Speech Recognition:
How it Works and How it's Used

Avi Drissman
Dr. Sethi
CSC 496
March 26, 1997

Speech recognition has always been an accepted and expected feature of futuristic computer systems. From HAL in *2001: A Space Odyssey* ("Open the pod bay doors, HAL." "I'm sorry Dave, but I'm afraid I can't do that.") to Lieutenant Commander Data in *Star Trek: The Next Generation*, authors who write of the future anticipate the day when computers can understand, interpret and follow the spoken words of humans. Even today, the promises of speech recognition technologies are alluring, allowing possibilities of alternative input for devices unable to use conventional means, and a more natural human-computer interaction scenario than is possible today.

The art of getting the computer to understand a command and interpret it correctly falls into the field of natural language processing, and it surely is an interesting field. However, what concerns us in this paper is the translation from speech (or more precisely, a digitized version of speech) to some encoded representation of the words that were spoken.

Recognition, in any form, involves the classification of input data into categories. For speech recognition, there are several choices for the categories that can be used [16]. For a simple "yes"/"no" recognizer, for example, all that is needed are two models—one for each word[1]. In a more ambitious recognizer, with perhaps thousands of words, since it would be impractical to create a model for each word, smaller units must be used. These units can be anything from syllables to phones (individual sounds in words—see [9] for more information), although a dictionary is then needed to interpret the sequence and turn it into a set of words.

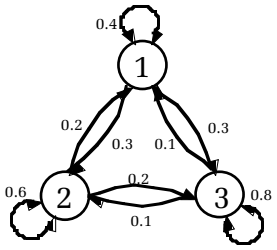
The most popular method of modeling the chosen units of speech is the hidden Markov model (HMM). A HMM is a group of states with a given set of transition probabilities between them, and at each state a set of probabilities for production of one signal from a set. Each transition probability is assigned a spot $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$ in our transition matrix A. Or, rephrased slightly, a_{ij} is the probability that, given that the current state is state *i*, the next state will be state *j*. For example, we can take as a set of states weather conditions for an area [14].

[1] Ignoring, of course, the common problem of people not following directions and giving input such as "yup," "uh-uh," "uh-huh," and the like. If we need to deal with it, we can reject input we cannot classify with a particular degree of confidence and reprompt the user with more explicit directions.

If we assume the weather to be one of three states: state 1 being rainy, state 2 being cloudy and state 3 being sunny, we can have a transition matrix

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}.$$

So, for example, if one day is cloudy, the probability of the next day being rainy is a_{21}, which is 0.2 (20%). Another way to picture it is using a state diagram, with circles representing states:



This is an example of a first-order observable Markov model. It is first-order because the kind of weather each day in this model only depends on the weather of the previous day. It is observable—not hidden—because, given the sequence of states, we know all there is to know.

In a *hidden* Markov model, we don't know the sequence of states. During each state, one of a set of output signals will be chosen. So, to continue our example, let's assume that the temperature during a day (cold or warm) depends on the weather. Let's say $b_j(k)$ is the probability that we will have an observable signal k at state j. Let's take two states, cold and warm, and assign probabilities depending on the weather:

$$B = \{b_j(k)\} = \begin{bmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \\ 0.1 & 0.9 \end{bmatrix}.$$

And finally, we have a sequence of states, but we don't know at which state to start.

So we take a set of probabilities $\pi=\{\pi_i\}$ where π_i is the probability we start at state *i*. For our example, take

$$\pi = \{\pi_i\} = \begin{bmatrix} 0.2 \\ 0.3 \\ 0.5 \end{bmatrix}.$$

So we have an observation sequence $O=\{O_1O_2O_3...O_T\}$; in the example, we could have O={cold, cold, warm, warm}. We have the parameters of the HMM, which we abbreviate $\lambda=(A, B, \pi)$. And we have three questions [14], [4], [5]:

1. If we have an observation sequence O, and a model λ, how do we calculate the probability of the observation O?

2. How do we choose the sequence of states which would best explain the observation O?

3. If we have a set of observations which we feel reflects reality better than our model, how can we tweak the parameters of the model to better match the observations?

