Automatic Speech Recognition with HTK

Automatic Speech Recognition with HTK

A Tutorial for the Course *Computational Intelligence*

http://www.igi.tugraz.at/lehre/CI

Barbara Resch (minor changes by Erhard Rank)
Signal Processing and Speech Communication Laboratory
Inffeldgasse 16c

Abstract

This tutorial introduces the main components for an automatic speech recognition system. The acoustic information is sampled as a signal suitable for processing by computers (a digital signal) and fed into a recognition process. The output of the system is a hypothesis for a transcription of the speech signal. The Hidden Markov Model Toolkit (HTK)[1] [5] is used for building and manipulating hidden Markov models, being the core of most state-of-the-art speech recognition systems. HTK is used within this tutorial to build a simple speech recognizer.

Usage

To make full use of this tutorial you have to

1. download the file `ASR.zip` which contains this tutorial (PDF and ps.gz) and an introduction of the HTK (PDF and ps.gz).
2. The necessary HTK programs and data files are available from the homework assignment page.

1 Automatic Speech Recognition

In automatic speech recognition (ASR) systems acoustic information is sampled as a signal suitable for processing by computers and fed into a recognition process. The output of the system is a hypothesis for a transcription of the utterance. Speech recognition is a complicated task and state-of-the-art recognition systems are very complex. There are a big number of different approaches for the implementation of the components. For further information the reader is referred to [1, 2, 3]. Here we only want to provide an overview over ASR, some of its main difficulties, the basic components, their functionality and interaction.

1.1 Components of ASR

Figure 1 shows the main components of an ASR system. In the first step, the *Feature Extraction*, the sampled

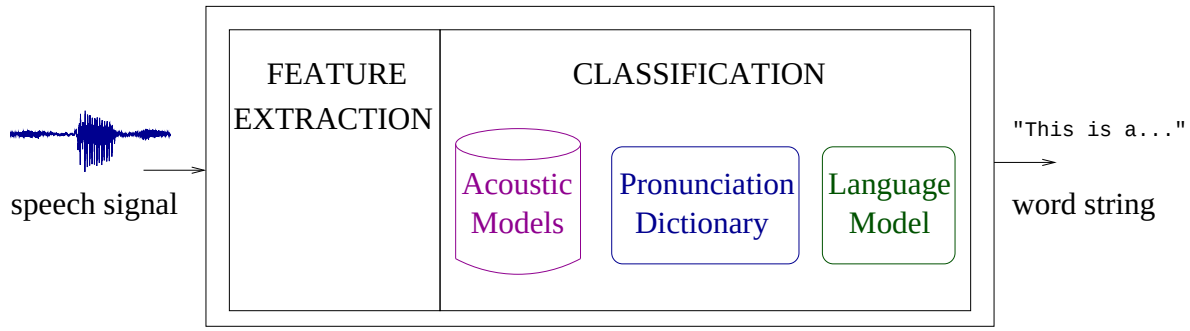

Figure 1: Principle components of an ASR system

speech signal is parameterized. The goal is to extract a number of parameters ('features') from the signal that

