LIMITED DOMAIN SYNTHESIS

Alan W Black[1], *Kevin A. Lenzo*[2]

[1]Language Technologies Institute, [2]International Software Research Institute,
Carnegie Mellon University,
{awb,lenzo }@cs.cmu. edu

ABSTRACT

This work presents a reliable and efficient method for building limited domain speech synthesis voices. By constructing databases close to the targeted domain of the speech application, unit selection synthesis techniques can be used to reliably give very high quality synthesis within domain. In addition to a high quality result we include the techniques and processes required to build such voices often allowing new voices in limited but quite complex domains such as dialog systems to be created in under a week. The full tools, documentation examples etc are available for free at `http://festvox.org` .

1. INTRODUCTION

With the recent increase in demand for speech applications, it has become obvious that current general speech synthesis technology is not at a quality that users accept. Many speech applications still use fixed, fully pre-recorded prompts rather than standard TTS (text-to-speech) systems to generate their speech output, because the quality of standard TTS systems is not perceived to be good enough.

Recent improvements in speech synthesis techniques, particularly in the area of so-called "unit selection synthesis," as typified by AT&T's NextGen system [1], have led to higher quality synthesis, but it remains an expert skill to build new voices for such systems. There is a requirement not simply for high quality speech synthesis, but also a reliable and efficient means of creating new, customized voices within the system. It is no longer acceptable for all speech technology systems to speak with one of only a few voices or prosodic styles.

In addressing this issue, we at CMU are making the process of building synthetic voices more reliable and faster, while requiring less arcane skills. Through the FestVox project [2] we release documentation, tools, scripts, etc. that allow new voices to be built in both the existing, supported languages, as well as new languages.

In developing both techniques for general diphone synthesis and unit selection, we noted a particular niche where a limited domain could be exploited to greatly improve reliability of high quality synthesis. In many speech applications, most of the language to be spoken is generated within the system. Despite this, many systems simply pass a raw text string, with no more than perhaps some special punctuation, to a general-purpose TTS system. The result is almost always disappointing, in that it sounds either quite bored (inappropriate prosodic realization) or the signal quality makes it unattractive. In noting that the quality of unit selection synthesis can be very good, and that the number of bad synthesis examples are much less when the sentences are closer to the domain of the recordings, we decided to exploit this by designing corpora specifically for each application.

2. HOW LIMITED IS LIMITED

Many speech applications have their speech output generated by some computed function. Although there are some truly open domains, like reading email, many systems are substantially limited. This may be a simple as slot-and-filler templates, where some known set of names, prices, numbers, etc., and some standard prompts are used. Many IVR systems still use fully recorded prompts to keep quality up, at the price of resource footprint and flexibility. Our initial investigations into limited domain synthesizers were in the form or talking clocks and fixed weather reports, but we have found that we can also deal with more general dialog systems, especially if a backup method is provided for rare out-of-domain cases.

A key aspect of building a limited domain synthesizer is the design of a prompt list that adequately covers the domain. Ideally, we like to have an explicit representation of the utterances that can be generated (e.g. the grammar or templates of the generation system) *plus* information about their frequency of use. From this, a prompt list can be generated to ensure frequent (and most important) forms will be well-represented, while coverage extends to all cases. In a new system the frequency information is not always available but can be estimated. In general, prompts should have at least one occurrence of each word in the vocabulary in each prosodic context.

3. BUILDING A VOICE

The task of building a voice consists of the following processes

– Design the corpus
– Synthesize each utterance
– Record the voice talent
– Annotate (label) the recordings
– Extract pitchmarks
– Extract pitch-synchronous parameters
– Build a cluster unit selection synthesizer
– Test and tune, repeating as necessary

Issues in designing prompts are discussed above. We synthesize the prompts for a number of reasons: first, to ensure that all the appropriate tokens are expanded properly. For example in our Communicator dialog domain, we must ensure flight numbers and dates (both strings of numeric characters) are given the correct expansion. Second, we use the synthesized utterance to estimate the time required for recording. We can, optionally, play the prompt to the human voice talent, but that often has the adverse effect of making the human speak more like the synthesizer, so we generally only present the text. The final reason to synthesize the output is that we use the synthesized prompt in labeling the human spoken utterance.

