PERFECT SYNTHESIS FOR ALL OF THE PEOPLE ALL OF THE TIME

Alan W Black

Language Technologies Institute, Carnegie Mellon University
and Cepstral, LLC
awb@cs.cmu.edu

ABSTRACT

The quality of speech synthesis has drastically improved over the last ten years. Or at least it appears that this is the case. We have moved from diphones to unit selection. However, although we can produce much more natural sounding examples we have also given up an certain amount of control over what can be synthesized. We have reached the stage where playing a few examples to a non-expert can easily convince them that speech synthesis is a solved problem. This paper looks at how we might not only convince some of the people some of the time, but what we must do to produce perfect synthesis for all of the people all of the time.

1. UNIT SELECTION SYNTHESIS

The basic unit selection premise is that we can synthesize new naturally sounding utterances by selecting appropriate sub-word units from a database of natural speech.

There are lots of conditions that must be met in order for such a system to work. Let us consider the following basic notion of unit selection. Although this particular instantiation comes from [1] it will be generalized to help illustrate the space of the problems.

In [1] and in later, and earlier unit selection techniques [2], there is a notion of a **target cost**, how close a database unit is to desired unit, and a **join cost**, how well two adjacently selected units join together. The unit selection process is designed to optimally minimise both target and join costs.

More formally we can define the target cost C^t to be the weighted sum of differences of relevant features.

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i)$$

Various features have been proposed, typically encoding phonetic, metrical, and prosodic context.

In addition to selecting based on target cost we can define **continuity cost** as a weighted sum of difference of features

$$C^c(u_{i-1}, u_i) = \sum_{k=1}^{q} w_k^c C_k^c(u_{i-1}, u_i)$$

These two costs must then be optimized in order top find the string of units from the database that minimise the over all cost

$$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i) + C^c(S, u_1) + C^c(u_n, S)$$

Where S denotes silence and $C^c(S, u_1)$ and $C^c(u_n, S)$ address the conditions at the start and end of the utterance.

There has been, and will continue to be, a substantial amount of work in looking at what features should be used, and how to weight them. Getting the algorithms, measures and weights right will be key to consistent high quality synthesis. Looking at the amount of work and experiments done in the similarly complex field of speech recognition we can see we have still much to do, in spite of our successes.

It is interesting to note that in comparing current algorithms, theoretic advantages may be identified but it is not clear if these hold up in any reals sense due to the variation in databases, their labelling and the time one spends tuning the parameters. In fact "tuning the parameters" seems to be the most important factor in getting good consistent synthesis.

But what if, for now, we assume that we have the perfect features and the best weights. There are still factors outside these that affect the speech quality. It is those factors that will be discussed in the following section.

1.1. More data

First it is clear than having better data in the database will lead to better synthesis. In the simplest sense this means more speech data from the speaker. It is not coincidental that unit selection has become more prominent as the cost of storage has reduced. [3] correctly identifies this and recommends very large databases to improve the coverage with respect to synthesized data. With more data it is more likely that a database will contain a unit that is closer to the target and also more likely to have a better join.

But the problem of increasing the size of the database is that you never get enough data. There will always be holes

in the data because of the phenomenon of frequently occurring rare events in language [4]. For example if we were to collect all triphone contexts, with and without stressed vowels, and consonants in onset and coda (singletons and clusters), then wish to cover these for even a few phrasal conditions we very quickly note that the database requirements become too large even for the cheap storage we now have available. Although referring to database for prosodic coverage, [5] describes exactly the problem in designing databases that cover predefined phenomena. But even if such a database could be designed the more limiting factor is the difficulty in having a speaker *correctly* deliver such coverage.

1.2. The right data

Rather than collecting everything we can try to collect only the "right" data. There are many suggestions of designing database inventory and utterances that cover the desired space. For example [6] use an elaborate scheme where they first model the acoustic space of a speaker, thus finding out which units are acoustically distinct, and frequent enough to deserve coverage. The second stage, more conventionally, greedily selects utterances from a database to best cover that required inventory. The result is a reasonably manageable set of utterances (perhaps 500-1000) which covers the identified acoustic/phonetic space well. Such techniques give better synthesis (per database size) than non-designed databases.