[1] http://htk.eng.cam.ac.uk/

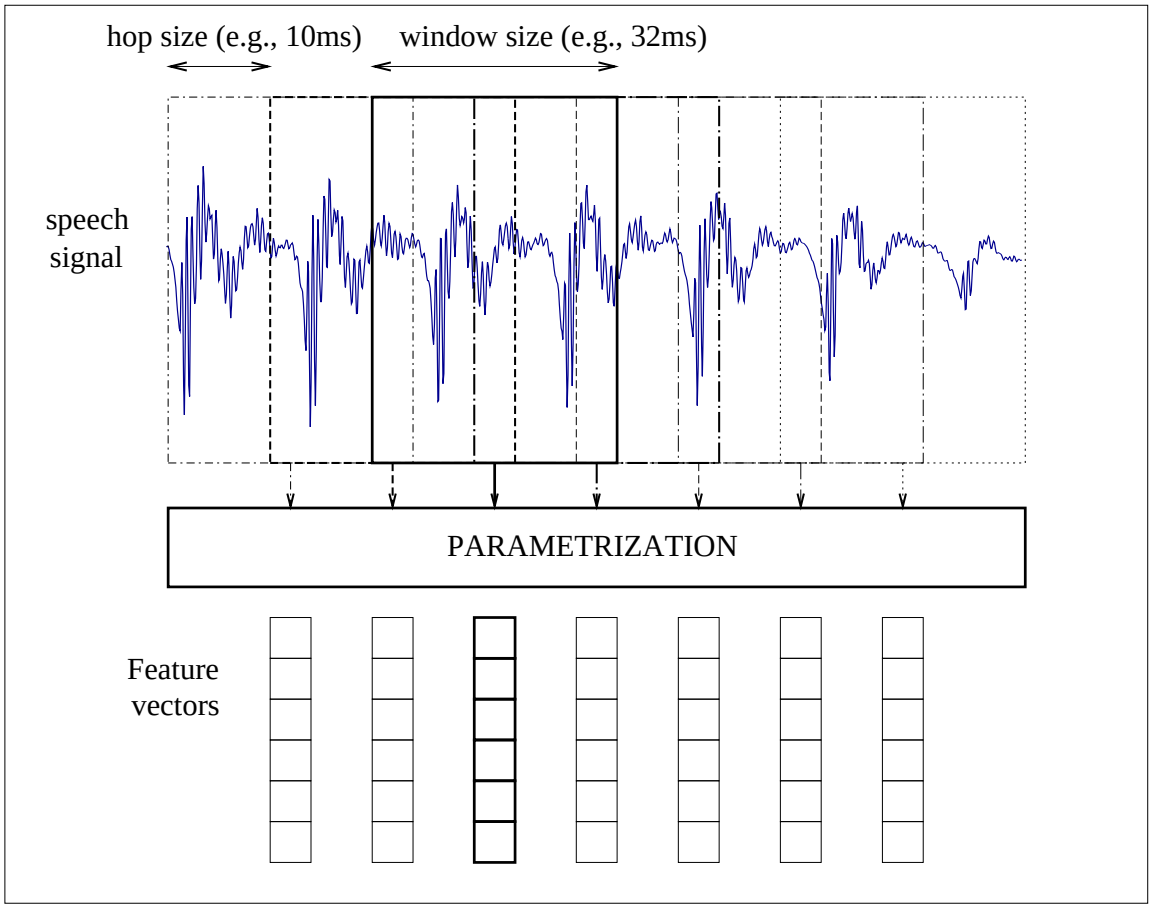

Figure 2: Feature extraction from a speech signal. Every 'hop-size' seconds a vector of features is computed from the speech samples in a window of length 'window-size'.

have a maximum of information relevant for the following classification. That means features are extracted that are *robust to acoustic variation* but *sensitive to linguistic content*. Put in other words, features that are discriminant and allow to distinguish between different linguistic units (e.g., phones) are required. On the other hand the features should also be robust against noise and factors that are irrelevant for the recognition process (e.g., the fundamental frequency of the speech signal). The number of features extracted from the waveform signal is commonly much lower than the number of signal samples, thus reducing the amount of data. The choice of suitable features varies depending on the classification technique.

Figure 2 indicates how features (or feature vectors) are derived from the speech signal. Typically, a frequency-domain based parametrization is performed to extract the features. Spectral analysis is performed, e.g., every 10 ms on the speech samples in a window of, e.g., 32 ms length. The speech signal is regarded stationary in this time-scale. Although this is not strictly true, it is a reasonable approximation. For each frame a vector of parameters, the feature vector, is determined and handed to the next stage, the classification.

In the *classification* module the feature vectors are matched with reference patterns, which are called acoustic models. The reference patterns are usually Hidden Markov Models (HMMs) trained for whole *words* or, more often, for *phones* as *linguistic units*. HMMs cope with temporal variation, which is important since the duration of individual phones may differ between the reference speech signal and the speech signal to be recognized. A linear normalization of the time axis is not sufficient here, since not all phones are expanded or compressed over time in the same way. For instance, stop consonants ("d", "t", "g", "k", "b", and "p") do not change their length much, whereas the length of vowels strongly depends on the overall speaking rate.

The *pronunciation dictionary* defines which combination of phones give valid words for the recognition. It can contain information about different pronunciation variants of the same word. Table 1 shows an extract of a dictionary. The words (graphemes) in the left column are related to their pronunciation (phones) in the right column (phone symbols like in the table are commonly used for the English language).

The *language model* contains rudimentary syntactic information. Its aim is to predict the likelihood of specific words occurring one after another in a certain language. In a more formal description, the probability of the k-th word following the $(k-1)$ previous words is defined as $P(w_k|w_{k-1}, w_{k-2}, ..., w_1)$. In practice the context (number of previous words considered in the model) is restricted to $(n-1)$ words $P(w_k|w_{k-1}, w_{k-2}, ..., w_1) \approx$

Table 1: Extract from a dictionary

word	*pronunciation*
INCREASE	ih n k r iy s
INCREASED	ih n k r iy s t
INCREASES	ih n k r iy s ah z
INCREASING	ih n k r iy s ih ng
INCREASINGLY	ih n k r iy s ih ng l iy
INCREDIBLE	ih n k r eh d ah b ah l

$P(w_k|w_{k-1}, w_{k-2}, ..., w_{k-n+1})$, and the resulting language model is called n-gram model.

