A Computational Architecture to Model Human Emotions

Arun Chandra

International Business Machines Corporation, 11400 Burnet Rd, Austin, TX 78758

Abstract

In this paper a computational architecture to model human emotions is described and developed. In previous work of brain modeling there has been significant effort to model the brain's analytical and learning capabilities. However, modeling feelings has not been pursued vigorously. The modeling of human emotions and emotional states is key to our ability to create next-generation human-like agents. The proposed computer architecture is based on Markov modeling theory. This rich mathematical theory provides us with the ability to model emotional states and how they change. We apply this theory and show how a computational engine can be built to model human emotions. A simple emulation of the proposed architecture is implemented in software and some initial results are presented which show the power of this architecture to develop future generation emotion engines.

1. Introduction

Recently intelligent agents have become very popular in areas of information retrieval, mail management, selection of books, and advertising. These agents capture the essential qualities of human intelligence. To have a believable intelligent agent the agent needs to display emotion. Agents that display emotions will be beneficial in certain contexts. If humans identify with and accept an agent as human they will be able to better communicate with it. Emotionless agents will be viewed as machines. As a result, it is important to develop the emotional component of an intelligent agent. To build emotion engines one needs to first understand how to model emotions.

In previous work of brain modeling there has been significant effort to model the brain's analytical and learning capabilities. Many expert systems have been proposed and developed to model the analytical and intelligent reasoning present in humans. Also, many machine learning techniques have been proposed to model the human learning ability. However, modeling feelings has not been pursued as vigorously. The modeling of human emotions and emotional states is key to our ability to create next-generation human-like agents. This is the primary motivation behind this work.

Human emotions have always been hard to understand. In recent years, in the field of psychology there has been a consensus that emotions are a dynamic process and that to understand emotion a dynamic analysis of state changes must be analyzed [5], [9], [11]. Additionally, in any emotional state there is uncertainty attached with a human behavior response.

Bates [2] suggests that there are three guidelines for proper portrayal of emotions in non-animate agents. These are: 1) Emotional state must be defined and represented, 2) The response (or thought process) reveals the feelings, and 3) Use time wisely to establish emotional response. Margulies [9] suggests that discontinuous rather than continuous state models can be used to model emotions. In discontinuous models, a human jumps only from one discrete emotional state to another and cannot be in an intermediate state. One way to encode emotions is to use case-based reasoning. However, our approach is to provide a more general framework.

We propose a computational engine to model emotions which captures these psychological phenomena. The proposed computer architecture is based on Markov modeling theory. This rich mathematical theory provides us with the ability to model emotional state, dynamic state change, and uncertainty. Markov models provide us with adequate solution techniques and are the underlying mathematical foundation for this work.

We apply this theory and show how a computational engine can be built to model human emotions. We also show how this engine can be expanded and made more complex and can in the future clone human emotions. Section 2 shows the underlying theory behind our architecture. Sections 3 describes our architecture. We emulate this architecture in software and present some initial results in Section 4. Finally, in Section 5 we present our conclusions and also show how we plan to expand our work in this area with the goal of finally building hardware emotion engines.
2. Markovian Emotion Models

Markov models are a type of stochastic models. A stochastic model (or chance model) consists of a family of random variables, which are defined on a given probability space, and are indexed by a parameter. A Markov model is a state-space representation of a stochastic process under certain assumed conditions. The nodes in a Markov model represent certain pre-defined states, and the arcs represent probabilities of movement between states. An important property of Markovian models is the memoryless property whereby the probability of a certain action depends upon a state one is in rather than the history of arriving at a state.

Markov models can be discrete-state or continuous-state. Also, Markov models can be (time)-homogeneous or non-homogeneous. For a homogeneous Markov model the entire past history of the Markov model is summarized in its current state. A discrete-state Markov model can be solved by matrix multiplication or by solving a set of linear equations and one can obtain the transient or steady-state probability of being in each state of a model. For more details see [13].

Discrete-state homogeneous Markov models are very suitable to model human emotions. This is justified by previous work in psychology of modeling emotions as discrete states [2], [9]. Also, the memoryless property of Markov models is important in modeling human emotions as our behavior is highly dependent on our emotional present state rather than how we got there.

