This document is also available in PDF[1] format.

Purpose: Introduce the processing of high-order HMMs and Hierarchical HMMs.

Material: Paper by Du Preez and Weber.

General: There are several factors determining the nature of HMMs. We have already discussed topology and the nature of the PDFs. Now we turn our attention to the order of the HMM as reflected in the Markov order assumption. Historically, very little work was done on this front. The typical approach was to extend the HMM order in equations governing specific algorithms such as the Viterbi or Forward-Backward algorithms (e.g. see equations 2-3, 2-4 and 2-5 in supplied notes). The approach we follow is just the opposite. Instead of increasing the order of the algorithm, we reduce the order of the model.

Topics:

- **High-order HMMs:**
  - **Representation:** The *ORder rEDucing (ORED)* algorithm, discussed in the accompanying notes, iteratively reduces any high-order HMM to a mathematically equivalent 1st-order version. This makes standard 1st-order algorithms applicable to high-order HMMs. The general idea of this reduction is quite simple: by replacing each pair of linked states with a new state, the structure in the model ”remembers” one extra time step of state information. One thus reduces the order of the model, but the resulting structure compensates for it to effectively maintain the original model order.
  - **Training:** Training high-order HMMs directly can be disastrous in terms of not only the computational resources necessary, but also the effect of local optima on the quality of the model. The *FIT* algorithm addresses both of these points quite effectively. The basic idea is to iteratively increase the order of the HMM by using the resultant HMM of the previous lower order to initialise the next.
  - **Topology:** High-order HMMs also are an interesting way to describe a required modelling capability in a model, thereby defining topology via a different route. See the sections of explicit context and duration modelling for more details.
- **Hierarchical HMMs:** It is a well-known technique to replace states in an HMM with whole sub-HMMs to build new bigger (flat) HMMs – this is typically used in recognisers, where words are build from phonemes and phrases from words. If however, you maintain a representation of the hierarchical structure, interesting possibilities arise. You can, for instance, build higher-order Hierarchical HMMs where the higher-order behaviour is restricted to a certain level in the hierarchy e.g. a 3rd-order model of how phonemes combine in a specific language where you restrict the higher-order behaviour to the between-phoneme level and leave the within-phoneme behaviour first-order (or whatever else you fancy). Training and using these beasts can still be done via the large one-level/flat HMM, as long as the proper structures are in place to relate its activity to its proper hierarchical components. Typically you will need to do fairly sophisticated sharing of HMM components in your software to implement an HHMM.

Task Only for PR813 students. (Combine with the previous HMM task and hand in at the next lecture):
Determine the first-order equivalent of the second-order version of the 3-state left-to-right HMM used in previous tasks. Then use it to redo the previous diphthong recognition task. Compare with your original results.

[1] `http://www.dsp.sun.ac.za/pr813/lectures/9_hmm_n/9_hmm_n.pdf`