

# The application of interactive speech unit selection in TTS systems

Peter Rutten, Justin Fackrell

Rhetorical Systems Ltd  
4 Crighton's Close, Canongate  
Edinburgh, U.K.

peter.rutten@rhetorical.com

## Abstract

Speech unit selection algorithms have the task to find a single sequence of speech units that optimally fit the target transcription of an utterance that must be synthesized. In doing so, these algorithms ignore a very large number of possible alternative unit sequences that lead to alternative renderings of that utterance. In this paper we set out to explore these alternative unit sequences - by introducing interactive unit selection.

Interactive unit selection is based on feedback of a listener. To collect this feedback we implement two levels of control: an elaborate GUI, and a simple XML tag mechanism. The GUI offers access to unit selection with a granularity of a single speech unit, and allows a user to set prosodic constraints for the selection of alternative speech units. The XML tag mechanism operates on words, and allows the user to request an nth-best alternative selection.

Results show that interactive unit selection succeeds in correcting most of the synthesis problems that occur in our default synthesis system, providing very detailed information that can be used to improve our run-time algorithms. This work not only provides a powerful research tool, it also leads to a number of commercial applications. The GUI can be used efficiently to improve speech synthesis off-line - to the extent that it eliminates the need to make special recordings for domain specific applications. The XML tag, on the other hand, can be used to quickly optimize the output of the system.

## 1. Introduction

Corpus based speech synthesis systems contain a unit selection algorithm that has the task of finding those speech units in the speech corpus that are best suited to synthesize a target utterance. Even for a moderate sized corpus, this means that the synthesizer has to pick one combination from an astronomically large number of possible combinations of speech units. E.g. for a phone based synthesizer, with 100 examples of each phone in the database, the number of alternative renderings of a sentence that is  $N$  phones long is  $10^{2N}$ .

A classical approach to this optimization problem is described in [1], where the optimality of the selection is defined in terms of target- and concatenation cost functions. The choice of elementary speech units, the features that describe these units and the design of cost functions is crucial for the quality of the output of corpus based synthesis systems - and a lot of research effort is spent on this topic (e.g. [2, 3, 4, 5, 6, 7, 8, 9]).

Nevertheless, because of the complexity of the research involved, we may assume that practical systems still mostly rely on ad-hoc definitions of cost functions and their associated features. These cost functions are likely to be sub-optimal, and it

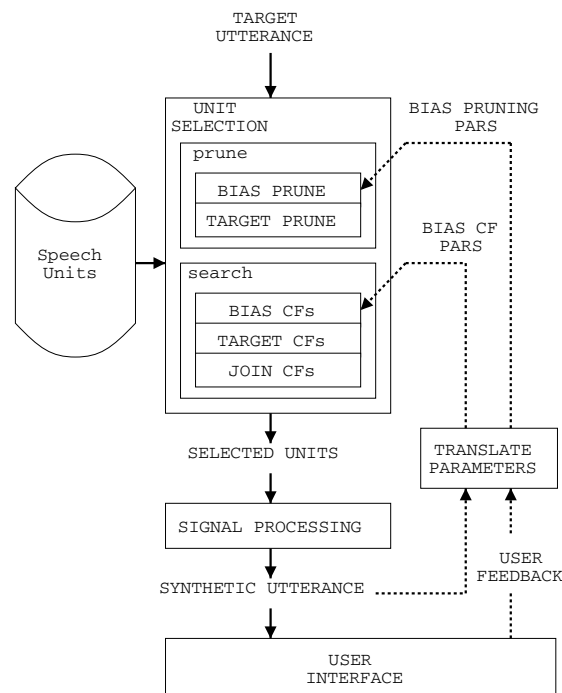


Figure 1: Interactive unit selection.

is unlikely that a unit selection algorithm that uses these cost functions will produce a truly optimal solution. Therefore, we may suspect that poor synthesis could be improved by using one of the many alternative unit sequences that are present in the corpus.

To investigate and exploit the use of alternative unit sequences we introduce interactive unit selection. In a first section we describe the basic mechanisms that are used to channel user feedback into the unit selection algorithm. Following sections describe our evaluation of the technique, how we have integrated interactive unit selection in a commercial TTS system, and how the results can be used for research purposes.

