Unifying Unit Selection and Hidden Markov Model Speech Synthesis

Paul Taylor

Engineering Department, University of Cambridge, Trumpington Street, Cambridge, CB2 1PZ, UK

Abstract

This paper presents a framework which can accommodate the two most widely used contemporary speech synthesis techniques, namely unit selection and hidden Markov models (HMMs). This is achieved by building a very general HMM where we have a network of states, each representing a single frame for a single unit. This network exactly mimics the behaviour of a unit selection system and is effectively memorising the data as an HMM. From this, we can merge states in the network so as to produce a synthesis system of any desired size. The paper discusses this technique as well as a statistical formulation of the join cost and a number of ways to represent the acoustic observations of the states.

index terms speech synthesis, unit selection, hidden markov models

1. Introduction

Our ongoing work is concerned with the formalisation of unit selection speech synthesis into a fully rigorous general statistical framework. This paper present work on one aspect of this, in which we have developed a generalised framework for both unit selection and hidden Markov model (HMM) synthesis. Our approach is effectively to expand the model topologies, distributions and training techniques for hidden Markov models, to the extent that we can show that unit selection and the normal type of HMM synthesis are special cases within our framework. The reason for proposing this unification is that we can build many more types of synthesizer within this more general framework, and choose synthesis techniques based on the desired mix of size, speed, naturalness and robustness.

2. Unit Selection Synthesis

The most popular unit selection algorithm is that proposed in Hunt and Black [1]. The text analysis part of the system produces a **specification**, which will shall assume is a list of linguistic items all from one **base type**, for example diphones, phones or syllables. The operation of the algorithm itself is independent of the type of base type used, so for purposes of exposition we shall assume we use phones. The algorithm operates on a database of **units**, where each unit is the same base type as in the specification.

Both the specification items and the database units are described by a **feature description**. At the very least this contains the phone identity of the item or unit, in addition features such as F0, phone context, stress and phrase position are also often present. We can think of the base type as the primary identifier and the features as the "fine detail" in the description of the units.

In a well designed system, we can expect to have at least one unit for every base type. Often we have more, such that we have to choose which of the possible units to use for each item in the specification. This is what gives the unit selection technique its name. We use the term **model** to refer to a collection of units which all share the same feature description. In general, the distribution of units within models is extremely uneven; in many cases there are no units at all for a particular feature description in which cases we say we have an empty model. (This use of the term "model" will become clearer once we have looked at HMM synthesis).

Unit selection works by considering the specification items one by one, assessing the suitability of the units which match these base types (the **candidates**) and then picking these best overall sequence. This corresponds to a global minimisation of two cost functions; a **target cost** which measures the "similarity" between a specification we have produced from out text-to-speech engine, and a given unit in the database; and a **join cost** which measures how well two units will join together. Synthesis is performed by finding the lowest cost path through the set of candidates using a dynamic programming algorithm.

3. Hidden Markov Model Synthesis

Hidden Markov model (HMM) synthesis [4] has been around for nearly as long as unit selection, but has gained considerably attention recently due to the increased speech quality of the best HMM systems.

At first glance, HMM synthesisers seem very much like HMM speech recognition systems, only used in reverse. A typical system has three states per phone, uses mel-scaled cepstral coefficients (MFCCs), their delta and acceleration coefficients, and uses context based models with tied states determined by decision tree clustering. One major difference however is that as we have to take prosody into account, we have to use a much broader definition of the notion of "context" in our models. In our unifying scheme, we can generalise the idea of HMM context into one of feature description so that we have one model for each unique feature description. This can result in many millions of potential models, of which only a few thousand will have been observed in the training data.