The first question is a matching question—does our observation fit the model λ? If we had more than one HMM to compare the data to, we could pick the one with the highest probability, the best match. We evaluate state by state the probability of being at that state with all previous observations in the sequence matching—$\alpha_t(i)$—and multiply it by the probability of the current observance. More precisely put [14]:

Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1),\ 1 \le i \le N.$$

Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}),\ 1 \le t \le T-1,\ 1 \le j \le N$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

This algorithm is called the forward method. So, for our weather example, we have:

t:	**1**	**2**	**3**	**4**
1:	$.2 \times .8 = .16$	$.099 \times .8 = .0792$	$.04716 \times .2 = .009432$	$.0148313 \times .2 = .00296636$
2:	$.3 \times .5 = .15$	$.143 \times .5 = .0715$	$.06784 \times .5 = .03392$	$.0274566 \times .5 = .0137283$
3:	$.5 \times .1 = .05$	$.118 \times .1 = .0118$	$.0475 \times .9 = .04275$	$.0438136 \times .9 = .03943224$
				Total: .0561269

This says that given our model, an occurrence O={cold, cold, warm, warm} would only happen 5.6% of the time.

The second question of most probable path comes up because a HMM has a hidden layer. Because there is no way to know which state the model is in, when it is desired to know the sequence of states a series of observations would use, the optimal path must be calculated.

Viterbi's method is similar to the forward method described above, but instead of summing all possible states and then multiplying by the probability, we must find the most likely transition and multiply it by the probability. In equations, we have [14]:

Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \ 1 \le i \le N.$$

$$\psi_1(i) = 0.$$

Induction:

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}]b_j(O_t), \ 2 \le t \le T, \ 1 \le j \le N$$

$$\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N}[\delta_{t-1}(i)a_{ij}], \ 2 \le t \le T, \ 1 \le j \le N$$

Termination:

$$P^* = \max_{1 \le i \le N}[\delta_T(i)]$$

$$q_T^* = \operatorname*{argmax}_{1 \le i \le N}[\delta_T(i)]$$

Path backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \ t = T-1, T-2, \ldots, 1$$

So, in our example, Viterbi's algorithm gives:

t:	**1**	**2**	**3**	**4**
1:	.2 × .8 = .16	① .064 × .8 = .0512	① .02048 × .2 = .004096	② .0027 × .2 = .00054
2:	.3 × .5 = .15	② .09 × .5 = .045	② .027 × .5 = .0135	② .0081 × .5 = .00405
3:	.5 × .1 = .05	① .048 × .1 = .0048	① .01536 × .9 = .013824	③ .0110592 × .9 = .00995328

The circled number to the left of each cell indicates the state which was the most probable predecessor state to that particular state. At the end of the calculation, look at the final state with the highest probability—3, sunny. From our record keeping (the numbers in the circles) the sunny state most probably came from the previous sunny at time=3. When it's all traced back, we can see that the most probable set of states to produce {cold, cold, warm, warm} is {rainy, rainy, sunny, sunny}.

The third problem of adjusting the parameters λ to fit the data better is a more difficult problem than the previous two, and will not be discussed here. The method is called Baum-Welch re-estimation, and details can be found in [14]. Briefly, from the current λ and the observation set, new frequencies are calculated. The new π is assigned the expected beginning frequency of the states, the new A is assigned a percentage of expected transitions from state to state, and the new B is assigned the expected emission of signals.

