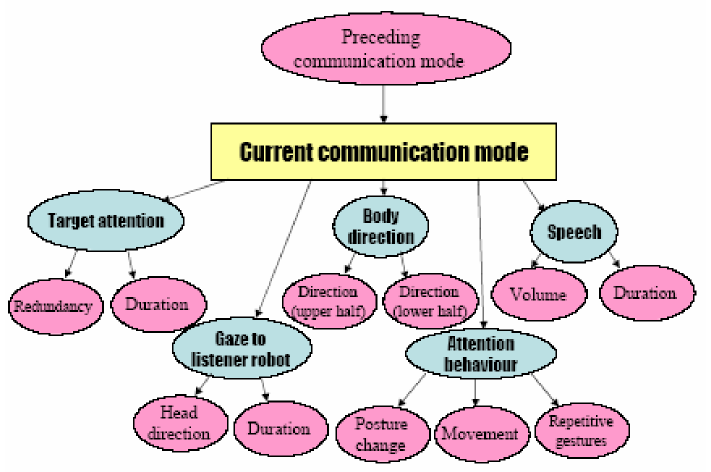


Figure 27: BN for Communication mode estimation

The method proposed by the authors is interesting. Humans have a natural tendency to act against object as if they are human, a robot that actually acts as a human might really improve the way the human is communication with or through the robot. The authors see using the robot as an improved camera as a major application of the listener robot. They think that the creation of explanatory video material will become easier because the presenter will feel more relaxed due to the listener robot's behavior.

The authors do not mention any results from experiments with the system, so the performance of the different BNs is not known. They may need more tweaking. They were based on result from recordings from the authors, but only two subjects were taped. This may have an influence on the performance of the BNs. Larger empirical studies might be needed to get better models.

4 Conclusions and Recommendations

4.1 Conclusions

Some concluding remarks can be written about the affective computing field, taking into account the results from the literature survey. These conclusions are followed by a topic matrix which describes some important features of the reviewed papers.

Field Overview

In its essence the field of affective computing is a mix of psychology, physiology and pattern recognition. To recognize emotions or other internal human states researchers use different theories from psychology and physiology and try to implement these theories into intelligent agents. Usually this boils down to a software agent that tries to recognize input patterns from its sensors to infer emotions or mental states that, according to the theories, should be occurring in the subject. The sensors used by the agents are often cameras for facial recognition, microphones for speech analysis and various sensors that measure physiological signals from the human body. Examples of these signals are heart rate, breathing rate and brain activity.

The general aim of the research is to make the software agents more intelligent and more natural towards humans by providing the agents with more information. How this information is used depends on the application. Sometimes acquiring the affective information is one of the main goals of a system [6][15] and sometimes it only serves as a tool to reach the system's intended result [12][13].

The research field has a potential wide range of applications. The main requirement for the implementation of affective computing techniques is that humans have to interact with machines/computers, so the results from this research can potentially be applied almost everywhere.

Field Maturity

Affective computing research started in the mid 90s with the work done by the MIT media lab. Recently, the amount of research groups doing affective computing related research is starting to increase. Most of the papers reviewed are quite recent, 13 of the 17 papers are from 2003-2005. Currently a lot of the reviewed groups are developing mathematical frameworks and are conducting empirical studies to create accurate models for their affective state recognition projects. Although there has been quite some progress the field hasn't matured much in the 10 years of its existence. Due to the availability of more powerful computers, truly complex systems only now have become feasible for actual implementation, but the biggest factor that is slowing down the field is the psychological side of the research. Since the spectrum of emotions is very diverse and the expression of emotions differs per culture, geographical location and many other factors, it is very hard or perhaps even impossible to have a single model for accurate emotion prediction. Research Groups are using different emotional models, developed by psychologists, which fit their application or are creating their own models based on experimental data.

Problems

The research groups have encountered three major problems:

- Algorithmic complexity of pattern recognition.
- Lack of sufficient and accurate data for training pattern recognizers.
- Lack of a general emotional model.

