Multimodal Adaptive Interfaces

Deb Roy and Alex Pentland
Perceptual Computing Group
MIT Media Lab
20 Ames Street, Rm. E15-388
Cambridge, MA 01239
dkroy@media.mit.edu
(617) 253-0378, (617) 253-8874 (fax)

October 16, 1997

1 Introduction

Our group is interested in creating human machine interfaces which use natural modalities such as vision and speech to sense and interpret a user's actions [7]. In this paper we describe recent work on multimodal adaptive interfaces which combine automatic speech recognition, computer vision for gesture tracking, and machine learning techniques.

Speech is the primary mode of communication between people and should also be used in computer human communication. Gesture usually accompanies speech and provides information which is at times complementary and at times redundant to the information in the speech stream. Depending on the task at hand and the user's preferences, she will use a combination of speech and gesture in different ways to communicate her intent.

In this paper we present preliminary results of an interface which lets the user communicate using a combination of speech and diectic (pointing) gestures. Although other efforts have been made to build multimodal interfaces, we present a system which centers around on-line learning to actively acquire communicative primitives from interactions with the user.

In this paper we begin in Section 2 by examining some of the problems of designing interfaces which use natural modalities and motivate our approach which centers around enabling the interface to learn from the user. In Section 3 we give an overview of our approach to addressing the issues raised in Section 2. Section 4 describes the multimodal sensory environment we have built for developing our interfaces. This environment includes a vision based hand tracking system and a phonetic speech recognizer. In Section 5 we introduce Toco the

Toucan, an animated synthetic character which provides embodiment for the interface. Section 6 details the learning algorithm which allows Toco to learn the acoustic models and meanings of words as the user points to virtual objects and talk about them. In Section 7 we summarize our work this far and finally in Section 8 we provide a road map for where this work is going.

2 Problem

2.1 Reference Resolution

The problem of reference resolution is the process of inferring the user's intent based on observing her actions. For example if the user says "make the box blue" in a graphical design application there are several references which must be resolved before the system can act. The referent of the words "the box" must be resolved. This might be done based on prior context of the interaction (for example, the user has just created a box) or perhaps by looking to where the user is pointing. Similarly "blue" might map to different parts of the color spectrum depending on the user's sense of color. Finally the verb "make" might be interpreted in more than one way. A line drawing of a box might be made blue by changing the color of the bounding lines to blue, or by filling the box with blue.

There are two intertwined subproblems which must be solved to address the reference problem. The first is to recognize the surface form actions which a person decides to make to communicate an intent. For example in speech, the exact acoustic word she chooses to refer to some color. This acoustic word depends on several factors including:

- The user's choice of words. One user might say "make" while another prefers to say "paint".
- The language the user speaks. For example one user might speak in English while another speaks in Hindi.
- Individual differences in pronunciation including regional accents and speech impairments.

A second subproblem is interpret a person's words correctly once the words have been recognized.

Perhaps the simplest solution to reference resolution is to prescribe an allowable set of actions for the user (i.e. words they can say, gestures they can make) and assign fixed meanings to each action. The problem with this method is that there is no set of actions and associated meanings which will match the expectations of all potential users of the system. For example Furnas et. al. [2] did a series of experiments on object naming and found that:

> People use a suprisingly great variety of words to refer to the same thing. In fact, the data show that no single access word, however well chosen, can be expected to cover more than a small proportion of users' attempts...The idea of an "obvious", "self-evident" or "natural" term is a myth! ... Any keyword system capable of providing a high hit rate for unfamiliar users must let them use words of their own choice for objects.

In some of their experiments, two users would choose the same word to describe a different action leading Furnas to the further conclusion:

> Because of the inherent ambiguity of users' own words, the system can never be sure it has correctly inferred the user's referent.

Our conclusion is that to make effective interfaces there need to be adaptive mechanisms which can learn what words and gestures an individual will wish to use, and how they expect those words and gestures to be interpreted. Thus, for example, a speech recognizer should be able to learn the vocabulary which a person wants to use rather than be preprogrammed with a static vocabulary, and furthermore the system should learn how to interpret the acquired words.