How HMMs are used

Most modern speech recognition systems use a HMM of each basic unit of speech, whether a phone or an entire word [9]. In general, the speech recognizer takes the digitized samples of the word, and runs them through a preprocessing stage [5]. In preprocessing, a window of 20 to 30 msec of audio is processed, and auditory characteristics are extracted. Since a whole vector of data is difficult to handle, a codebook [14] can be formulated so that the audio data can be dealt with as one number. Then, to process a word, the data the word comprises is compared with the HMMs of the words in the dictionary, and HMM which the audio data matches the best is picked as the word that the user intended to say. And finally, some post-processing may be needed to deal with similar-sounding words such as "to", "too" and

"two". Methods of post-processing vary from application to application, and may not be needed in recognition systems with a very small, strict vocabulary.

The solutions to the three problems find uses in a speech recognition system. For a typical system with *W* words [14],

> for each vocabulary word, we have a training sequence consisting of a number of repetitions of sequences of codebook indices of the word (by one or more talkers). The first task is to build individual word models. This task is done by using the solution to Problem 3 to optimally estimate model parameters for each word model. To develop an understanding of the physical meaning of the model states, we use the solution to Problem 2 to segment each of the word training sequences into states, and then study the properties of the spectral vectors that lead to the observations occurring in each state.... Finally, once the set of *W* HMMs has been designed and optimized and thoroughly studied, recognition of an unknown word is performed using the solution to Problem 1 to score each word model based upon the given test observation sequence, and select the word whose model score is highest.

Neural network recognizers

The natural alternative to HMMs in speech recognition is a neural network. Neural networks are well known for their strengths in classification of noisy or vague data, a trait which would seem to be useful in a speech recognizer. In addition, since speech really isn't a true first-order Markovian process in which each frame only depends on the previous one, recognizers have been built using neural networks as their main method of classification.

A neural network, also known as a connectionist model, comprises numerous small units, sometimes called neurons, which compute an output based on the inputs given. In a multi-layer perceptron (MLP), neurons are arranged in layers. Each neuron receives as input the outputs of all the neurons from the previous layers, performs a weighted average on the inputs, runs the result through a "squashing function" which constrains the result to 0 – 1 or -1 – 1, and passes its result to all the neurons of the next layer. Eventually, after the processing of the input by the multiple layers, the topmost layer's outputs are considered to be the outputs of the neural network.

The outputs of the neural network are classes, into one of which the input falls. Neural networks can be trained to discriminate between classes using learning data and a procedure called error back-propogation. In error back-propogation, the input signals are applied to the inputs of the neural network, and the outputs of the network are compared to the desired results. Using the weights of the connections, any error is traced back through the net, and corrections are made to the weights of the connections. For more information on neural networks, see [2] and [7].

An excellent example of a neural network-based recognizer is a speech recognition engine developed at the Oregon Graduate Institute [12], designed for speech recognition over the telephone system. The system works [3] by dividing up the voice input stream into 10 msec frames, each overlapping the previous frame by 4 msec. A Perceptual Linear Predictive [6] is performed on each frame, and the seven seventh-order coefficients are used as inputs.

The system actually has two neural networks [3]. The first one is trained to determine if the phoneme being spoken in the current frame is voiced. It takes as inputs the PLP parameters and the energy of the current frame, and of the three frames both preceding and following the current frame. It produces two outputs: voiced and non-voiced.

Two variations of the second neural network were tried [3]. The first variation took as inputs the PLP parameters, the energy, and the voiced/unvoiced state of the current frame and the three frames preceding and following the current frame. It was split into three sets of outputs at the output layer. Each of the phonemes (the chosen building block of this recognizer) was split into three parts—beginning, middle, and end. One output set was trained on the probabilities that the first part of the phoneme was spoken, the second set on the probabilities of the middle of the phoneme, and the third set on the probabilities of the ends of the phonemes.

The second version of the classification neural network took the same inputs but used seven sets of outputs [3]. Taking into account variations of phonemes in different contexts, there was one class for probabilities of the middle of the phoneme. Three of

the output groups calculated probabilities of the beginnings of the phonemes in three different phonemical contexts, and the last three groups calculated probabilities of the ends of the phonemes in three different phonemical contexts.