Although recording with studio quality equipment can give better results, we are interested in making the process as accessible as possible. When studios are used for recording to DAT tape the transfer process and splitting of the files is laborious and time consuming. For all of the limited domain systems we have built, we have recorded directly to computer files. Most commonly we use a laptop (not connected to the mains power) in a quiet room (i.e. without other computers or air-conditioning), to reduce background noise. The recording quality, once audio devices are set up appropriately, is acceptable though taking care at this point is important. More information on the recording process is given in [**?**], including the use of a GUI tool for recording session management (**pointyclicky**).

The prompts are recorded in the desired style for the synthesizer. A talking clock, consisting of 24 simple utterances is one of our standard baseline examples. Building clocks with “funny voices” is easy, but importantly the resulting synthesizer retains the style of the speaker exactly – Scottish accents, falsetto, “laid-back” speakers, and even cartoonish voices are all captured well.

After recording, we label the text using a simple but effective technique based on [6]: we use DTW to align between the mel-scale cepstral coefficients (and delta) of the synthesized and recorded waveforms. As we know the position of the labels in the synthesized prompt, we can map this onto the collected recording. This technique was originally developed for labeling diphone data, where the phonetics are much more clearly defined, but we have found this technique perfectly adequate for this task also. In fact, there are distinct advantages of this often loose labeling over hand crafted low level phonetic labelling. For example when a speaker pronounces the word “Tuesday”, in a Scottish accent, it might better be phonetically labelled as /ch y uw z d ey/, while the synthesizer labels (US English) are given as /t uw z d ey/. But the alignment will match the label /t/ to the spoken /ch y/ and hence when a /t/ followed by /uw/ is selected for synthesis it will select the appropriate piece of speech preserving the original speaker’s idiolect. The speaker must produce utterances that are close to the desired form, but they do not need to be phonetically exact.

Although the labeling is often good, it is never perfect. Hand correction will improve it, with diminishing returns. After labeling, we extract mel-scale cepstral coefficients; we have found that our unit selection techniques work much better if this is done pitch synchronously rather than at a fixed frame rate. As we do not (normally) record these databases with an EGG (electro-glottograph) signal, we extract the pitch marks from the waveform directly, although this is not as accurate as extracting from an EGG signal.

The unit selection technique we use is an updated version of that more fully described in [3]. However, there are a number of substantive improvements in that algorithm since we last published, as well as some specific tuning we have found useful for limited domain synthesis.

The general algorithm takes all units of the same type and calculates an acoustic distance between each, using a weighted Euclidean mahalanobis distance of cepstrum parameters plus F0. Selected features including phonetic and prosodic context are used to build a decision tree that minimizes acoustic distance in each partition. Although [5] makes similar use of decision trees for clustering, we do not use HMMs to first label nor use sub-phonetic acoustic states; nor do we build the tree to its maximum depth, but (optionally) stop with 5 to 10 instances in each cluster.

At synthesis time, we select the appropriate cluster using the decision tree, and then find the best path through the candidates, taking account the costs (and optimal position) of joins using another acoustic based cost (cf. optimal coupling [4]).

For limited domain synthesis, we have determined that certain parameters are more likely to give reliable synthesis. First, in addition to taking candidates from the selected cluster we also include any units that are consecutive in the database to units selected for the previous segment and are of the right type. Thus, selection is not just for candidate units, but we are effectively selecting the beginning of longer units.