2.2 Learning to Communicate Requires Multiple Modalities

We have argued that the best solution for choosing which user actions the interface will recognize and respond to should be learned by the interface. The type of learning we would require in an adaptive interface needs some type of training information to guide the learning process. For example if we want the interface to learn the meaning of a certain hand gesture, the system must have some information from a different input channel (i.e. other than the gesture channel) which gives the machine some indication that the gesture should be learned and how to act upon it. In other words, to learn user actions in some modality, we must introduce a second modality which provides enough additional information for the system to learn.

2.3 Training Protocols for an Adaptive Interface Must be Natural

A learning interface forces the user into the role of teacher. For the interface to be usable it is crucial that the protocol for teaching the machine be easy to use and not significantly burden the user. Reeves and Nass [9] have performed numerous studies of social interactions between humans and machines and have found that best way to design an interface is to ensure the the interface follows social conventions which people already know:

When media conform to social and natural rules...no instruction is necessary. People will automatically become experts in how computers, television, interfaces, and new media work. (p. 8)

This suggests that an effective interface should assume the role of a social actor which ellicits certain social behaviors from the user and is able to interact appropriately.

3 Overview of Our Approach

Our approach to dealing with the problem of reference resolution is to enable the interface to learn from interactions with the user. Rather than force the user to learn to use a predetermined interface, we put adaptive capabilities into the interface so that it can learn from the user. As we note above, learning to communicate in one modality requires training information from a second modality. We have developed a multimodal environment which uses speech and gesture input, and displays both graphical and auditory information. To address the issue of social interactions and natural protocols, we have embodied the interface as an animated synthetic character called Toco the Toucan. In the rest of this paper we will use the name Toco to refer to various aspects of the interface (Section 5).

An issue related to reference resolution is how to ground the semantics of words and gestures. Our approach is to represent all possible referents of the user's speech as graphical objects which are visible on a color display. Toco is situated in a virtual world populated with simple objects of various colors and shapes. The color and shape attributes of each object and their location are available internally to Toco. The user is expected to only refer to objects which are in view, similar to interactions with small children where adults typically avoid reference to items which are not present in space and time. Attributes of the virtual objects such as their color, shape and location, are encoded internally and are accessible to the system.

When the system is first started, Toco has only one acoustic word model hardwired, his name. To get Toco's attention, the user must call his name. When Toco recognizes his name, he will look up at the use and wait for one of two things:

- If the user points to an object and speaks a word, Toco will try to associate the word which some attribute of the object.
- If the user says a word without pointing, Toco will respond to the speech by looking at an object which he associates with the word.

The goal of this work is to demonstrate generalized learning of word meanings using natural diectic gestures to enable reference resolution. Although the user might point to the same object and say both "red" and "ball", over time Toco must learn that the word "red" refers to the color attribute of objects and not shape. Thus there is a logical inference problem which must be solved. In addition, Toco is dealing with acoustic input which tends be be noisy, so the logical inference must be solved in a statistical framework which can account for acoustic variation. A statistical framework is also useful since it can better cope with errors which occur during training due to the teacher.

We now describe the various components of our system which lead to the interaction described above.

4 A Multimodal Sensory Environment for Adaptive Interfaces

We have created an environment to facilitate development of multimodal adaptive interfaces based on the smart desk environment [7]. In its current configuration, the user sits at a desk facing a 70" color projection screen. The screen is used to display a graphical embodiment of the interface and virtual objects (described in Section 5). The user's hand gestures and speech are sensed and analyzed in real time using methods described in this section.

4.1 Diectic Gesture Tracking using Real-time Vision

The vision system uses two color video cameras to sense the person's hand gestures. One camera is mounted directly overhead and the second provides an orthogonal view from the right side.

To locate the user's hand, we use a mixture of Gaussians to model the color of human skin under normal room lighting. The mixture model produces a probability of belonging to skin for each pixel from the camera's image. These probabilities are biased by the location of the pixel; pixels closer to the screen (where user's are expected to point) are weighted higher than pixels further from the screen. In practice we have found this location based bias to effectively locate the tip of a persons arm when then point to the screen (i.e. during diectic gestures).