The system using the second neural network with seven output groups achieved a high recognition rate [3]. Armed with the sequence of spoken phonemes, a search algorithm is used to locate the word in the dictionary most likely fitting the sequence.

Sensory, Inc.'s One Chip Recognizer

Speech recognition is starting to be used in a wide variety of places. An interesting product, aimed at quite a different market than the previously mentioned systems, is Sensory, Inc.'s RSC-164 chip [11]. Using a neural network, it can provide speaker-independent recognition of 15 alternatives, speaker-dependent recognition of 60 alternatives after training, speaker verification, and keyword spotting in continuous speech. There is 64K of ROM available, but only 384 bytes of RAM [11], so memory use obviously must be minimized.

For speaker-independent recognition [11], the input data is segmented, filtered, normalized with respect to time, and acoustic features are extracted. The resulting data is fed to a neural network which then classifies the input into one of several output categories, including a category indicating no match. Due to the low memory constraints, the neural network not built or trained on the chip itself.

For speaker-dependent recognition [11], "templates" are built from the training data spoken by the user. When in use, a neural network takes as inputs the templates, one at a time, and the spoken word, and outputs whether or not they match. Speaker verification works on similar principles, with a few changes.

Project LISTEN's Emily

Emily is a computerized reading coach designed at Project LISTEN [13] at Carnegie Mellon University. Emily [10] takes as input the text of a story as well as the real-time reading of it by a child, and has the goal of detecting mispronounced and skipped words. The recognizer system must, when given a sentence and a reading of it, must detect the mispronounced and skipped words, detect the end of spoken fragments of

text, and detect when the reader is stuck.

Emily takes simple approaches to solving those three tasks [10]:

> Emily consists of two basic components—an *intervenor* ... and a *speech recognizer* .... The intervenor tells the recognizer where in the text to start listening—either at the beginning of a new sentence, after a word spoken by the coach, or at a word the coach has just prompted the reader to reread. Four times a second, the recognizer reports the sequence of words it thinks it has heard so far. Capability 1 is implemented by aligning the output of the recognizer against the text. Capability 2 is implemented by checking if the recognizer has output the last word of the fragment. Capability 3 is implemented by a time limit for progressing to the next word in the text; the intervenor assumes that the reader is stuck on this word if the time limit is exceeded without a previously unread word appearing in the recognizer output.

Emily uses [10] the Sphinx-II recognizer, and uses a set of hidden Markov models as the basis for the recognizer's phonetic recognition. The pronunciations of the words are looked up in a dictionary or computed. The model of the sentences have to take into account the disfluent reading of the users, so each word—in the model—has a 97% chance of being followed by the correct word while false starts, near misses, repetitions and omissions are modeled as the other 3%.

Air Travel Information System

The Air Travel Information System [1] is a system designed to take and process voice queries of airline flights. The HMM-based DECIPHER system is used for recognition [8]. ATIS is set up to use one of two different post-processing systems to achieve its desired results. The first is a template-matching system—there are eight templates which specify criteria for searching the database. The engine wades through the recognized words, and fills in the templates as best it can, and runs the query. The alternative engine is the grammar parser. Given a grammar of English, the parser attempts to deconstruct the recognized sentence and recognize the parts of the sentences that are the query. It is a bit slower but can start processing before the entire sentence is recognized.

SpeechWear

SpeechWear, developed at Carnegie Mellon, uses speech recognition technology to replace alternative input devices in situations when both hands of the user are busy. SpeechWear [15] was developed as an alternative input device for the VuMan project, a wearable computer currently in use for inspections. Speech is recognized using the Sphinx-II recognition engine, which uses HMMs for word modeling.

The inspection database is set up to run on a standard HTTP server [15] with CGI links to the database. A special version of the Mosaic web browser runs on the computer, and recognizes custom tags in the HTML document that provide for the filling out of standard forms using voice commands. The error rate was approximately 15%, of which a significant amount was due to a microphone of insufficient quality and too little training.