1.1.1 Sub-word modeling with HMMs

In large vocabulary ASR systems, HMMs are used to represent sub units of words (such as phones). For English it is typical to have around 40 models (phones). The exact phone set depends on the dictionary that is used. Word models can be constructed as a combination of the sub word models.

In practice, the realization of one and the same phone differs a lot depending on its neighboring phones (the phone 'context'). Therefore *context dependent* phone models are most widely used. *Biphone models* consider either the left (preceding) or right (succeeding) phone, in *triphone models* both neighboring phones are taken into account, and for each phone different models are used for a different context. In Figure 3, the English word "bat" [b ae t] is shown in a monophone, biphone and triphone representation. The underlying sub-models for the phones or their combinations (in the bi- and triphone case) are in most cases HMMs. Because of lack of enough occurrences of all the triphone combinations in the training data (a phonetic alphabet of 40 phones results in a number of $40^3 = 64000$ possible triphones) clustering techniques, e.g., by using binary regression trees, are often used to get more reliable models for rarely occurring phone combinations.

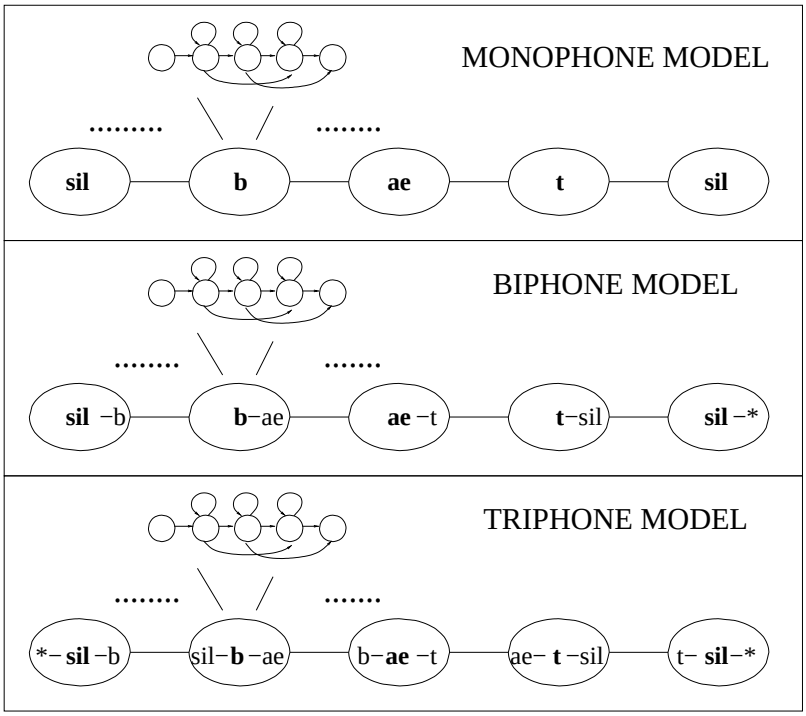

Figure 3: Monophone, biphone, and triphone HMMs for the English word "bat" [b ae t]. 'sil' stands for silence at the beginning and end of the utterance, which is modeled as a 'phone', too.

The information coming from the language model and acoustic models as well as the information from the pronunciation dictionary has to be balanced during speech recognition.

2 Hidden Markov Model Toolkit (HTK)

The Hidden Markov Model Toolkit (HTK) [5] is a portable toolkit for building and manipulating hidden Markov models. HTK is primarily used for research in speech recognition. HTK is in use at hundreds of sites worldwide.

2.1 Experiment - with HTK

We will demonstrate the HTK on a rather simple recognition experiment. The lexicon contains two entries: These are the German words 'ja' and 'nein'. We will first train the acoustic models (HMMs) with a set of training data, and then evaluate the HMMs on a distinct set of test utterances.

HTK tools are designed to run with a traditional command-line style interface. Each tool has a number of required arguments plus optional arguments. The speech data used for the following experiment is part of the SpeechDat-AT, a database for Austrian German [4].

2.1.1 Starting point

We need speech data to train the HMMs. These are saved in the Q/ directory. Furthermore, a dictionary is needed to define the valid words and their pronunciation (cf. table 1) for the recognition. The dictionary for our task is in file 'dict'. The file "phones0.mlf" contains the pronunciation of all the training data. We will use 2 different sets of training data, one set of about 40, the other of about 1000 utterances. The sets are defined in the files "train1.scp" (smaller set) and "train2.scp" (bigger set).