Synthesis is performed by generating a sentence level HMM using the individual model HMMs that match the specification. If a requested feature combination was unseen during training, the "next best" model is selected by means of the decision tree. Using the sentence level HMM, we next generate the most likely sequence of observation frames. At its simplest, this would just correspond to the means of each state, and as such would not produce particularly good speech. The key "trick" in the HMM synthesis is to make the observations obey the implied state dynamics given by the delta and acceleration coefficients. Firstly, this ensures that within a model the trajectory of a particular coefficient is nearly always continuously evolving; no coefficient jumps at state boundaries are seen. Secondly, these same dynamic constraints are applied at the state transition between phone models, thereby ensuring smooth phone-

to-phone transitions. This is particularly clever and avoids any notion of join cost in HMM synthesis.

4. Comparing the approaches

In "pure" unit selection synthesis; we make no deconstruction of the waveform; no signal processing is used in generation and synthesis is performed by stitching together waveform samples from a large inventory using the search algorithm. This is the strength of the approach; if we consider natural recorded speech as being perfect, then the less we manipulation we perform the less degradation we will impose, with the result that highly natural speech should be generated.

The main problem with unit selection synthesis is that it is in some sense inherently "fragile". When we have the units we want the synthesis can sound excellent, but it is a matter of chance whether we do, and if we don't the speech can degrade by an arbitrary amount. Likewise with joins; often the joins are completely imperceptible, but occasionally no good join is possible and poor speech results. These problems are fundamental to the technique. In addition, we have the "unsolved" (as opposed to "unsolvable") problem of training. While a huge number of training techniques have been proposed, none is perfect and most researchers would agree that there is considerable room for improvement here. Finally "pure" unit selection by its nature is expensive in terms of storage as we are required to store every database sample. Even with the best compression techniques, the size of such a database can be considerable.

HMM synthesis gets around many of these problems. The models used to generate the speech are inherently more robust as they have been trained on a representative number of examples. The training is automatic and based on sound mathematical principles (statistics), and because only a few hundred parameters are required for each model, the memory requirements can be orders of magnitude lower.

For systems where space is not a problem and for which there is plenty of training data, unit selection generally sounds much better. This is because the robustness comes from the statistical averaging that occurs during training and hence HMM synthesis is always in some sense "playing it safe". It is constrained to generate from the feature representation it is using (often MFCCs + f0) and because of this and the statistical nature of the generation often much of the fine detail of real speech is not synthesised adequately.

5. Unifying the approaches

At first glance; these seem to be completely different technologies; one based on stitching samples and using costs, the other based on statistics and generalisation. Putting aside trivial differences in terminology and so on, we see that in fact there are three main differences in the systems; namely **model topology**, **acoustic representations** and **training techniques**.

5.1. Model topologies

To show the general equivalence of unit selection and HMM synthesis, let us for sake of demonstration use the same signal processing approach for both. While "pure" unit selection uses unprocessed raw waveforms, many techniques use slightly more complicated representations, where for instance we have frames of waveforms (allowing PSOLA-like operations), LPC representations (allowing separate source and filter modifications) and so on. Hence there is nothing in the unit selection technique per se than insists we use waveforms. It is therefore straightforward to use the most popular HMM representations, namely MFCC + F0 in unit selection. Everything else is the same, but now the units pass on a sequence of MFCC + F0 frames to a simple signal processing module which generates the final waveform. Some quality is lost in this process and the speech becomes slightly buzzy but as we shall see this is not the only option for HMM synthesis and we shall return to the signal representation below.

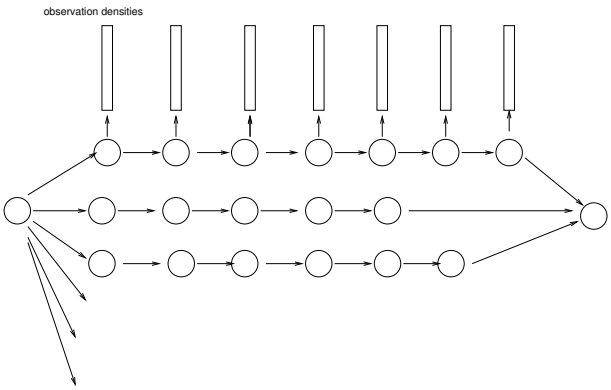

Figure 1: Full state HMM. Every unique path represents exactly one unit in the training data. Only the first three paths and the observation probabilities for the first path are shown in the diagram. In reality all states have one observation density and all units have one path.