The location biased mixture model probabilities are thresholded to locate skin in the camera's view. A connected regions algorithm is then run to locate the largest skin region in view, and the mean and covariance of the hand shape are estimated from this region. Currently we are only using the mean of the region. The hand coordinates of each camera are fused via a geometric mapping which the user specifies off line using anchor points. These are used to calibrate the 2D data into screen coordinates (effectively recovering a 3D estimate of hand position).

4.2 Speech Analysis using Real-time Phoneme Recognition

4.2.1 Front-end Analysis

Audio is sensed using a head mounted noise cancelling microphone. The audio signal from the microphone is sampled using a 16-bit 16 kHz analog to digital converter and them processed using the Relative Spectral (RASTA) [4] algorithm. RASTA provides a spectral representation of the audio signal which is relatively invariant to background noise[1].

The RASTA coefficients are computed on 20ms frames of audio (recomputed every 10ms) and fed into a recurrent neural network (RNN) similar to the system described in [10] to produce phoneme probability estimates at a rate of 100Hz. The RNN has been trained using back propagation in time [13] on the TIMIT database [12] which is a database of phonetically labeled speech recorded from 630 adult male and female native English speakers from all the major dialect regions of the United States. The RNN produces a 40-dimensional phoneme probability vector every 10ms.

Thirty nine of the outputs estimate the probability of the 39 phonemes which constitute spoken English. We use the 39 phoneme class grouping described in [5].

The final RNN output is the probability estimate for silence. We use this output to detect "speech events" (i.e. when the user says something) as described in the next section.

4.2.2 Speech Event Detection

The phoneme recognizer runs continuously (there is no push-to-talk button). We drive the finite state machine (FSM) shown in Figure 1 to detect sustained speech activity in the input stream. Normally the FSM will be in **state 1**. When the RNN estimates that an input frame is non-silence[2], the FSM transitions to **state 2** and clears *count1* . If a silence frame is detected before *threshold1* speech frames, the FSM returns to **state 1** and the speech frames are discarded. If *count1* exceeds *threshold1* the FSM enters **state 3**. Similarly **state 4**is entered whenever a silence frame is detected. The FSM transitions to **state 5** when more than *threshold2* contiguous frames of silence have been observed. At that point the speech frames which were observed in the second and third states are concatenated and form a speech event which are processed further. To clarify, if a person speaks a one second utterance, the resulting speech event will be a 40 by 100 matrix of probability estimates (40 phonemes by 100 frames).

[1]Background noise is not a major issue in our current configuration which uses a noise canceling close talking microphone, but in the near future we plan to use a desk mounted microphone array which will be prone to greater amounts of background noise.

[2]i.e. the probability estimate of silence from the RNN is lower than the probability estimate of one of the other 39 RNN outputs

