

Lecture 12: An Overview of Speech Recognition

1. Introduction

We can classify speech recognition tasks and systems along a set of dimensions that produce various tradeoffs in applicability and robustness.

Isolated word versus continuous speech: Some speech systems only need identify single words at a time (e.g., speaking a number to route a phone call to a company to the appropriate person), while others must recognize sequences of words at a time. The isolated word systems are, not surprisingly, easier to construct and can be quite robust as they have a complete set of patterns for the possible inputs. Continuous word systems cannot have complete representations of all possible inputs, but must assemble patterns of smaller speech events (e.g., words) into larger sequences (e.g., sentences).

Speaker dependent versus speaker independent systems: A speaker dependent system is a system where the speech patterns are constructed (or adapted) to a single speaker. Speaker independent systems must handle a wide range of speakers. Speaker dependent systems are more accurate, but the training is not feasible in many applications. For instance, an automated telephone operator system must handle any person that calls in, and cannot ask the person to go through a training phase before using the system. With a dictation system on your personal computer, on the other hand, it is feasible to ask the user to perform a hour or so of training in order to build a recognition model.

Small versus vocabulary systems: Small vocabulary systems are typically less than 100 words (e.g., a speech interface for long distance dialing), and it is possible to get quite accurate recognition for a wide range of users. Large vocabulary systems (e.g., say 20,000 words or greater), typically need to be speaker dependent to get good accuracy (at least for systems that recognize in real time). Finally, there are mid-size systems, on the order to 1000-3000 words, which are typical sizes for current research-based spoken dialogue systems.

Some applications can make every restrictive assumption possible. For instance, voice dialing on cell phones has a small vocabulary (less than 100 names), is speaker dependent (the user says every word that needs to be recognized a couple of times to train it), and isolated word. On the other extreme, there are research systems that attempt to transcribe recordings of meetings among several people. These must handle speaker independent, continuous speech, with large vocabularies. At present, the best research systems cannot achieve much better than a 50% recognition rate, even with fairly high quality recordings.

2. Speech Recognition as a “Tagging” Problem,

Speech recognition can be viewed as a generalization of the tagging problem: Given an acoustic output $A_{1,T}$ (consisting of a sequence of acoustic events a_1, \dots, a_T) we want to find a sequence of words $W_{1,R}$ that maximizes the probability

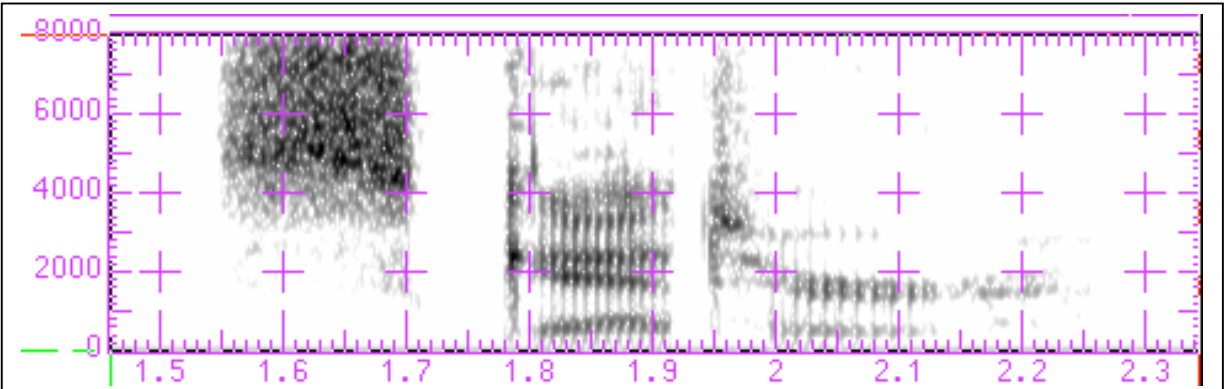


Figure 1: A Spectrogram of the word “sad”

$$\text{Argmax}_w P(W_{1,R} | A_{1,T})$$

Using the standard reformulation for tagging by rewriting this by Bayes Rule and then dropping the denominator since it doesn't affect the maximization, we transform the problem into computing

$$\text{argmax}_w P(A_{1,T} | W_{1,R}) P(W_{1,R})$$

In the speech recognition work, $P(W_{1,R})$ is called the **language model** as before, and $P(A_{1,T} | W_{1,R})$ is called the **acoustic model**.

This formulation so far, however, seems to raise more questions than answers. In particular, we will address the main issues briefly here and then return to look at them in detail in the following chapters.

What does a speech signal look like?