Conclusion

Even though the only speaking most people do to their computers is due to frustration and displeasure, the state of speech recognition is such that soon it will be a feasible option. Speech recognition promises to open up new opportunities for computers and new ways for users to use them.

References

1. Air Travel Information System, http://www.speech.sri.com/demos/atis.html

2. Aleksander, Igor and Helen Morton. *An Introduction to Neural Computing*. Second Edition. Thompson Computer Press, London, 1995.

3. Barnard, Etienne, Ronald Cole, Mark Fanty, Pieter Vermeulen. "Real-world speech recognition with neural networks." *Proceedings of the International Symposium on Aerospace/Defense Sensing & Control and Dual-Use Photonics*, International Society for Optical Engineering, Technical Conference no. 2492. Available from ftp://speech.cse.ogi.edu/pub/nsf_arpa95/nspie.ps.Z

4. Bengio, Yoshua. *Neural Networks for Speech and Sequence Recognition*. International Thomson Computer Press, London, 1996.

5. Bourland, Hervé A., and Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, Boston, 1994.

6. Hermansky, H. "Perceptual Linear Predictive PLP Analysis for Speech." *Journal of the Acoustical Society of America*. 87(4), 1738–1752.

7. Hertz, John, *et al*. *Introduction to the Theory of Neural Computation*. Addison-Wesley Publishing Company, Redwood City, California, 1991.

8. Julia, Luc, Adam Cheyer, Leonardo Neumeyer, John Dowding, Mehdi Charafeddine. "HTTP://WWW.SPEECH.SRI.COM/DEMOS/ATIS.HTML." *Proceedings of the AAAI' 97*. Available from ftp://ftp.speech.sri.com/pub/papers/aaai97.ps.gz

9. Keller, Eric. "Fundamentals of Phonetic Science." In *Fundamentals of Speech Synthesis and Speech Recognition*, Keller, Eric ed. John Wiley & Sons, Chichester, England, 1994.

10. Mostow, Jack, Steven F. Roth, Alexander G. Hauptmann, and Matthew Kane. “A Prototype Reading Coach that Listens.” *Proceedings of the Twelfth National Conference on Artificial Intelligence* (AAAI-94), American Association for Artificial Intelligence, Seattle, WA, August 1994, pp. 785-792. Available from http://www.cs.cmu.edu/afs/cs.cmu.edu/user/hcii/www/project_listen/aaai94-online.ps

11. Mozer, Michael C. “Neural Network Speech Processing for Toys and Consumer Electronics.” Available from ftp://ftp.cs.colorado.edu/users/mozer/papers/speech.ps.Z

12. Oregon Graduate Institute, http://www.ogi.edu/

13. Project LISTEN, http://www.cs.cmu.edu/afs/cs.cmu.edu/user/hcii/www/project_listen/

14. Rabiner, Lawrence R. “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.” *Proceedings of the IEEE*, 77, 257-286.

15. Rudnicky, Alexander I., Stephen D. Reed, Eric H. Thayer. “SpeechWear: A Mobile Speech System.” Available from http://www.speech.cs.cmu.edu/rspeech-1/air/papers/speechwear.ps

16. Torkkola, Kari. “Stochastic Models and Artificial Neural Networks for Automatic Speech Recognition.” In *Fundamentals of Speech Synthesis and Speech Recognition*, Keller, Eric ed. John Wiley & Sons, Chichester, England, 1994.

Yes, this paper is mine—I wrote it myself without assistance of any kind. Despite the pain, suffering, and anguish endured during its creation, I did not plagiarize *anything at all*, and all material I used from outside sources is properly attributed to the best of my knowledge and ability. If swallowed: Drink a glass of water and call a physician. Contains no phosphorus.

_____

Avi Drissman