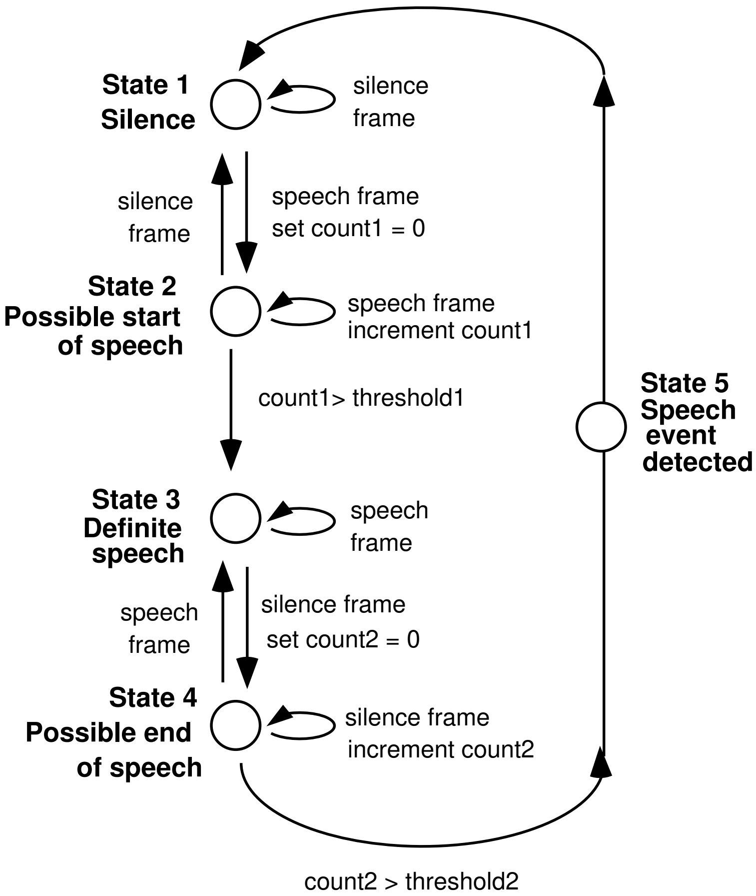

Figure 1: Finite state machine for detecting speech events

4.2.3 Converting Speech Events to Phoneme Strings

We use the RNN outputs as emission probabilities within an Hidden Markov Model (HMM) framework [6]. When a speech event is detected, a Viterbi search is performed on the output from the neural network representing the speech event to find the most likely phoneme sequence for the event. The Viterbi search allows any sequence of the 39 phonemes or silence. We refer to this as a "phoneme loop" model. The Viterbi search incorporates duration models and bigram phoneme transition probabilities to improve phoneme recognition accuracy. The duration and bigram transition statistics were estimated from the TIMIT training data set.

Using the Viterbi search, our system recognizes phonemes with 68% accuracy on the standard speaker independent TIMIT recognition task. This performance is close to current state of the art recognition for context independent phoneme recognition [10].

4.2.4 A Metric for Comparing Speech Events to Phoneme Strings

In this section we define a distance metric for comparing a speech events to a reference phoneme string. Typically the phoneme string is generated by the Viterbi search although it may also be generated manually in some cases. The distance metric is used for both spotting keywords and for clustering words which are being learned (see Section 6 for details).

The reference phoneme string may be thought of as a hidden Markov model (HMM) which can generate arbitrary speech events. We can then compute a confidence measure that an event was generated by an HMM using standard Viterbi search. Following methods developed for keyword spotting confidence measures [11] we normalize this measure across utterance as follows.

First we compute the log probability of an event e using a forced Viterbi alignment with phoneme transitions determined by the reference phoneme string. We denote this as $log(p(reference \mid e))$.

Next we compute the log probability of the event e using a Viterbi search with a phoneme loop model with phoneme bigram transition probabilities estimated from the original TIMIT data. We denote this as $log(p(phonemeloop \mid e))$.

Finally, we can define the distance between the event and the reference string to be:

$$d(ref, e) = log(p(ref \mid e)) - log(p(phonemeloop \mid e)) \tag{1}$$

The details for computing the probability of an observation sequence through a HMM state space may be found in [8]. We have found that this normalized measure works quite well for both keyword spotting (which is done by having a pre-specified set of phoneme strings representing the keyword), and for clustering acoustic data (see Section 6.3).

5 Toco the Toucan: Embodiment of the Interface to Support Natural Social Interactions

Toco is rendered in real time using 3-D graphics and is displayed on a 70 inch display. He can move his head to look at any location in 3-D space. His eyes can blink and squint, his beak can open, close and smile, and his eyebrows can be raised, lowered, and tilted.

Toco's face and body language may be used as an output display to convey the system's internal state. For example his direction of gaze gives the person immediate feedback of where Toco thinks the user is pointing. Subtle facial cues such as widening of eyes and raised eyebrows are used to signal when Toco is alert and attending to the user. Confidence levels for speech or gesture recognition can also be displayed by showing confusion in Toco's face, or a knowing nod for a clearly understood action.

For output speech generation, we are using Entropic's TrueTalk concatenative phonetic speech synthesizer. Toco can output acquired words by sending phoneme sequences (generated by the RNN-Viterbi recognizer) to the synthesizer. Currently Toco repeats words which he hears giving the user immediate feedback of how well his phoneme recognition is working and which words Toco can consistently recognize.