1.3. The right domain

But even this may not be enough. Another direction which is obvious too any one who has built a unit selection based speech synthesis. The quality of output reflects very heavily the style and coverage of the recorded databases. This fact can be explicitly exploited by building specific databases for specific applications. As actual applications often use only a limited number of expressions, or at least a well-defined subset of the language. Databases can be designed to cover that space, and not hit the exponential increase in size that a general coverage database may require. [7] describes how to use such techniques to build reliable high quality synthesizers easily for specific applications.

As no significant spectral or prosodic modification is done to the signal in these basic systems, it is not surprising that even general unit selection synthesizers are still somewhat tied to a domain. That is if the voice is based on a news-reader database it will still sound like a news-reader even used for dialog.

1.4. Synthesizing in style

It is possible to explicitly record different styles of speech and different prosodic contexts. For example we build a databases from a smaller 500 utterance prompt list where every second word was read with emphasis. For example

_Allow me _to interpret _this interesting _silence.
_Tarzan and _Jane raised _their heads.

Then each segment in each emphasized word is marked with an emphasis feature. During synthesis each word desired to have emphasis is constructed from the only those segment with that emphasis feature. This is a crude, but adequate, way to get explicit style. But without methods to modify the selected units, such explicit techniques are required.

1.5. Unit size

Another way to help address the coverage question is to vary the size of the units we are selected. The smaller the units the easier it might be to have coverage over the whole acoustic phonetic space as each units may provide better sharing of contexts. Smaller units such as half-phones are used in [8] or even smaller units based on HMM states as typified by [9] will allow better coverage with a smaller amount of total speech.

Most systems use a fixed size unit, though longer contiguous sections may be selected from the database as a consequence of the selection algorithm. Some system however explicitly allow for mixed sized units. Bonn's HADIFIX system was more explicit in its varying unit length including consonant clusters sized units as well as single phone units [10].

Phonological Structure Matching [11] is explicit in its selection of non-uniform lengthed units. The database is labelled with tree structures. An utterance to be synthesized is also labelled with a tree structure. The database is then searched top down for the largest sub-trees that are contained within the desired utterance. Thus longer units of the database can be selected. There are two advantages here, first selecting longer units will mean less joins which should mean less chance for bad joins. Second, because there are less units being selection this should be computationally more efficient.

In almost all of the current unit selection synthesis systems very little prosodic or spectral modification is done to the selected units. The major consequence of this is that the resulting synthesized utterances can mostly sound very good, and when they do sound good they sound as if the person who recorded the database said the new utterance.

1.6. A finite or infinite number of units

Unit selection systems typically select from a *finite* set of units in the database. They are looking for the best path through a given set of units. Of course when there are no examples of good units in that set, this can viewed either as a lacking in the database coverage or that the desired sentence to be synthesized is not in domain.

Many system do some localised smoothing at boundaries. While [12] introduces the notion of *fusion units*. Thus he effectively increases the number of units available for selection by allowing the construction (fusion) of new units from the existing ones. This direction will gives us a more general solution towards a more general set of units.

A even more general solution is that taken by HTS, [13]. Using a HMM-based framework, in contract to [9] which selects sub-parts of the database, HTS uses the HMM parameter representation to *generate* the speech. Thus effectively a much wider range of units is available, as context affect generation through constraining deltas, and smoother joins are possible. There is a cost though. In its basic form the excitation part of the signal is not modelled thus reducing the quality to vocoded speech, though better excitation modelling is being worked on.

What is important about such directions in unit selection is that the size of the inventory is effectively much larger. However, although it can potentially cover the given space better than conventional unit selection synthesis systems, it its still limited by the examples in the database.

2. SOME OF THE PEOPLE ALL OF THE TIME

In spite of our desire to produce perfect natural sounding synthesis all the time, there are a number of users who do not actually require this. In fact given the current restrictions of general unit selection synthesis, more traditional processes may be adequate or even better.