An application of Markov models for our purposes is the Markovian Emotion Model. A very basic Markovian emotion model is shown in Figure 1. In this Markovian emotion model the nodes represent human emotional states: sad, joy. The arcs in a Markovian emotion model represent the probability of getting out of a state. The arc values (e.g. Arc1) are set to an initial value (e.g. ω1) which represent an initial state of emotion in a person. These values can get modified by positive stimuli or experiences (β), or negative stimuli or experiences (α). Each experience contributes to the alpha or beta values, and modifies the overall emotional state of a person.

Each new Markov model generated by modification of the arc values can then be solved for obtaining a behavior at a specific emotional state. The behavior can be both a short-term behavior response or a long-term behavior response. The Markov model inherently captures the uncertainty associated with a human response. The instant at which the response is needed is used to pull out the appropriate Markov model or the current Markov model representing the human emotional state diagram. The behavior response (long term or short term) required is used as the solution time into the Markov model.

The Markovian emotion model can be used in two modes of operation. In the first mode---emotion update mode--it gets input from external stimuli and values of α and β are changed. As an example, if the initial value of Arc1 is 0.2 and a person gets a promotion β would increase and α would decrease and the resulting value of Arc1 would increase (Arc1 = 0.25). At the same time the value of Arc2 would decrease. This represents a change in the overall emotional state of a person which will influence future behaviors.

\[
\begin{align*}
\text{Arc1} & : \omega_1 + \beta \omega_1 - \alpha \omega_1 \\
\text{Arc2} & : \omega_2 - \beta \omega_2 + \alpha \omega_2 
\end{align*}
\]

Figure 1 Markovian Emotion Model

In the second mode--behavior query mode--the current Markov emotion model based on a query stimuli and a time value (τ) input is solved from time 0 to time τ. The stimuli provides the initial state for the solution. In this mode, the human emotional state is used to compute a probabilistic short term (τ small) or a long term response (τ large). The state probabilities obtained for each state are used to compute a final response.

As an example, Figure 2 shows how a positive stimuli (like promotion) modifies the emotional state from Figure 2a to Figure 2b. Now if a behavior query is made by providing an query stimuli like a 'word of praise' and a solution time on the second or current emotional state, the Markov model when solved gives the probabilities of being in each state. The query stimuli can also be expressed as a distribution. For this example, if a short term response (τ = 2) is required solving the Markov model (See [13]) would result in:

\[
P = \begin{bmatrix} 0.75 & 0.25 \\ 0.50 & 0.50 \end{bmatrix}
\]

\[
P_{\text{joy}} P_{\text{joy}} = 1/3 + 2/3(1/4)^2 = 3/8 = 0.375
\]

\[
P_{\text{sad}} P_{\text{joy}} = 2/3 - 2/3(1/4)^2 = 5/8 = 0.625
\]
If a long term response is required \( \tau > 10 \). Also, the steady-state response can be found in the limit of \( \tau \) approaching \( \infty \).

To obtain the final behavioral response a discrete integer between 0 and 1 is drawn where 0 represents a sad response and 1 represents a joy response. This random number picking process is biased by the state probabilities of the Markov model (pool of 62.5% 0s and 37.5% 1s).

### Figure 2 Example Emotional States

#### 3. The Emotion Engine

The emotion engine can be implemented in hardware or software. In our estimate the software solution is easier, however, for a real complex emotion engine with many Markovian emotion states the software solution becomes very slow. As a result, we propose a hardware architecture for the design of an emotion engine. The overall emotion engine architecture is shown in Figure 3.

The emotion engine inputs are the input stimuli or the behavior query. The input stimuli are human experiences and are fed into the Stimuli Processor which processes these stimuli and assigns \( \alpha \) and \( \beta \) values in the Alpha and Beta Registers. The specific format of the input stimuli and the implementation of the stimuli processor is project specific. The simplest processor takes a rating of a stimuli (e.g., 0 for good, 100 for bad) and assigns values for each rating (e.g. for 60, \( \alpha = 2 \), \( \beta = 1 \)).