The third problem has been discussed in the previous section. The other two are problems common to pattern recognition applications. For this literature survey one of the constraints for the papers was that Bayesian networks should be used for the inference of the affective state. The training and inference procedures used for Bayesian networks are very computationally expensive. Solutions exist to speed up computation but this generally means trading accuracy for speed. Using approximate algorithms is one example to speed up inference, using a less complex emotional model is an example to speed up inference and training.

The lack of sufficient data is a large problem. Although the CPT entries of a Bayesian network can be obtained by using expert knowledge, if the network size increases the amount of necessary CPT entries becomes so large that the expert knowledge approach isn't feasible anymore. From that point on machine learning is the only option and this approach requires a lot of data to have satisfactory results. The problem with obtaining data for emotion recognition is that emotions are "hidden" events, they cannot be measured directly. Indirect measurements like the heart rate or facial expressions are needed. The only way to really know in which emotional state a person is, is to ask him. Acquiring data is a very time consuming procedure. One needs to measure the "indirect" signals and at the same time, or afterwards, ask the subject in which emotional state he thinks he is. This procedure can lead to very subjective data, which could have an influence on the recognition rate of the system. Some papers have used results from psychological research; this speeds up the process because they can use the data and the conclusions from this research.

Most Promising Work

From the 17 reviewed papers the most promising was "Mind Reading Machines: Automated Inference of Cognitive Mental States from Video" [15]. The authors have created a completely automated mental state inference system that can detect a subject's mental state using a video stream. The system, based on the Facial Action Coding System, uses a combination of facial recognition algorithms, hidden Markov Models and Dynamic Bayesian networks to convert the video stream into a prediction for the mental state. The system runs in real time and is totally unobtrusive so it can be used with causing the subject any discomfort. The authors propose an "emotional hearing aid" for people diagnosed with Asperger's Syndrome. This aid should give these people insight in the mental state of people they are communicating with.

Topic matrix

To give an overview of the reviewed papers a topic matrix has been created. The papers have been classified using a few features:

- Is the paper theoretical or practical?
- Did the authors develop a model or an application?
- Is it a detailed paper is does it only give an overview?
- Does the paper show any experimental results?
- What is the difficulty rating / quality of the paper?

The papers are presented in the same order as in the literature survey; they are grouped by topic area and are represented by their respective section number.

Table 3: Topic Matrix

Papers	Theoretical	Practical	Develop Model	Develop Application	Overview	Detailed	Results	Difficulty rating
3.1.1		x	x			x	yes	++++
3.1.2		x	x			x	yes	+++
3.2.1	x		x		x		no	+++
3.2.2		x		x		x	no	+++
3.2.3		x		x	x		yes	++
3.2.4	x		x			x	yes	+++
3.3.1		x		x		x	no	+++
3.3.2	x		x		x		no	+++
3.3.3	x		x			x	yes	+++
3.3.4	x		x			x	yes	+++
3.4.1		x		x		x	yes	+++++
3.5.1	x		x			x	no	+++
3.5.2		x		x	x		no	++
3.5.3		x		x		x	yes	+
3.6.1		x		x		x	no	+++
3.7.1		x		x		x	no	+++
3.8.1	x		x			x	no	++++

4.2 Recommendations

After examining the different papers regarding affective state detection there are a few recommendations that should be considered when designing a similar system.

Multimodality improves recognition results

Generally, adding more modalities to the system will give better recognition results. The extra information can make a system more redundant and less sensitive to sensor errors and anomalies. However, one paper [6] states that one should not have too many sensors, because using sensors does have a cost. Eventually too many different sensors could have a negative effect on system performance.

Keep model complexity in mind

Inference in Bayesian networks is very hard. Using exact inference algorithms on large networks can take so long that a system is no longer effective, especially in situations where real-time or close to real-time performance is necessary. Using approximate inference algorithms and/or reducing the model complexity may be necessary to achieve the required performance level. It all depends on the requirements for the application that has to be realized. So when necessary, remember Ockham's razor.