People who **must** listen to synthetic output a lot, very quickly learn to accept the limitations of voice quality. In fact they often prefer that the voice is less natural but more consistent. As those who work in speech synthesis know, the more you listen to a voice the more acceptable it will become as the human ear tunes to the idiosyncrasies of the particular voice.

In some applications the content being spoken is more important than the style it is delivered in. Listening to lots of data through an audio channel is slow and many people would prefer it to delivered faster than a natural voice could. This brings in the more general issue of delivering information by audio using voice-like methods but they do not need to use only those techniques used in the human voice. [14] has a number of examples which exploit the fact that a synthesizer is being used rather than a natural voice to allow more information to be packed into the channel. For example Emacsspeak can use pitch to denote level of super/sub scripts in formula.

What is important to note here is that for some applications, natural voice is not the most ideal, when we consider the task as information presentation through the somewhat constrained bandwidth of audio other techniques may be better. This is not just voice based aspects but ear-cons, background noises etc. can be used to help information transfer rates.

3. ALL OF THE PEOPLE SOME OF THE TIME

For any particular application of speech synthesis the type of output it will generate is not *everything*. Although we try to build synthesizers which are general enough to be good at everything they are typically not tuned for particular applications.

[7] takes an extreme view of how to get good synthesis all of the time by restricting what the synthesizer can say. Thus a simple talking clock can easily be built that sounds better than a general speech synthesizer, though of course it can *only* tell the time and nothing else.

This is cheating, though is taking advantage of what unit selection synthesis does best. By designing your data explicitly to cover the expected output one can achieve near perfect synthesis for that domain. Often this is sufficient for many applications.

We have built a number of voices specifically designed for applications. Apart from trivial talking clocks, weather information is a useful but constrained domain. Note for easiest construction and best results, developing the generation part of the system in conjunction to the synthesizer itself makes for best results.

For example in [7] we report on a simple weather system for any US city based on live web data giving, time, temperature, outlook, wind direction. A total of 100 utterances were recorded, each of the basic templated form of the intended synthetic utterances. The quality provided is excellent, though of course it can only say the weather.

With the CMU DARPA Communicator system, a telephone based flight information spoken dialog systems [15], a much more general spoken output structure was required. We first analysed what the system had said (using a previous general TTS synthesizer) and built a set of prompts that covered that space. The resulting synthesizer says the in domain text very well as its designed to cover, though sometimes is required to deliver out of domain text, e.g. when a new airport is referred to or some change is made to the language generation systems.

It is clear, through simple blind listening tests, that domain synthesizers can sound much better than general synthesizers. Knowing the desired style and context of the voice allows much more appropriate delivery.

Examples like weather are the extreme cases where the domain can be fully defined and a reasonable set of prompts can be explicitly designed to cover the space. In more general cases there is still a well defined core of expected output. Thus the database can be designed as mixture of domain specific prompts and general prompts.

In fact we have defined this relationship in more detail [16]. One can construct different voices for different tasks (e.g. weather, stocks, email reading) and make explicit changes in voice when changing domains. We called this **tiering**. The second route is combining domain related prompts together into a single voice, typically with a significant amount prompts to support general synthesis. This we call **blending**.

Blending allows, potentially, a smaller footprint and also less firm boundaries between the domains, thus switching between voice types is not required. Though blended voices are harder to get right while small well defined tiered voices are probably the easiest to guarantee high quality all of the time.

However it should be noted that this is only really a solution if the amount of work to design and build a domain

directed synthesizer is sufficiently less than building a general voice.

4. ALL OF THE PEOPLE ALL OF THE TIME

We are now at the stage that we can craft high quality voices that mostly sound human. However its not just the unit selection technology itself that will allow satisfied customers. Even when the voice sounds human, it may still not be appropriate. We are already seeing people comment on voices as being, too direct, not friendly, too friendly, overly polite etc. That is, people are commenting on underlying style rather than naturalness. No matter how natural the voice maybe some people may just not like that voice.