We believe that by embodying our system in an interactive character, we can evoke a set of assumptions from the user about what to expect from the interface. If the character can meet these expectations, then the interface may be understood and used with minimal instructions [9].

6 Learning in the Interface

The most natural way to deal with the problem of reference resolution is to learn the set of words and gestures which a user prefers to use, and also learn their semantics within the context of some application. In our current work we have focused on learning spoken words and their meanings but we note that similar lines of research can also be pursued to learn gestures and their meanings. We now describe several aspects of Toco which lead to word learning.

6.1 Behavior State Model

Toco's current interaction model is driven by the finite state machine shown in Figure 3. In its default state, the FSM stays in the **inattentive** state. If Toco hears his name (using an "innate" set of reference phoneme strings for his name, and thresholding the distance metric described in Section 4.2.4), the FSM moves to the **attentive** state. To display that Toco is paying attention, his eyes and mouth open wide and he looks up at the user.

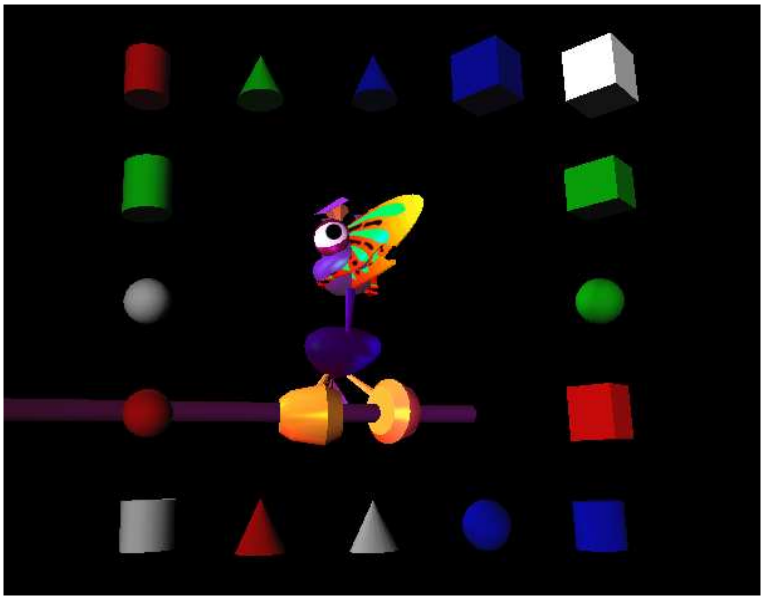

Figure 2: Toco the Toucan, the interface's graphical embodiment. Shown looking at a white cube in the virtual environment

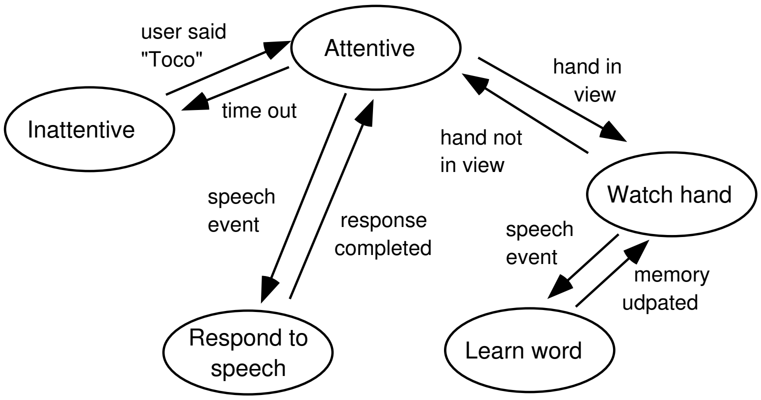

Figure 3: Toco's behavioral finite state machine

If the user does not say anything within about 5 seconds, the FSM returns to the **inattentive** state and his face returns to an inattentive expression.

While in the **attentive** state, if the vision system detects the user's hand, Toco enters the **watch hand** state. In this mode, Toco will look at the closest object in the virtual world (see Section 6.2 for details) and actively move his gaze as the user points to different objects. If the user removes her hand from view, the FSM returns to the **attentive** state. If the user utters speech while Toco is watching an object, Toco learns the word and its associations with the object (see Section 6.3).