Now let us consider the case of a feature description for which we have N matching units in our database. Each unit can be represented as a sequence of frames, and in turn we can depict this as a series of states, each corresponding to exactly one frame. The states are configured in a strict left to right topology, with no looping or skip states. If we now add a dummy state at the beginning and ends of each sequence, we can join at this points to construct a single network in which we immediately branch in one of N paths after the first state, follow that path to the end and then come back to a final end state. This is shown in Figure 1.

This network can be converted to a hidden Markov model by converting it into a statistical representation. First we assign a probability to each arc. For all except the first state this is 1.0 as there is no branching and only one path can be followed. The first state has N branches and so the probability for each of these is just $1/N$.

The MFCC + F0 frames associated with each state can be converted into observation probabilities with a mean and covariance. In each case, the mean is simply the values of the states. In a strict maximum likelihood sense we should set the covariances to be 0, as the state emits just exactly this and no other observation, but for practical purposes we set the covariances to be some small amount so as we can give some non-zero probability to any observation. As is common in HMM systems the near statistical independence of the MFCC parameters allows us to use diagonal covariances only.

We now in effect have an HMM formulation which will generate exactly the same speech as the unit selection system. In other words, the network has "memorised the data". The only difference between this and the standard synthesis HMM is the network topology with its much larger number of states.

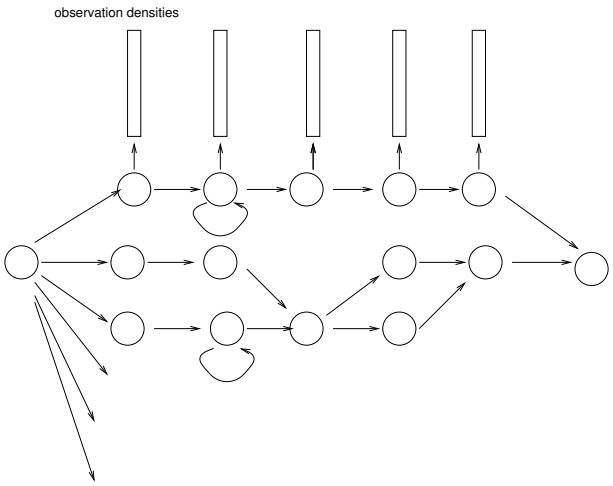

Figure 2: Example of how an HMM may look after four merging iterations.

5.2. Modifying the topology of the HMM system

We can now start to build a more general HMM from this "full state" HMM. This is done by a process of merging states, performed as follows.

1. Examine every possible pair of states and for each pair:
2. merge the two states into one new state such that:
   (a) the mean of the new state is the mean of the means of the old states
   (b) the variances are added
   (c) the transitions are combined so that the topology for the rest of the network is unchanged.
3. Recognise the training data using the Viterbi algorithm and record the likelihood
4. Choose the pair which produces the smallest decrease in likelihood.
5. Repeat this process until the stopping criteria is met.

In effect, we are seeking to merge the two most similar states, and we measure this by finding the network which gives the maximum likelihood of generating the training data. In other words, we are reducing the size of the network by one state, and examining all possible ways of doing this until we find the one that gives the maximum likelihood for the training data.

We can continue this process as far as we wish; in effect every time we merge two original states we are producing a probability density which summarises these two states: unlike the original case where we set the variance to 0 or some arbitrary small value, the merged states now have a "real" variance. In each iteration, the topology of the network changes. Loop and skip states (where a outgoing transition is made to the same, next or previous state) occur when two states in sequence are merged.