Emphasis, style, voice quality can only currently be controlled with explicitly recording such varied data. Current unit selection techniques typically do not model the speech itself in any sophisticated way, usually because that would introduce degradation in the signal.

If we are to please all the people all of the time we need to be able to control the voice quality and control the style. Which means we need to better model the speech signal, probably using techniques that were developed for early formant type synthesis techniques. This is hard research and may take time before it will reach reliability of current unit selection synthesizers but will give us the flexibility that we require.

5. CONCLUSIONS

Although we have improved the quality of speech synthesis substantially with unit selection techniques, we are far from providing the general, flexible, efficient system that users actually require. We have to be careful in demonstrating synthesis that people understand its limitations, and how good examples do not necessarily translate to continuous good and appropriate synthesis when embedded in an application.

In the short term, domain directed synthesis is clearly better than pre-recorded prompts, and we can already cater for very large domains. But we have to be thinking about the next stage. We must better represent the speech signal to allow for variation, and we must define the controls at a suitable level of abstraction that will allow applications to choose the style and quality they desire.

6. REFERENCES

[1] A. Hunt and A. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *ICASSP-96*, Atlanta, Georgia, 1996, vol. 1, pp. 373–376.

[2] N. Iwahashi, N. Kaiki, and Y. Sagisaka, "speech segment selection for concatenative synthesis based on spectral distortion minimization," *IEICE Transaction Fundamentals*, vol. E76-A, pp. 1942–1948, 1993.

[3] N. Campbell and A. Black, "Prosody and the selection of source units for concatenative synthesis," in *Progress in speech synthesis*, J. van Santen, R. Sproat, J. Olive, and J. Hirschberg, Eds., pp. 279–282. Springer Verlag, 1996.

[4] B. Mobius, "Rare events and closed domains: Two delicate concepts in speech synthesis," in *4rd ESCA Workshop on Speech Synthesis*, Scotland., 2001, http://www.ssw4.org.

[5] van Santen J. and A. Buchsbaum, "Methods for optimal text selection," in *Eurospeech97*, Rhodes, Greece, 1997, vol. 2, pp. 553–556.

[6] A. Black and K. Lenzo, "Optimal data selection for unit selection synthesis," in *4rd ESCA Workshop on Speech Synthesis*, Scotland., 2001.

[7] A. Black and K. Lenzo, "Limited domain synthesis," in *ICSLP2000*, Beijing, China., 2000.

[8] M. Beutnagel, A. Conkie, J. Schroeter, Y. Stylianou, and A. Syrdal, "The AT&T Next-Gen TTS system," in *Joint Meeting of ASA, EAA, and DAGA*, Berlin, Germany, 1999, pp. 18–24.

[9] R. Donovan and P. Woodland, "Improvements in an HMM-based speech synthesiser," in *Eurospeech95*, Madrid, Spain, 1995, vol. 1, pp. 573–576.

[10] T. Portele, W.F. Sendlmeier, and Hess W., "Hadifix: A system for german speech synthesis based on demisyllables, diphones, and suffixes," in *1st ESCA Workshop on Speech Synthesis*, Autran, France, 1990.

[11] P. Taylor and A. Black, "Speech synthesis by phonological structure matching," in *Eurospeech99*, Budapest, Hungary, 1999, vol. 4, pp. 1531–1534.

[12] J. Wouters and M. Macon, "Spectral modification for concatenative speech synthesis," in *Proc. ICASSP*, Istanbul, Turkey., 2000, pp. 941–944.

[13] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *ICASSP*, 2000, vol. 3, pp. 1315–1318.

[14] T.V. Raman, *Auditory User Interfaces : Toward the Speaking Computer*, Kluwer Academic, 1997.

[15] A. Rudnicky, C. Bennett, A. Black, A. Chotimongkol, K. Lenzo, A. Oh, and R. Singh, "Task and domain specific modelling in the Carnegie Mellon Communicator system," in *ICSLP200*, Beijing, China., 2000.

[16] K. Lenzo and A. Black, "Customized synthesis: blending and tiering," in *AVIOS2002*, San Jose, CA., 2002.