Finally, if Toco is in the **attentive** state and the user utters some speech without her hand in view, Toco responds to the speech by looking at an object which is most likely the referent of the speech (see Section 6.5).

Toco also generates speech in response to the user's speech when Toco is in either the **respond to speech** or **learn word**states. The speech synthesis output gives the user immediate feedback of how well Toco's phonetic recognizer transcribed the user's speech. This feedback may lead the user to adapt their speaking style to better match Toco's capabilities.

6.2 Grounding Semantics: Objects in Toco's Virtual Environment

Objects in Toco's graphical environment are internally represented by a set of *attribute vectors* which encode characteristics of the object. For example the white cube in the top right corner of Figure 2 is represented as:

```
Object {
   r,g,b = 1.0 , 1.0 , 1.0
   shape = 0, 0, 1, 0
}
```

We use the following notation to represent the an element of attribute vector set:

$$a^i_j(k), \quad i = 1, 2 \ldots n, \;\; j = 1, 2 \ldots m_i \tag{2}$$

where a^i_j is the j^{th} element of the i^{th} attribute vector representing the k^{th} object, there are n vectors in an attribute set, and the i^{th} vector has m_i elements. Thus for example the attribute vector set for the white cube described above has $n = 2$, $m_1 = 3$, $m_2 = 4$ and $a^2_3 = 1$.

The shape vector encodes shapes discretely as a binary four dimensional vector encoded as (cone, sphere, cube, cylinder). Thus the cube is represented by setting the third bit and clearing the remaining bits.

The goal of representing objects as attribute vectors is to facilitate learning of word meanings in terms of these attribute primitives. For example if the user points to the object specified above and says the words "white" and "cube" several times, the goal would be to learn high associations between "white" and the color vector (1.0, 1.0, 1.0), and similarly to learn high associations between "ball" and the shape vector (0,1,0,0). Multiple training examples are necessary to learn not to associate "ball" with any particular color vector.

We hope that by encoding attributes of the objects which are perceptually salient to the user, Toco will be able to discover the meaning of words grounded in these perceptual primitives. Of course as the virtual world becomes more complex and objects take on functional meaning, this representation scheme will become a complex issue.

6.3 Learning Words

In this section we describe how Toco learns words. Words must be learned at two levels: their acoustic models, and their association with object attribute vectors.

At the highest level, Toco's memory consists of a set of *word clusters*. Each cluster is comprised of a set of one or more phoneme strings, and an *association weight vector*:

$$w_j^i(l), \quad i = 1, 2 \ldots n, \;\; j = 1, 2 \ldots m_i \tag{3}$$

where $w_j^i(l)$ is the j^{th} element of the i^{th} weight vector of the l^{th} word cluster, and n and m_i are as defined in Equation 2.

Similar to [3] we set the weight vectors of cluster l to the mutual information [1] between the observation of word cluster l and each attribute vector:

$$w_j^i(l) = log \left\{ \frac{p(a_j^i \mid V_l)}{p(a_j^i)} \right\} \tag{4}$$

where V_l signifies the presence of an spoken word belonging to word class l. Since the elements of the attributes are binary variables, we are able to use simple smoothed relative frequencies to estimate the probabilities in Equation 4 [3].

Equation 4 has some intuitively satisfying properties. If the probability of an attribute element conditioned on the presence of a word from class l is equal to the unconditioned probability of that element, the weight will be set to zero and indicates that the word class is not useful for predicting the value of the attribute vector element. If the conditioned probability is higher than the unconditioned probability, the weight is assigned a positive value indicating the positive correlation between the word class and the attribute element. And finally if the conditioned probability is lower than the unconditioned probability, the word class will be assigned a negative weight.

When Toco first starts running, he has no word clusters in memory. When the user points to an object and utters a word, Toco will create a word cluster and initialize it with the phoneme string extracted from the user's speech. The association weight vector for this cluster is then set using Equation 4. Subsequent training examples (i.e. where the user is pointing to an object as she says a word) are incorporated into Toco's associative memory by the following steps.