## 2. Interactive unit selection

One way of setting up an interactive environment for unit selection would be to expose parameters of the unit selection algorithm to a listener, and give the listener the ability to manipulate these parameters. In a classical framework, this would mean that a user could manipulate target- and join cost function weights, or even individual cost function shapes. Although this

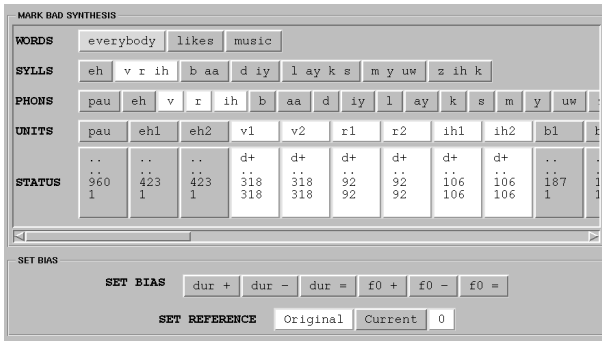


Figure 2: Example of a user interface.

approach would seem straightforward, it is very unlikely that it would be efficient, for the following reasons:

- A high level of expertise is required to interpret the unit selection parameters. This would make the approach unsuitable for naive listeners.
- Any change in unit selection parameters would not only change the problematic parts of an utterance, but also the parts that sound good. We would like to improve synthesis only for the bad sounding stretches of an utterance.
- The standard unit selection algorithm has no way of making use of previously rejected unit sequences to optimize subsequent unit selection attempts.

A more efficient approach to interactive unit selection is depicted in Fig. 1. In this approach we do not expose any unit selection parameters directly to a user. Instead, we provide a user interface (described in section 2.1) that allows a user to describe what is wrong about the synthesized utterance. This information is translated to low level parameters that drive two extensions to the basic unit selection algorithm. A first extension is a bias pruning step (described in section 2.2), that is able to remove rejected variants, or fix accepted variants, for each target speech unit in the utterance. A second extension is formed by a set of bias cost functions (described in section 2.3), that bias unit selection in the direction that is indicated by the user.

## 2.1. User interface

An example user interface is depicted in Fig. 2.

From the listener, we essentially want to know which parts of the utterance sound bad, so we know which sequences of speech units should be replaced. For this purpose, we display the utterance as a sequence of words, syllables and individual speech units. This gives the user the ability to mark problems on the appropriate level. E.g. in Fig. 2, the syllable /v r ih/, belonging to the word "everybody", has been marked. This selection progresses upwards and downwards in the hierarchy of levels, lighting up all elements that will be affected or completely replaced when the utterance is re-synthesized.

When the problematic parts of the utterance are identified, it may be furthermore useful if the user could provide information about what is wrong with those parts of the utterance. We can take this information into account in the search for a better alternative speech unit sequence. In the example interface in Fig. 2, buttons are provided to request duration and pitch

modifications (+, -, =), where the original or current synthetic utterance can be chosen as a reference.

The example interface further contains some status information, displayed below each speech unit. It shows the bias settings that were selected by the listener - for this specific example, a longer duration is requested for the speech units that make up the problematic syllable ("d+"). It also gives some information concerning the number of speech units that are still available at each iteration, which gives an indication of how likely it is that unit selection may still improve.

The collected information will be translated into parameters that control two extensions to the unit selection algorithm: a bias pruning module, and bias cost functions.

## 2.2. Bias pruning

The bias pruning module makes a hard decision about which speech unit variants should be taken into consideration for synthesis. It takes the following parameters for each speech unit in the utterance:

- The identifier of a "frozen" speech unit variant. If this identifier is present, the bias pruning module will only let this particular variant pass to the unit selection search. This makes it possible to only modify the bad parts of an utterance during interactive synthesis.
- A list of identifiers of "bad" speech units. The bias pruning module will not let these units pass to the unit selection search. This makes it possible to reject bad unit sequences.
- Bias reference and bias direction values for speech unit features. This information will be used to remove all variants that are very unlikely to improve the synthesis results, given the user's feedback on e.g. duration or pitch.

The bias pruning module alone is sufficient to support interactive unit selection, because it makes the selective modification of parts of an utterance, through unit selection, possible. Bias cost functions, presented in the next section, can make the procedure more efficient.