Human speech can be best explored by looking at the intensity at different frequencies over time. This can be shown graphically in a **spectrogram** of a signal containing one word such as the one shown in Figure 1. Time is on the X axis, frequency is on the Y axis, and the darkness of the area corresponds to intensity. This word starts out with a lot of high frequency noise with no noticeable lower frequencies or resonance (typical of an /s/ or /sh/ sound), starting at time 1.55, then at 1.7 there is a period of silence. This initial signal is indicative of a “stop” consonant such as a /t/, /d/, /p/. Between 1.8 and 1.9 we see a typical vowel, with strong lower frequencies and distinctive bars of resonance called the **formants** clearly visible. After 1.9 it looks like another stop consonant, that includes an area with lower frequency and resonance.

The Sounds of Language

The sounds of language are classified into what are called **phonemes**. A phoneme is minimal unit of sound that has semantic content. e.g., the phoneme AE versus the phoneme EH captures the difference between the words “bat” and “bet”. Not all acoustic

changes change meaning. For instance, singing words at different notes doesn't change meaning in English. Thus changes in pitch does not lead to phonemic distinctions.

Often we talk of specific features by which the phonemes can be distinguished. One of the most important features distinguishing phonemes is **Voicing**. A voiced phoneme is one that involves sound from the vocal chords. For instance, F (e.g., "fat") and V (e.g., "vat") differ primarily in the fact that in the latter the vocal chords are producing sound, (i.e., is **voiced**), whereas the former does not (i.e., it is **unvoiced**).

Here's a quick breakdown of the major classes of phonemes.

Vowels

Vowels are always voiced, and the differences in English depend mostly on the formants (prominent resonances) that are produced by different positions of the tongue and lips. Vowels generally stay steady over the time they are produced. By holding the tongue to the front and varying its height we get vowels IY (beat), IH (bit), EH (bat), and AE (bet). With tongue in the mid position, we get AA (Bob - Bahb), ER (Bird), AH (but), and AO (bought). With the tongue in the Back Position we get UW (boot), UH (book), and OW (boat). There is another class of vowels called the **Diphongs**, which do change during their duration. They can be thought of as starting with one vowel and ending with another. For example, AY (buy) can be approximated by starting with AA (Bob) and ending with IY - (beat). Similarly, we have AW (down, cf. AA W), EY (bait, cf. EH IY), and OY (boy, cf. AO IY).

Consonants

Consonants fall into general classes, with many classes having voiced and unvoiced members.

Stops or Plosives: These all involve stopping the speech stream (using the lips, tongue, and so on) and then rapidly releasing a stream of sound. They come in unvoiced, voiced pairs: P and B, T and D, and K and G.

Fricatives: These all involves "hissing" sounds generated by constraining the speech stream by the lips, teeth and so on. They also come in unvoiced, voiced pairs: F and V, TH and DH (e.g., thing versus that), S and Z, and finally SH and ZH (e.g., shut and azure).

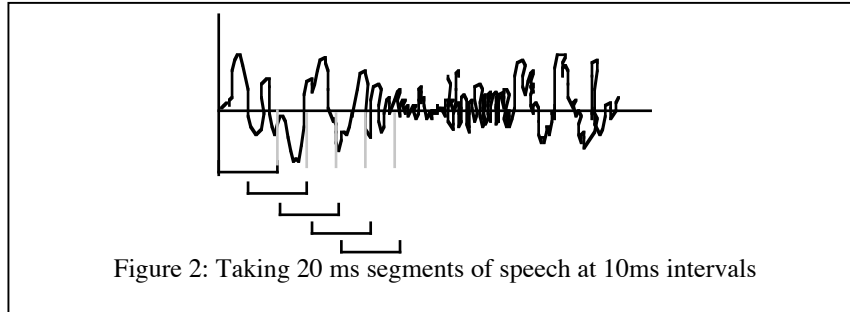
Nasals: These are all voiced and involve moving air through the nasal cavities by blocking it with the lips, gums, and so on. We have M, N, and NX (sing).

Affricatives: These are like stops followed by a fricative: CH (church), JH (judge)

Semi Vowels: These are consonants that have vowel-like characteristics, and include W, Y, L, and R.

Whisper: H

This is the complete set of phonemes typically identified for English.



How do we map from a continuous speech signal to a discrete sequence $A_{1,T}$?