Normally for general unit selection we have used phone name as the unit “type name”, though the acoustic distance may also include X% of the previous phone, so these are much closer to diphones than phones. In the limited domain synthesizers, we construct the type from the phone plus the word the phone comes from. Thus a /d/ in the word “limited” is distinct from a /d/ in the word “domain”. This apparently severe restriction may give rise to a claim that we are doing “merely” word concatenation, but this is not true. We are still selecting individual phones, though they come form some instance of the word to be synthesized. In fact, what happens is that a word is often synthesized from phones from different instances of the desired word and the join point between parts is chosen dynamically at the best point, typically in mid-vowel or fricative or silence of a stop.

This choice of unit type means there are now much fewer instances of each type, which has the distinct advantage of much faster synthesis – the initial motivation for this restriction. However we have also found that when words not in the original vocabulary are synthesized they are often poorly synthesized. Therefore, at present, we see this as a good cut-off point at which we can guarantee high quality synthesis. Although this restriction may be disappointing to some, what we are presenting is limited domain synthesis and find this restriction acceptable for many applications; work continues on methods of backing off acceptably.

We now have the selection system working in slightly less time that is takes to do standard diphone synthesis. Although the unit selection process is computationally more expensive than diphone selection, in the unit selection case we do not (usually) do prosodic modification, though we do pitch-synchronous smoothing for some databases. The unit selection database is substantially larger than a diphone database. We have not yet experimented with data compression algorithms, but as the quality of unit selection synthesis depends on larger variety of units available, it will always be the case that all but the smallest limited domain synthesizers require a larger space than diphone synthesizers.

4. A TALKING CLOCK

The original demonstration of this technique was a simple talking clock. The prompts consist of 24 simple utterances of the form

> The time is now, a little after quarter past two in the afternoon.

The basic template of which is

> The time is now, EXACTNESS MINS HOURS DAYPART.

We have successfully built a large number such clocks, some of which are available on-line at `http://festvox.org/ldom/ldom_time.html` .

Not counting recording time, this takes around 3 minutes to build. Such clocks have also been built in languages other than English, such as Chinese and Nepali. On a recent visit to Barcelona, we built a talking clock in Catalan, a language we had no previous synthesis experience in. We designed the prompt list, based on a native informant, and used an existing English synthesizer to cross synthesize the prompts which is adequate enough for automatic labeling. The 12 prompts were recorded and in less than an hour we had a high-quality natural sounding Catalan talking clock.

5. DOES IT REALLY WORK?

Talking clocks are good as toy examples, and for debugging the process but there aren't many applications that require such a closed domain. The question we need to address is how this technique performs on larger domains. As with general unit selection synthesizers, it is clear than when its works the quality is excellent, but what must be more properly investigated is how often this technique fails and how badly. As we are proposing a system that doesn't just offer high quality synthesis, but also a method for building such voices we also must test the reliability of building voices.

We devised a simple weather report system that downloaded weather reports for named US cities from `weather.gov` . This is a simple slot filling template problem with the template of the form

> The weather at, HOUR, on DAY DATE, outlook OUTLOOK, TEMPERATURE degrees, winds WINDIRECTION, WINDSPEED (with gusts to WINDSPEED).

We generated 250 utterances of this type, looping through values for the slots e.g.

> The weather at 1 A.M., on Sunday January 1. outlook cloudy, 20 degrees, winds, North 2 miles per hour.

The first hundred were recorded and used to build a limited domain synthesizer as described above. The second hundred were used to find problems that were then fixed by correcting the automatic labeling. The final 50 utterances were used for testing alone.

Once recorded, it takes less than an hour to build the basic voice on a 500 MHz Pentium III running Linux. Then, less than a day was spent by one person on fixing problems; most of that time was spent doing a visual check over all the phone labels. The second set of one hundred test sentences were used as a diagnostic test. Of the problems found, most were minor segmental labeling errors, though three errors we found where the speaker said a different word from the prompt, "west" for "east" and "pm" for "am" (twice). The autolabeller can (unfortunately) cope with such mismatches but of course this causes a problem when semantically different but phonetically similar utterances are spoken from what is requested. However, as pointed out above this robustness is also sometimes valuable.