2.1.2 Step 0 - Feature Extraction

In the first stage of the ASR system the raw speech data (signal waveforms) are parametrized into sequences of feature vectors (cf. fig. 2). Since this is a somewhat time consuming task (and since the speech signal files are very large), this feature extraction has been done in advance for the training data set. The feature vector sequences for the training and test data set can be found in the directory Q/ in files SNNNN.mfc.

2.1.3 Step 1 - Training the Acoustical Models (HMMs)

The parameters of the HMMs are trained using the EM-algorithm. The emission pdfs are mixtures of Gaussians (cf. tutorial 'Mixtures of Gaussian'). The EM-algorithm is an iterative algorithm, each iteration is invoked in HTK with a call of the function `HERest`. To find initial parameters the HTK tool `HCompV` can be used. It scans the set of training feature files, computes the global mean and variance and sets the parameters of all pdfs in a given HMM to this mean and variance values.

The invocation of these 2 HTK tools is implemented in the perl program `train.pl`. Its first argument specifies the base names of the new directories that will be created to save the parameters of the trained HMMs. The second argument is the training file (use either train1.scp or train2.scp) and the third argument specifies the number of iterations which are performed to train the HMMs. For example, if you use the command `perl train.pl ABC train1.scp 2`, you will get 3 new directories called `ABC_hmm0` (with the initial HMM parameters), `ABC_hmm1` (the HMM parameters after the first iteration) and `ABC_hmm2` (the HMM parameters after the second and final iteration). In the command line window the commands used by the perl program to invoke the HTK tools are echoed.

2.1.4 Step 2 - Recognizing Test Data and Evaluation of the Recognition Result

The HTK Tool `HVite` is a general-purpose Viterbi word recognizer. It matches speech signals against a network of HMMs and returns a transcription for each speech signal. `HResults` is the HTK performance analysis tool. It reads in a set of label files (typically output from the recognition tool such as HVite) and compares them with the corresponding reference transcription.

The perl script `test.pl` first calls `HVite` to perform speech recognition and obtain a transcription of the test speech signals[2] (this may take a while!), and then HResult to compute recognition statistics, such as the percentage of correctly recognized words. Its first and only argument is the name of the directory where the HMM specifications are saved. For example, to use the trained models from the last subsection for recognition, type `perl test.pl ABC_hmm2` Again, the commands for calling the HTK functions are echoed in the command line window. Eventually, the recognition statistics are displayed.

In the recognition statistics, the first line gives the *sentence-level accuracy* based on the total number of transcriptions generated by the recognizer which are identical to the according reference transcriptions. The second line contains numbers concerning the *word accuracy* of the transcriptions generated by the recognizer. Here, H is the number of correct words, D is the number of deletions (words that are present in the reference

[2]The set of test signals is defined in file `test.scp`.

transcription, but are 'deleted' by the recognizer and do not occur in the recognizer's transcription), S is the number of substitutions (words in the reference transcription that are 'substituted' by other words in the recognizer's transscription), I is the number of insertions (words that are present in the recognizer's transcription but not in the reference), and N is the total number of words in the reference transcription.

The percentage of *correctly recognized words* is given by

$$\text{Correct} = \frac{H}{N} \times 100\%,$$

and the word recognition *accuracy* is computed by

$$\text{Accuracy} = \frac{H - I}{N} \times 100\%.$$

2.2 Do

- Train HMMs using `train.pl`. Try it with the small (train1.scp) and the large (train2.scp) data set. Try out different number of iterations.
- Test the trained models using `test.pl`.
- Write a report about all experiments you performed, state the chosen settings, the results, and your interpretations.
- Do you get an improvement using the larger training set? How many iterations of the EM algorithm are necessary, in the case of the small/large training set?
- If you wanted to build a yes/no recognizer, would you use monophone-models, as we did, or do you think whole-word models are more suitable? Give reasons!

References

[1] L. Rabiner, B. Juang. Fundamentals of Speech Recognition. *PTR Prentice Hall (Signal Processing Series), Englewood Cliffs NJ, 1993, ISBN 0-13-015157-2.*

[2] B. Gold, N. Morgan. Speech and audio signal processing: processing and perception of speech, and music. *John Wiley and Sons Inc., 1999, ISBN: 0471351547.*

[3] X. Huang, A. Acero, H. Hon. Spoken Language processing: A Guide to Theory, Algorithm and System Development. *Prentice Hall, Redmond, Washington, 2001.*

[4] M. Baum. SpeechDat-AT: A telephone speech database for Austrian German. Proceedings of the LREC Workshop "Very Large Telephone Databases (XLDB)", Athens, Greece, 2000. *http://www.inw.tugraz.at/spsc/PubSpeech0001.pdf*

[5] S. Young, G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. Woodland. The HTK Book. Revised for HTK Version 3.2 Dec. 2002. *http://htk.eng.cam.ac.uk/*