One of the common approaches to mapping a signal to discrete events is to define a set of symbols that correspond to useful acoustic features of the signal over a short time interval. We might first think that phonemes are the ideal unit, but they turn out to be far too complex and extended over time to be classified with simple signal processing techniques. Rather, we need to consider much smaller intervals of speech (and then, as we see in a minute, model phonemes by HMMs over these values). Typically, the speech signal is divided into very small segments (e.g., 20 milliseconds), and these intervals often overlap in time as shown in Figure 1. These segments are then classified into a set of different types, each corresponding to a new symbol in a vocabulary called the **codebook**. Speech systems typically use a vocabulary size between 256 and 1024. Each symbol in the codebook is associated with a vector of acoustic features that defines its prototypical acoustic properties. Rather than trying to define the properties of codebook symbols by hand, clustering algorithms are used on training data to find a set of properties that best distinguish the speech data. To do this, however, we first need to design a representation of the speech signal as a relatively small vector of parameters. These parameters must be chosen so that they capture the important distinguishing characteristics that we use to classify different speech sounds. One of the most commonly used features record the intensity of the signal over various bands of frequencies. The lower bands help distinguish parts of voiced sounds, and the upper help distinguish fricatives. In addition, other features capture the rate of change of intensity from one segment to the next. These “delta” parameters are useful for identifying areas of rapid change in the signal, such as at the points where we have stops and plosives. Details on specific useful representations and the training algorithms to build prototype vectors for codebook symbols will be presented in a later lecture.

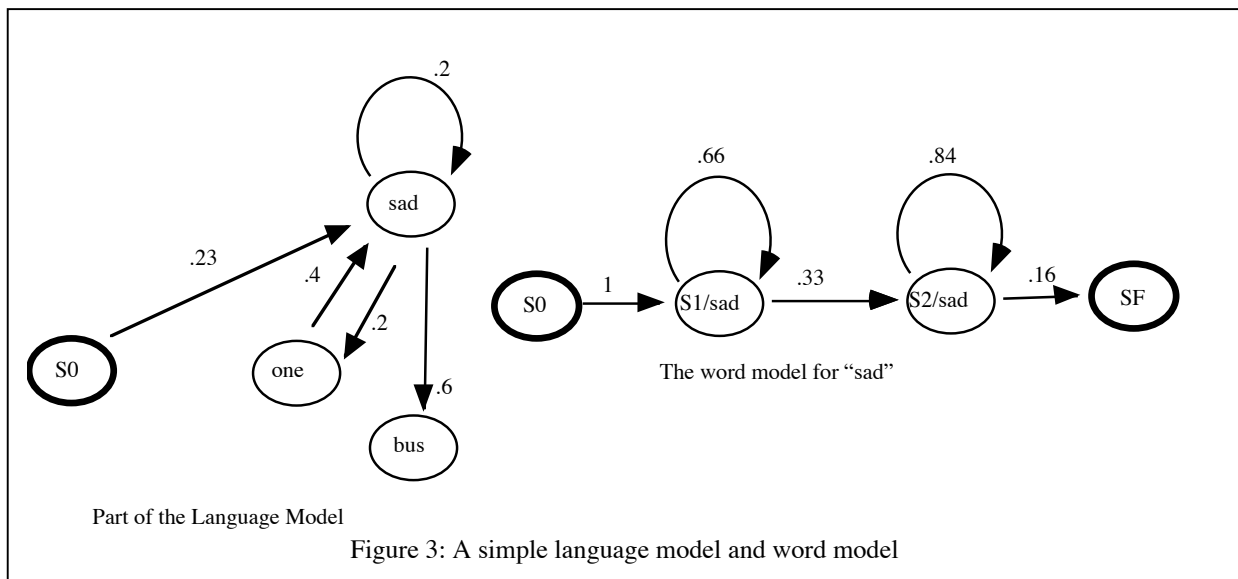
How do we map from acoustic events (codebook symbols) to words?

Since there are many more acoustic events than there are words, we need to introduce intermediate structures to represent words. Since words will correspond to codebook symbol sequences, it makes sense to use an HMM for each word. These word HMMs then would need to be combined with the language model to produce an HMM model for sentences. To see this technique, consider a highly simplified HMM model for a word such as “sad”, and a bigram language model for a language that contains the words “one”, ”sad” and “but”, as shown in Figure 3. We can construct a sentence HMM by simply inserting the word HMM’s in place of each word node in the bigram model. Figure 4 shows an HMM where we have substituted just the word “sag”. To complete the process we’d do the same from *bus* and *one*. Note that the probabilities from the bigram model are used now to move between the word networks.