The 50 held-out test sentences were then evaluated, both with the fully automatic, but uncorrected labeling, and then the corrected form. Three categories were identified, **correct** where no notable errors in synthesis were heard, **minor** where some notable glitch in synthesis occurs (but the sentence is still fully understandable), and **wrong** where a semantic error occurs (wrong word) or the synthesis has a major problem that affects understandability.

	Correct	Minor	Wrong
Automatic	60%	32%	8%
Corrected	90%	10%	0%

In the corrected case, there were three actual errors (two occurred twice) all of which were easily fixed, and none were particularly serious.

This experiment implies that we do have a relatively robust system for reliably building new voices in a very short time.

6. SCALING UP TO REAL TASKS

A third and more serious limited domain synthesizer we have built using these techniques is for the CMU DARPA Communicator system [7]. The Communicator is a telephone based, mixed initiative dialog system for planning trips, flights, and booking cars and hotels. At first it appears the domain is not closed as it includes greeting to registered users by name, and allows reference to (at least in principle) any airport in the world.

Since the project began some two years ago, we have logs of everything the system has said. To develop our recording corpus, we selected the latest three months of logs and found the most frequent phrases used by the system. Around 100 phrases are what could be term fixed form, in that they contain no variable parts, such as "Welcome to the CMU Communicator," and "I'm sorry, I don't understand that." We then extracted the set of basic templates used by the language generation system and collected the possible values, cities, airports, airlines and the closed classes of dates, times, prices, etc.

For the obvious closed class slots, namely dates, flight numbers, prices, times etc, we constructed a small number of fillers which provided word coverage for each class, without having to list them exhaustively.

For cities and airports, which are essentially an open class, we used the frequency information in our logs to select which set to include in our recordings. For the more frequently mentioned cities we included more than one occurrence in our prompts (in differing prosodic position) and for less frequent names we only included them once, in an intended prosodically neutral position. With around 300 cities and airports we could cover all of cities in the three month logs. On checking through previous logs the percentage of out of domain words was very small.

The templates were filled out with actual values giving rise to around 500 more prompts. These were recorded in the style of a helpful agent, labelled, and a unit selection synthesizer was built. To test the system we used the phrases from our existing logs and listened to many examples. This pointed at errors in labeling which were corrected. The most common form of error was a misplacement of silence (pauses). We had constructed the sentences to use punctuation when a pause is desired, though some of the utterances generated by the language generation system do not always use punctuation consistently. Also, the speaker did not always insert a pause where the synthesizer expected them. These problems are easily hand corrected, and we also used automatic techniques to find pauses which had an unusually large amount of power which tended to be mislabelled sections.

Various text processing issues also were included in this voice to properly deal with flight numbers and homographs such as “US Airways”.

Although we had built an initial test voice for communicator using this technique, as we changed many of the basic prompts and styles for a later version, we rebuilt a new voice once we were confident the system was stable and the code was thoroughly debugged. The final voice was built in under one-man week with a break down of approximately one day to design the prompts, one day to record the prompts and build the basic voice, and the rest of the time for tuning and correction.

After this version was running, we made some changes to the language generation system and decided to add some extra airport names and some more (foreign) city names. We constructed a further 50 utterances and recorded these and added them into the system in another morning’s work. This exercise was important to us, as for many domains although they may be limited they may not remain static so the ability to add new content easily is important.

In an open domain like Communicator we also have to deal with out of vocabulary words. As the unit selection algorithm deliberately fails when an unknown word is present we must provide a backup. We initially intended to only use a diphone synthesizer for the out of vocabulary word alone but it is very obvious when listening to such examples that the voice quality switch midway in a sentence is extremely distracting, especially as the unknown word is typically an important content word like a place name, even though the diphone synthesizer is based on the same voice as our limited domain voice. Thus if a phrase contains an out of vocabulary word we back-off for the *whole* phrase, which although is not ideal, is much more understandable.