The heart of the emotion engine is the Emotional State Matrix Buffer which stores the transition probability matrix. Depending on the complexity of the Markovian emotion model this buffer is fully or partially loaded. For the simple model shown in Figure 1 only four entries are needed. For complex models representing Markovian emotion states (Sad, Joy, Fear, Anger, etc) \( n \times n \) entries are needed in the Emotional State Matrix Buffer. The maximum allowable states in the Markov model are limited to be 256 (256*256 buffer) in the first generation emotion engine. We feel that this maximum value is more than sufficient to allow modeling complex human emotions.

The Matrix Processor is the computation intensive unit of the emotion engine and takes the Emotional State Matrix Buffer and the Alpha, Beta, and the Solution Time registers and solves the Markov model in hardware. The Solution Time register stores the time used for solving the Markov model. The results of the solution are put in the Emotional State Registers. Each Emotional State Register consists of a state \( S_i \) and its associated probability value \( P_i \). The Matrix processor is shown in more detail in Figure 4.

#### Figure 3 Emotion Engine Architecture

The Response Processor takes the values of the Emotional State Registers and predicts a response which is the output from the emotion engine. For Figure 1, only two registers are needed and the Response Processor picks up the response probabilistically. For more complex implementations of the engine, the Response Processor can do some form of Monte Carlo prediction based on the values of the Emotional State Registers.

The matrix processor consists of a Time Counter which keeps track of the steps of solution. It also has a Partial Results Buffer which stores results for intermediate steps. The Array Processing Unit implements an array style unit [10] which solves a matrix in hardware. The matrix solution for solving a Markov model can be done in two ways. First, one can use matrix multiplication techniques to obtain the solution. Secondly, one can use LU decomposition and solve a triangular linear system [13]. Both of these can be done using hexagonal and linear array cells. Hardware using array cells can be implemented in hardware using VLSI techniques. For more details on
implementing array cell hardware see [10]. Also, other project specific techniques can also be used to solve the Markov model.

4. Initial Results

We emulate the Emotion Engine in software for a variety of Markovian emotion models. The results are shown in Table 1. The simplest model is the two state model shown in Figure 1. Our most complex model is a five state model with the five emotional states being Sad, Joy, Fear, Anger, and Disgust. These five emotional states are consistent with the basic emotions laid out by psychologists [11]. The run time to achieve a query is $\tau = 10$. The run time for the software emulation and the estimated hardware run time are also shown.

It can be clearly seen that the solution time increases exponentially with the state size. Also, for a five state model a response time of 250 seconds seems unreasonable. However, our initial solution is only an non optimized emulation. It is estimated that a hardware implementation would take about 250 $\mu$ seconds. This is the reason we recommend a hardware approach in building complex emotion engines.

<table>
<thead>
<tr>
<th>Model</th>
<th>States</th>
<th>Emulation Time</th>
<th>Hardware Time*</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 State</td>
<td>(sad, joy)</td>
<td>2 sec</td>
<td>2 $\mu$ sec</td>
</tr>
<tr>
<td>3 State</td>
<td>(sad, joy, fear)</td>
<td>12 sec</td>
<td>12 $\mu$ sec</td>
</tr>
<tr>
<td>4 State</td>
<td>(sad, joy, fear, anger)</td>
<td>64 sec</td>
<td>64 $\mu$ sec</td>
</tr>
<tr>
<td>5 State</td>
<td>(sad, joy, fear, anger, disgust)</td>
<td>250 sec</td>
<td>250 $\mu$ sec</td>
</tr>
</tbody>
</table>

*Estimate Only

Figure 4 Matrix Processor Architecture

5. Conclusions

In this paper, a computational architecture to model human emotions is described and developed. The proposed computer architecture is based on the Markov modeling theory. This mathematical theory allows us to model human emotions by using Markovian emotion models. We also developed an emotion engine architecture which can be implemented in hardware.

A simple emulation of the proposed architecture is implemented and initial results are presented. Concluding, in this work we show how to model emotions and feelings. The modeling of human emotions is key to our ability to create next-generation human-like agents. In the future, we plan to do some more software emulation and refine the theoretical aspects of our modeling methodology. Following that, we will refine the hardware architecture and run detailed simulations. A long term goal is to build a prototype emotion engine in hardware.

References