How can we train large vocabularies? Applications: Sub Word Models

When we are working with applications of a few hundred words, it is often feasible to obtain enough training data to train each word model individually. As applications grow into the thousands of words, or tens of thousands, it becomes impractical to obtain the necessary training data. Say we need a corpus that has at least 5 instances of each word in order to train reasonably robust patterns. Then for a ten thousand word vocabulary we would need well over 50,000 training words as the words will not be uniformly distributed. Besides being impractical to collect such data, this method would lose all the generalities that could be exploited between words that contain similar sound patterns, and make it very difficult to design systems that could adapt to a particular speaker. Thus, large-vocabulary speech recognition systems do not use word based models, but rather use **subword models**.

The idea is that we build HMM models for parts of words, which can then be reused in different word models. There is a range of possibilities for what to base the subword units on, including the following:



- phoneme-based models;
- syllable-based models;
- demi-syllable based models, where each sound represents a consonant cluster preceding a vowel, or following a vowel;
- phonemes in context, or triphones: context dependent models of phonemes dependent on what precedes and follows the phoneme;

There are advantages and disadvantages of each possible subword unit. Phoneme-based models are attractive since there are typically few of them (e.g., we only need 40 to 50 for English). Thus we need relatively little training data to construct good models.

Unfortunately, because of co-articulation and other context dependent effects, pure phoneme-based models do not perform well. Some tests have shown they have only half the word accuracy of word based models.

Syllable-based models provide a much more accurate model, but the prospect of having to train the estimated 10,000 syllables in English makes the training problem nearly as difficult as the word based models we had to start with. In a syllable based representation, all single syllable words such as *sad*, would of course be represented identically to the word based models.

Demi-syllables produces a reasonable compromise. Here we'd have models such as /B AE/ and /AE D/ combining to represent the word *bad*. With about 2000 possibilities, training would be difficult but not impossible.

Triphones (or **phonemes in context, PIC**) produce a similar style of representation to demi-syllables. Here, we model each phoneme in terms of its context (i.e., the phones on each side). The word *bad* might break down into the triphone sequence $SIL B_{AE} B_{AE} D_{AE} D_{SIL}$, i.e., a B preceded by silence and followed by AE, then AE preceded by B and followed by D, and then D preceded by AE and followed by silence. Of course, in continuous speech, we might not have a silence before or after the word, and we'd then have a different context depending on the words. For instance, the triphone string for the utterance *Bad art* might be $SIL B_{AE} B_{AE} D_{AE} D_{AA} D_{AA} R_{AA} R_{AA} T_{RT} T_{SIL}$. In other words, the last model in *bad* is a D preceded by AE and followed by AA. Unfortunately, with 50 phonemes, we would have $50^3 (= 125,000!)$ possible triphones. But many of these don't occur in English. Also we can reduce the number of possibilities by abstracting away the context parameters. For instance, rather than using the 50 phonemes for each context, we use a reduced set including categories like silence, vowel, stop, nasal, liquid consonant, etc (e.g., we have an entry like \sqrt{T}_V , a /T/ between two vowels, rather than $AE T_{AE}$, $AE T_{AH}$, $AE T_{EH}$, and so on).

Whatever subword units are each, each word in the lexicon is represented by a sequence of units represented as a Finite State Machine or a Markov model. We then have three levels of Markov and Hidden Markov models which need to be combined into a large network. For instance, Figure 4 shows the three levels that might be used for the word *sad* with a phoneme-based model