First we need to define the distance from an event to a word cluster. We denote the g^{th} phoneme string of cluster V_l as s^l_g. The distance from word cluster V_l to event e is defined as the distance between the event and the closest phoneme string within the word cluster:

$$d_{cluster}(V_l, e) = \min_g \; d(s^l_g, e) \tag{5}$$

where $d()$ is defined in Equation 1.

Using Equation 5 the index of the word cluster closest to the speech event is found:

$$l_{best} = arg \min_l \; d(V_l, e) \tag{6}$$

and the distance from the event e to the closest cluster is simply:

$$dist = d_{cluster}(V_{l_{best}}, e) \tag{7}$$

At this point the algorithm compares $dist$ to a predefined split/merge threshold. If $dist$ is greater than the threshold, a new cluster is formed and initialized with the phoneme string extracted from the event e. If $dist$ is less than the threshold, the phoneme string extracted from the event e is added to the cluster $V_{l_{best}}$, effectively merging the acoustic model of the event with the existing models of the cluster. After the clusters are updated (by either a merge or split), the weights of the word clusters are updated according to Equation 4.

6.4 An Example of Word Clusters Formed from Interactions with Toco

To clarify the learning scheme presented in the previous section, this section presents two examples of clusters which were formed during an interaction with Toco and the virtual objects shown in Figure 2.

Below we have listed the connection weights of two word clusters, labeled Cluster 1 and 2. Cluster 1 was automatically formed from seven repetitions of the word "ball" by a single person. Five of the seven repetitions were transcribed consistently as "b aa l" by the recognizer, and the remaining two were transcribed as minor variants. The first three weights show mutual information with each of the three color dimensions in the RGB space. The relatively high correlation with the third dimension means the user probably pointed to more green balls while teaching this word than any other color. The second set of weights encodes correlations to the shape attribute vector. This cluster has a strong association with the sphere element. Thus this cluster encodes that the word "ball" refers to spherical objects.

A similar analysis of Cluster 2 shows that the cluster is associated most strongly with the red dimension of the RGB space (and this association is stronger than the association to any particular shape), and the cluster was formed from three repetitions of the word "red".

```
Cluster 1
  count = 7
  Phoneme strings: b aa l (5), m aa l (1), b aa ow l (1)
  weights: -1.20 -0.19 0.25 | -1.20 0.71 -1.20 -1.20
  condSum: 0 3 4 | 0 7 0 0 |
  condProb: 0.09 0.47 0.52 | 0.09 0.82 0.04 0.05

Cluster 2
  count = 3
  Phoneme strings:  r ae n (1),  r ae (2)
  weights: 0.41 -0.36 -0.36 | 0.33 -0.36 0.21 -0.36
  condSum: 3 0 0 | 2 0 1 0
  condProb: 0.56 0.38 0.29 | 0.40 0.26 0.23 0.10
```

6.5 Responding to Words

Toco can respond to words based on the word clusters which have been formed from previous interactions with the user. When the user says a word without pointing to the object, Toco finds the closest word cluster to the speech event using Equation 6, and the distance to the cluster using Equation 7. The distance is compared to a response threshold which determines whether Toco will respond to the event. If the distance is greater than the threshold, Toco treats the event as unknown and takes no action.

If the distance is less than the threshold, Toco finds the object in view which has highest association with the cluster (see below) and responds by looking towards the object and vocalizing (by sending the phoneme string extracted from the new event to the speech synthesizer).

The object is selected by computing the association strength between each object and the word cluster and selecting the object with the highest association. The association between an object k and word cluster V_l is computed as:

$$y_k^l = \begin{matrix} max \\ i \end{matrix} \; \frac{\sum_{j=1}^{m_i} a_j^i(k) w_j^i(l)}{m_i}, \quad i = 1 \ldots n \tag{8}$$

The max operator in Equation 8 implements a competition among attribute vectors within the vector set. The dot product of association weights for the cluster and the associated attribute vector is computed for each vector in the set and normalized by m_i, the dimensionality of the i^{th} vector. The association of the entire word cluster to the object is set to its highest normalized vector association. It is important to note that this equation implies that a word will never have a meaning which combines information across multiple attribute vectors (due to the max operator). Thus, for example, this learning scheme will be unable to reliably learn a word which means both red in color, and spherical in shape[3].