5.3. Stopping Criteria

We can stop the merging process at any stage; it should be clear that as we are starting with a model that has exactly memorised the training data the likelihood is always getting worse every time we merge two states, and so there is no objectively defined stopping point with respect to that. We could use model size; after all one of the main motivations may be to reduce the size of the unit selection database, and if we know in advance how big the required database should be then we can keep merging until that size has been achieved.

It is of course possible to keep on merging until we have the classic three state looping HMM; this is of course an expensive way to do this as we could have just started with this topology in the first place. The advantage of our technique is that any model size between the single frame-state model and the classic 3 state HMM can be created.

6. Calculating Join Costs via Frame Sequence Probabilities

Here we introduce the **frame sequence probability algorithm**, a means of replacing the traditional join costs by a purely statistical method. The join cost is normally not used in HMM synthesis; the models are simply concatenated and the trajectory generation algorithm is used to create smooth transitions across joins. One can argue that this is in fact divergent from the ASR HMM paradigm where there is a language model which determines probabilities between larger units. Exploring this idea, we have developed a purely statistical way of calculating the join "cost" (because it is statistical it is not longer a cost). This can be used in the model we have just developed, in normal HMM synthesis and even in pure unit selection synthesis.

While we can of course just use the standard HMM formulation and leave all join issues to the trajectory algorithm, our new join model seeks to provide a genuine probability that one section of speech will follow another. In doing this, we remove the dummy start and end states from all the models and join their real start and end states in an ergodic fashion. That is, if we have a model A_m with M real final states being joined to a model A_n with N real starting states, we construct a full $M \times N$ transition matrix at the boundary, and assign probabilities to each transition.

The normal way of calculating an acoustic join cost is to compare the frame at the end of the first unit with the frame of the second unit, and take use similarity measure to judge the degree of fit. Our approach is different and is based on the idea of studying naturally occurring sequences and using those as a model for what should constitute a good join.

Our probabilistic formulation is based on calculating the joint probability, $P(O) = P(o_1, o_2, o_3, ..., o_M)$, that a sequence of frames will occur. If we consider the probability of observing a frame of speech given all the proceeding frames we can use an approximation to this where we consider only the previous N frames. In its simplest form, we calculate this over two frames only:

$$P(O) = P(o_1, o_2, o_3, ..., o_M) \approx= \prod_{i=1}^{M} P(o_i|o_{i-1})$$

which can then be added into the network as the probability on the arc between the last state of the previous model and the first state of the next.

The attractive part about this formulation is that these probabilities are easy to train; in fact, all we have to do is study the sequences of naturally occurring frames in our training data and use this. Furthermore, we do no have to limit ourselves to units which occur in join locations; in fact we use *all* the frames in the entire corpus to calculate these probabilities. This is taken from the observation that every single naturally occurring frame in the database is a perfect join with its naturally occurring neighbour.

The specific formulation we use is in fact the same as the language model formulation in ASR. That is, we use a non-parametric model which simple counts (with smoothing) the occurrences of particular frames in sequence with other particular frames. We can only use this formulation with discrete entities, so as a first stage we vector quantise all the frames in the corpus. This gives is a codebook of "words", from which we then map the acoustic sequence $o_1, ...o_M$ to give a "word" sequences $w_1, ..., w_M$. From this it is then possible to count occurrences. As in language modelling, a back-off model is used to ensure that non-observed sequences have some small probability.

This formulation has a number of advantages over the usual acoustic join cost distance. In that, only frames which are acoustically similar get a low cost, but in real speech there are times when the spectrum is evolving very quickly, so that successive frames are quite different, but still entirely natural. The statistical formulation accounts for this and allows big acoustic differences across joins so long as that behaviour has been observed in the training data. Secondly, this technique is completely automatic, and doesn't rely on setting weights by hand or finding a space that is we hope is perceptually accurate. With care, it should be possible to train this model on virtually any speech database (not just the particular one being uses for synthesis), as we expect the evolution patterns to be common across all speech. The above model was estimated with about 1,000,000 data points (frames) but it should be possible to build more robust models that have longer contexts and bigger codebooks as we potentially have access to many thousands of times more data.