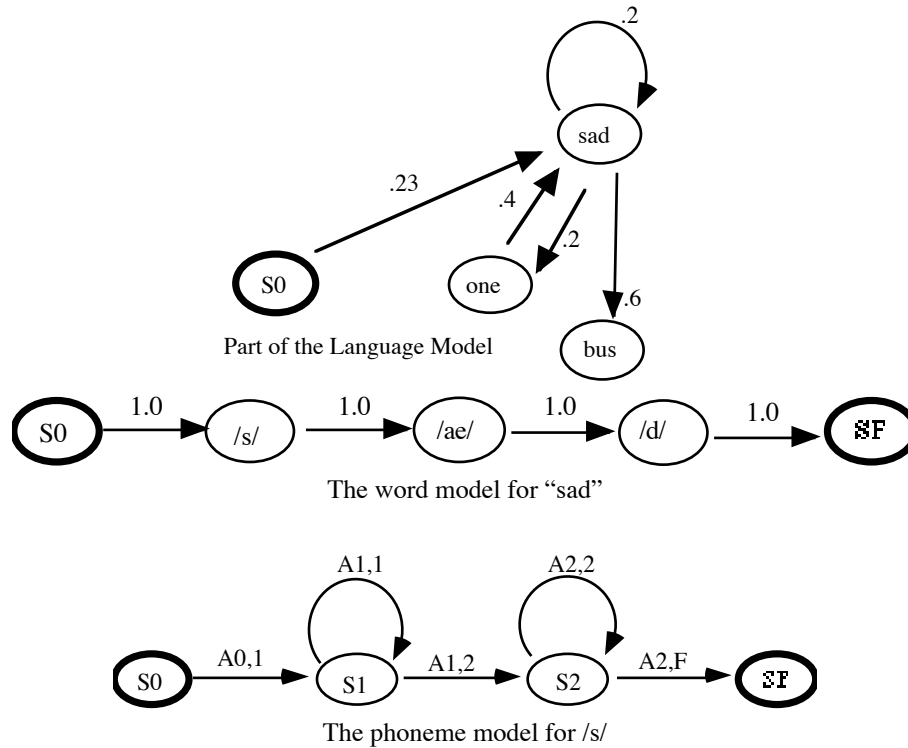


Figure 4: Three Levels in a Phoneme-based model

To construct the exploded HMM, we would first replace the phoneme nodes in the word model for *sad* with the appropriate phoneme models as shown in Figure 5, and then use this network to replace the node for *sad* in the language model as shown in Figure 6. Once this is down for all nodes representing words, we would have single HMM for recognizing utterances using phoneme based pattern recognition.

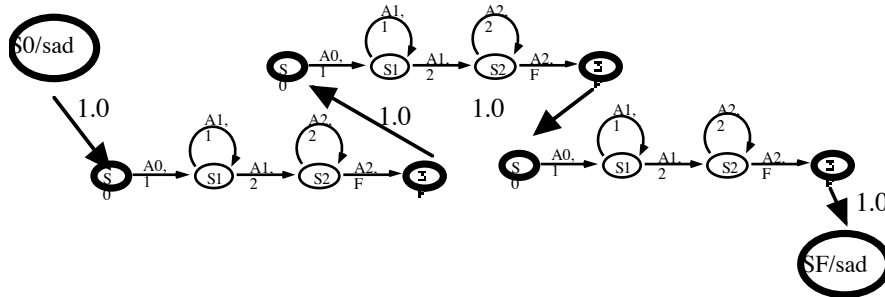


Figure 5: The constructed word model for *sad*

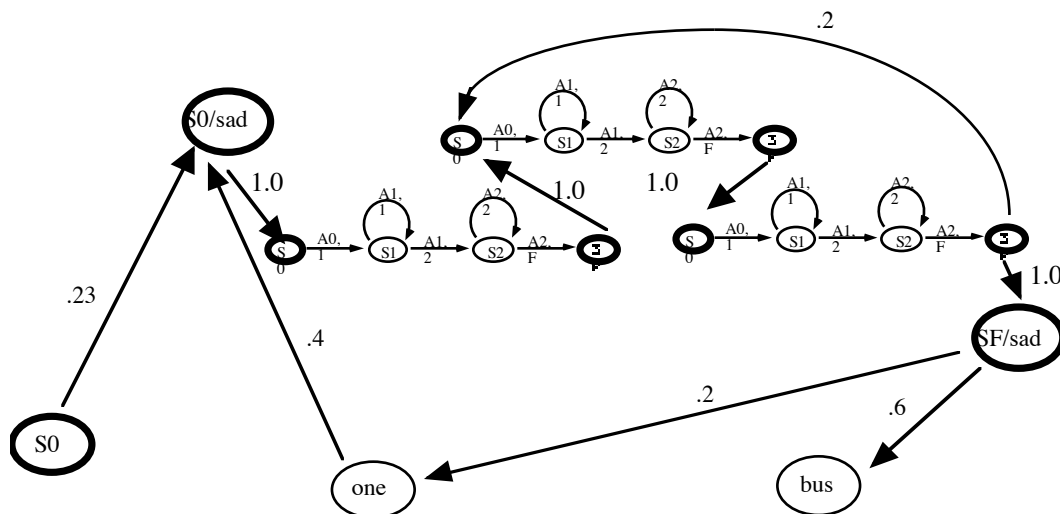


Figure 6: The language model with *sad* expanded

How do we find the word boundaries?

We have solved this problem, in theory at least. If we can run a Viterbi algorithm on a network such as that in Figure 6, the best path will identify where the words are (and in fact provide a phonemic segmentation as well).

How can we train and search such a large network?

While we have already examined methods for searching and training HMMs, the scale of the networks used in speech recognition are such that some additional techniques are needed. For instance, the networks are so large that we need to use a beam or best-first search strategy, and often will build the networks incrementally as the search progresses. This will be discussed in a future chapter.