7 Summary

We have provided a snap shot of our work in progress on adaptive multimodal interfaces. Toco the Toucan demonstrates an interface which an learn words and their context-limited semantics by natural multimodal interactions with people.

Toco can learn acoustic words and their meanings by continuously updating association weight vectors which estimate the mutual information between acoustic words and attribute vectors which represent virtual objects in Toco's world.

We ground the reference of all speech in perceptual attributes of objects in Toco's world. The interface is embodied as an animated character which is situated in a graphical world. We are able to use Toco's facial expressions and body language as a natural output display for the system's internal state including focus and level of attention, and internal confidence levels. Toco is able to generate speech by sending phoneme strings to a speech synthesizer.

8 Future Work

Some areas of future work on our interface includes:

- Attribute vector sets in our current implementation take on binary values. We plan to begin encoding continuous valued attributes which will require modification of Equation 4 since smoothed relative frequencies will no longer be sufficient to estimate the conditioned and unconditioned probabilities.
- Provide an interface (for example buttons) to give Toco explicit positive and negative feedback depending on his actions. This will enable us to incorporate reinforcement learning techniques into the learning algorithm and accelerate learning rates.
- Build a garbage collection mechanism for pruning word clusters which have insufficient activity over time.
- Modify the word cluster growing algorithm to combine acoustic distance metrics with a semantic distance metric which measures distances between attribute vectors.

[3] Thus Toco cannot learn that "apple" is object which is red and spherical. We plan to address this issue in the future

- Although Toco must learn words spoken in isolation, we plan to enable Toco to recognize strings of connected speech. An exciting extension of this is to then learn word ordering over time and effectively acquire simple grammars which can be fed back into the recognition process to improve accuracy. The acquired grammars can also be used during speech generation to select word order.
- Introduce a limited set of actions which Toco can perform on objects. Toco can then learn verb words which are associated with these actions.
- We plan to use active learning so that Toco can ask questions about objects to actively illicit training data from the user. For example if Toco is not sure whether "blue" means (0,0,1) in RGB color space or whether it means spherical, Toco could look towards a non-spherical blue object and say "blue?" (using an acquired phoneme string and by putting an upwards intonation curve in the speech synthesizer) to get new data to resolve the reference ambiguity.

9 Acknowledgments

Tony Jebara designed and implemented the hand tracking system. Mike Hlavac created the graphical models for Toco. Brian Clarkson wrote the Viterbi search software. Thanks also to Allen Gorin, Bruce Blumberg and Alex Pentland for useful discussions and suggestions.

References

[1] T.M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[2] G.W. Furnas, T.K. Landauer, L.M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communications. *Communications of the Association for Compuring Machinery*, 30:964–972, 1987.

[3] A.L. Gorin, S.E. Levinson, and A. Sankar. An experiment in spoken language acquisition. *IEEE Transactions on Speech and Audio Processing*, 2(1):224–240, january 1994.

[4] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing*, October 1994.

[5] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, November 1989.

[6] Nelson Morgan and Herve Bourlard. An introduction to hybrid hmm/connectionist continuous speech recognition. *Signal Processing Magazine*, pages 25–43, May 1995.

[7] Alex Pentland. Smart desks, desks, and clothes. In *Proceedings of ICASSP*, pages 171–174, Munich, Germany, April 1997. IEEE Computer Society Press.

[8] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[9] B. Reeves and C. Nass. *The Media Equation*. Cambridge Univeristy Press, 1996.

[10] Tony Robinson. An application of recurrent nets to phone probability estimation. *IEEE Trans. Neural Networks*, 5(3), 1994.

[11] Richard Rose. *Word Spotting from Continuous Speech Utterances*, chapter 13, pages 303–329. Kluwer Academic, 1996.

[12] Stephanie Seneff and Victor Zue. Transcription and alignment of the timit database. In *Proceedings of the Second Symposium on Advanced Man-Machine Interface through Spoken Language*, Oahu, Hawaii, November 1988.

[13] Paul Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78:1150–1160, 1990.