To evaluate this new technique we performed a simple test giving listeners pairs of utterances, where one utterance was synthesised using an acoustic distance measure and the other the new technique. In both cases we used MFCCs + F0 as the acoustic representation with delta and acceleration coefficients calculated also. The listeners were asked to say which version they preferred. In 38% of the cases the new technique was preferred, as against 12% for the old technique and 50% which had no preference.

7. Acoustic representations

For demonstration purposes, we used the representation most commonly use in standard HMM synthesis, namely MFCCs with F0. From this we can generate speech using the algorithm described in Imai [2]. The speech generated in this way has often been criticised as being somewhat buzzy, and this comes from the lack of an accurate source representation in all such models.

In fact we can accommodate nearly any type of acoustic representation in the observation densities and so far we have tried waveforms, magnitude spectra and full cepstra. In the full state HMM, all of these representations can be easily accomodated, but as we start to collapse states we find that certain representations are easier to model than others, mostly arising from issue of whether full or diaogonal covariances can be used. In the full state HMM, we can generate waveforms from the state means alone, and even after a few state merges this is still possible. However as we use fewer states, eventually the classic HMM synthesis problem arises where we start to generate one static section of speech, which then jumps to a different static section of speech as we move from one state to another. At this stage, the dynamic HMM synthesis algorithm of Tokuda et al [4] has been can be applied. We are still investigating whether this algorithm should be applied for all HMM topologies, including the full state case.

Our preferred measure is the normal, full cepstrum (i.e. without mel-scaling). Rather that smooth the DFT or discard the higher cepstral coefficients (the normal practice in ASR) we use the full cepstrum, as this retains the detailed source information and therefore allows us to accurately recreate the original waveform. This in general sounds better than waveforms generated by MFCCs and an F0 value, but requires many more parameters. To model this type of cepstrum, we perform a separation where we split the cepstrum into two streams, with the first having the spectral envelope MFCC representation as before, and the second having the higher part of the cepstrum representing the source information. Experiments have shown that a diagonal covariance matrix is not directly appropriate for this second stream, and current work is investigating the use of transforms to reduce the dimensionality of this source component so as to allow diagonal covariance modelling.

8. Conclusions and Further Work

The main findings in this paper are:

1. It is possible to reformulate unit selection synthesis within a generalised HMM framework.
2. This allows us to build to a fully scalable synthesiser
3. A number of acoustic representations for the observation densities are possible
4. It is possible to reformulate the join cost in unit selection as a transition probability between models, and train this on unlabelled sequences of naturally occurring speech.

Along with the target cost formulation described in [3], these techniques can be used to constitute a fully formal statistical speech synthesis framework that can encompass many of the unit selection and HMM synthesis techniques used today. There is considerable scope for further work, in particular in the areas of the join cost probability calculation and the acoustic representations of the states.

9. Acknowledgements

The author is supported by the Royal Society and he wishes to gratefully acknowledge that support. Thanks also go to many colleagues in the Machine Intelligence lab for useful discussions concerning this work.

10. References

[1] HUNT, A. J., AND BLACK, A. W. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP '96, Atlanta* (1996), IEEE, pp. 373–376.

[2] IMAI, S. Cepstral analysis synthesis on the mel frequency scale. In *ICASSP* (1983).

[3] TAYLOR, P. The target cost formulation in unit selection synthesis. In *Submitted to Interspeech 2006* (2006).

[4] TOKUDA, K., KOBAYASHI, T., AND IMAI, S. Speech parameter generation from hmm using dynamic features. In *ICASSP* (1995).