Use large amounts of data for model development

The quality of a system using probabilistic inference depends on the quality of expert knowledge and training data. When the models get larger, it will become increasingly difficult for an expert to choose the probabilities for the CPTs. When taking the machine learning route, which becomes inevitable when the Bayesian networks grow, a lot of data will be necessary to have a good recognition rate. The advice is to acquire as much data as possible to be able to learn the best possible model structure and CPT entries. Log sensor readings, let subject fill in questionnaires, videotape and analyze experiments, anything that could be usable data should be used. Emotions cannot be measured directly so finding a lot of indirect emotion signals is really necessary to get a good result.

Make the system as unobtrusive as possible

To detect emotions sensors are needed, the literature survey has shown that many different sensors can be used to infer emotions. Cameras are used for facial recognition, a person's heart rate, breathing rate or some other physiological signal can be measured. All can be used as sensor input. But when measuring physiological signals, sensors will have to be attached to the human body. This may not be very practical in real-world systems, where it isn't always an option to attach a dozen sensors to someone body to measure his emotion. Camera based systems or systems that analyze the human voice or human gestures do not need to be attached and cause the subject no discomfort. When creating an affective state detection system for use in the "real world" it is advised to use sensors that are unobtrusive.

5 Bibliography

[1] R.W. Picard, “Affective Computing”, in *M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 321*, 1995, http://vismod.media.mit.edu/tech-reports/TR-321.pdf.

[2] R.W. Picard , ”Affective Computing for HCI”, in *Proceedings of HCI International (the 8th International Conference on Human-Computer Interaction) on Human-Computer Interaction: Ergonomics and User Interfaces-Volume I - Volume I*, pg 829-833, 1999, http://affect.media.mit.edu/pdfs/99.picard-hci.pdf.

[3] C. Conati, X. Zhou , “A Probabilistic Framework for Recognizing and Affecting Emotions”, in *Proceedings of the AAAI 2004 Spring Symposium on Architectures for Modeling Emotions*, pg 13-16, 2004, http://www.cs.ubc.ca/~conati/my-papers/SS204ConatiC.pdf.

[4] S. Russell, P. Norvig, “Artificial Intelligence: A Modern Approach”, *Prentice Hall,* New Jersey, pg 462-580, 2003.

[5] F.M. Dekking, C. Kraaikamp, H.P. Lopuhaä, L.E. Meester, “Kanstat: Probability and Statistics for the 21st Century”, *Delft University of Technology*, Delft, pg 11-115, 2004.

[6] X. Li, Q. Ji, “Active Affective State Detection and User Assistance With Dynamic Bayesian Networks”, in *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, Vol. 35, No. 1, January 2005*, pg 93-105, 2005, http://ieeexplore.ieee.org/iel5/3468/29969/01369348.pdf?arnumber=1369348.

[7] Q. Ji, P. Lan, C. Looney, “A Probabilistic Framework for Modeling and Real-Time Monitoring Human Fatigue”, *Rensselaer Polytechnic Institute*, Troy, 2003, http://www.ecse.rpi.edu/homepages/qji/Fatigue/smc_paper.pdf.

[8] G. Ball, J. Breese, “Modeling the Emotional State of Computer Users”, in *Workshop on Attitude, Personality and Emotions in User-Adapted Interaction, International Conference on User Modeling*, 1999, http://www.mmi.ethz.ch/zim/emopapers/ball-Modeling_the_Emotional_State_of_Computer_Users.pdf.

[9] E. Horvitz, T. Paek, “Harnessing Models of Users’ Goals to Mediate Clarification Dialog in Spoken Language Systems”, in *Proceedings of the 8th International Conference on User Modeling 2001*, pg 3-13, 2001, http://www.springerlink.com/(3sn0n545qvtj1bq2b5nh5545)/app/home/contribution.asp?referrer=parent&backto=issue,1,54;journal,1680,2312;linkingpublicationresults,1:105633,1.

[10] K. Sakai, M. Masaaki, K. Yokoyama, “Modeling Patient Responses to Surgical Procedures during Endoscopic Sinus Surgery using Local Anesthesia”, in *2004 IEEE International Conference on Systems, Man and Cybernetics Volume 1*, pg 364-369, 2004, http://ieeexplore.ieee.org/iel5/9622/30409/01398324.pdf?tp=&arnumber=1398324&isnumber=30409.