We have also considered backing off to a more general a unit selection synthesizer for the unknown word as this would, perhaps, better preserve voice quality. However although the quality of this is sometimes good, it can also be very bad, and have yet no automatic way to distinguishing the quality. It is this wide variation in quality in unit selection that the limited domain synthesis is addressing, hence using a diphones synthesizer currently for us is the best solution.

During recent evaluations of the whole dialog system by external parties, we logged the number of utterances synthesized and also how many contained words out of vocabulary, and hence required the backup diphone synthesizer. Over a three week period 18,276 phrases were synthesized. 459 (2.5%) contained out of vocabulary words (71 distinct words). These were all less frequent (or forgotten) place names.

It is important to note that, although Communicator was not designed as a system that would have a limited output vocabulary, using these limited domain synthesis techniques we have more than adequately given it a more interesting and higher quality voice than a conventional TTS system.

7. CONCLUSIONS

The first important observation to make is that this system does *not* solve the general synthesis problem. We must make that clear as too often a single high quality example is played giving the impression anything can be synthesized at that high quality. However, what we do conclude here is that these techniques allow reliable high-quality synthetic voices to be developed quickly, if they are targeted towards a limited domain.

The advantage that these techniques bring, in that the synthesis implicitly models the quality in the recorded database, is in the long run, a disadvantage too. As more general synthesis is required, with varying prosody, varying emphasis and focus as well as larger vocabularies, the amount of data that needs to be recorded will become too large. At some point we need to properly model prosodic and spectral phenomena explicitly so that we can get the same quality of synthesis without having to record such large databases.

We see this technique as offering a more general solution to system currently using recorded prompts. This offers the quality of recorded prompts but also the generality of simple synthesis so phrases other than those in the recordings can be generated. We do not currently recommend this system for truly general synthesis, such as reading email or news stories, but there still are many speech applications which fall within the scope of this technique.

Full documentation with scripts, code and explicit walkthroughs of these techniques with examples are available at http://festvox.org

8. ACKNOWLEDGEMENTS

This research was sponsored in part by the Space and Naval Warfare Systems Center, San Diego, under Grant No. N66001-99-1-8905. The content of the information in this publication does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

9. REFERENCES

1. Beutnagel, M., Conkie, A., Schroeter, J., Stylianou, Y., and Syrdal, A. The AT&T Next-Gen TTS system. In *Joint Meeting of ASA, EAA, and DAGA* (Berlin, Germany, 1999), pp. 18–24.
2. Black, A., and Lenzo, K. Building voices in the Festival speech synthesis system. http://festvox.org, 2000.
3. Black, A., and Taylor, P. Automatically clustering similar units for unit selection in speech synthesis. In *Eurospeech97* (Rhodes, Greece, 1997), vol. 2, pp. 601–604.
4. Conkie, A., and Isard, S. Optimal coupling of diphones. In *Progress in speech synthesis*, J. van Santen, R. Sproat, J. Olive, and J. Hirschberg, Eds. Springer Verlag, 1996, pp. 293–305.
5. Donovan, R., and Woodland, P. Improvements in an HMM-based speech synthesiser. In *Eurospeech95* (Madrid, Spain, 1995), vol. 1, pp. 573–576.
6. Malfrere, F., and Dutoit, T. High quality speech synthesis for phonetic speech segmentation. In *Eurospeech97* (Rhodes, Greece, 1997), pp. 2631–2634.
7. Rudnicky, A., Bennett, C. Black, A., Chotimongkol, A., Lenzo, K., Oh, A., and Singh, R. Task and domain specific modelling in the carnegie mellon communicator system. In *ICSLP200* (Beijing, China., 2000).