[11] K. Crews Baumgartner, S. Ferrari, C.G. Salfati, “Bayesian Network Modeling of Offender Behavior for Criminal Profiling”, *Duke University*, 2005, http://fred.mems.duke.edu/projects/crime/Papers/CPfinal_truely.pdf.

[12] R.C. Murray, K. VanLehn, "DT Tutor: A Decision-Theoretic, Dynamic Approach for Optimal Selection of Tutorial Actions", in *Lecture Notes in Computer Science vol. 1839/2000*, pg 153-162, 2000, http://www.springerlink.com/media/4h83eykk4k7jvl586p4u/contributions/j/t/0/x/jt0xppp564nbvklu.pdf.

[13] W. Bosma, E. André , "Exploiting Emotions to Disambiguate Dialogue Acts", in *Proceedings of the 9th international conference on Intelligent user interface*, pg 85-92, 2004, http://delivery.acm.org/10.1145/970000/964459/p85-bosma.pdf?key1=964459&key2=7624883311&coll=GUIDE&dl=ACM&CFID=11111111&CFTOKEN=2222222.

[14] R. Bekele, W. Menzel , "A Bayesian Approach to Predict Performance of a Student (BAPPS): A Case with Ethiopian Students", in *Proceedings of the international conference on Artificial Intelligence and Applications*, pg 191-198, 2005, http://nats-www.informatik.uni-hamburg.de/~wolfgang/papers/aia2005.pdf.

[15] R. El Kaliouby, P. Robinson, "Mind Reading Machines: Automated Inference of Cognitive Mental States from Video", in *2004 IEEE International Conference on Systems, Man and Cybernetics, vol. 1*, pg 682-688, 2004, http://ieeexplore.ieee.org/iel5/9622/30409/01398380.pdf?arnumber=1398380.

[16] A. Cavalluzzi, V. Carofiglio, F. de Rosis, "Affective Advice Giving Dialogs", in *Lecture Notes in Computer Science vol. 3068/2004*, pg 77-88, 2004, http://www.springerlink.com/media/np33vnluqj6q6d62eaak/contributions/4/m/d/y/4mdyq0q3eknefnhn.pdf.

[17] C. Becker, A. Nakasone, H. Prendinger, M. Ishizuka, I. Wachsmuth, "Physiologically Interactive Gaming with the 3D Agent Max", in *2005 International Workshop on Conversational Informatics*, 2005, http://www.ii.ist.i.kyoto-u.ac.jp/jsai2005ws/proceedings/becker.pdf.

[18] M. Yuasa, Y. Yasumura, K. Nitta, "A Tool for Animated Agents in Network-Based Negotiation", in *Proceedings of The 12th IEEE International Workshop on Robot and Human Interactive Communication*, pg 259-264, 2003, http://ieeexplore.ieee.org/iel5/8869/28022/01251855.pdf.

[19] A. Jameson, R. Schäfer, T. Weis, A. Berthold, T. Weyrath, "Making Systems Sensitive to the User's Time and Working Memory Constraints", in *Proceedings of the 4th international conference on Intelligent user interfaces*, pg 79-86, 1998, http://delivery.acm.org/10.1145/300000/291094/p79-jameson.pdf?key1=291094&key2=0895883311&coll=GUIDE&dl=GUIDE&CFID=59435643&CFTOKEN=49261603.

[20] D. Datcu, L.J.M. Rothkrantz, "Automatic recognition of facial expressions using Bayesian Belief Networks", in *2004 IEEE International Conference on Systems, Man and Cybernetics Volume 3*, pg 2209-2214, 2004, http://mmi.tudelft.nl/pub/dragos/Automatic%20recognition%20of%20facial%20expressions%20using%20BBN.pdf.

[21] Y. Ogasawara, M. Okamoto, Y.I. Nakano, T. Nishida, "Establishing Natural Communication Environment between a Human and a Listener Robot", in *AISB 2005 Symposium: Conversational Informatics for Supporting Social Intelligence & Interaction*, 2005, http://kaiwa.ristex.jst.go.jp/AISB_CI/papers/6_AISB05_ogasawara_final.pdf.