## 2.3. Bias cost functions

Bias cost functions are specially designed to channel a listener's feedback into the dynamic search algorithm. They essentially allow the unit selection algorithm to take a bad example as a reference, penalize units that are "worse" than the reference, and reward units that are "better" than the reference. A possible implementation of such a cost function is depicted in Fig. 3. This cost function returns a positive cost for feature values below a reference value, and a negative cost (i.e. lowering the total cost) for feature values that are larger than the reference value.

We assign one bias cost function to each speech unit feature that can be manipulated (directly or indirectly) through the interactive unit selection interface. The bias cost functions are configured for each speech unit in the utterance individually, based on the parameters that are calculated from the listener's feedback.

For the example in Fig. 2 this would mean that we would activate bias cost functions for the duration of the speech units that make up the syllable /v r ih/. The cost function would be of the shape depicted in Fig. 3, causing a bias towards

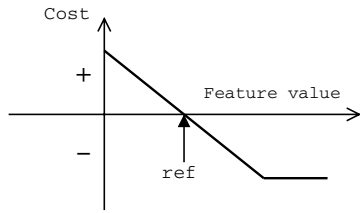


Figure 3: Example of a bias cost function.

longer durations than the reference value - taken from the rejected speech unit sequence.

Bias cost functions are added to the weighted sum of target cost functions. They also have a weight, that should be balanced carefully against the weights of other cost functions - so we achieve the desired biasing effect without distorting the general quality of unit selection.

### 3. Evaluation of interactive unit selection

To evaluate interactive unit selection, we implemented a basic version in our TTS system, and tried to use it to improve synthesis. Informal experiments quickly convinced us that we could correct almost any synthesis quality problem (related to unit selection) we encountered, with a minimal amount of effort.

The results confirm our belief that our standard unit selection algorithm only returns the "tip of the iceberg" of possible good alternatives to synthesize a target sentence, and that the selection it makes is not necessarily the best one. It also showed us that, as far as unit selection is concerned, we have no reason to believe that we should increase the size of our databases (confirming results reported in [10]). We were able to find suitable speech unit sequences for practically any sentence that initially had synthesis problems.

On the other hand, this shows that there must be quite some room for improving the unit selection algorithm. As discussed in section 5, there may be ways to combine interactive unit selection results with automatic machine learning techniques to optimize the performance of the system.

Based on our experience with an initial implementation of interactive unit selection, we decided to integrate it in our TTS system, as described in the following section.

## 4. Application of interactive unit selection in a TTS system

Interactive unit selection has proven to be a very efficient way of manipulating the output of our TTS system, and we will describe two applications of this technique to demonstrate how interactive unit selection can be used in commercial TTS systems.

### 4.1. Prompt sculpting

The prompt sculptor ([11]) is an application that allows the user to efficiently improve speech synthesis off-line, and integrate the results seamlessly into TTS. The prompt sculptor GUI offers access to unit selection with a granularity of a single speech unit, and it allows a user to set prosodic constraints for the se-

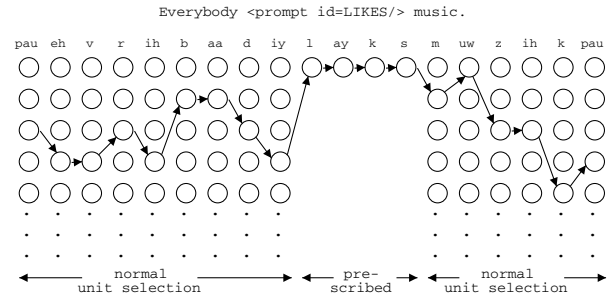


Figure 4: Prescribed unit selection as a means of integrating prompts in TTS.

lection of alternative speech units.

The improvements obtained by interactive unit selection are stored as a sequence of speech unit identifiers that should be used to synthesize a particular text in a particular context. During synthesis, this information is used to pre-scribe unit selection for the concerned parts of the text, as shown in Fig. 4.

In this figure we assume that we have used the prompt sculptor to sculpt the word "likes", and that the prompt for this word is identified by string "LIKES". (This is a somewhat unrealistic example of a prompt, but it serves well to illustrate the mechanism of prompt insertion).

If we trigger the insertion of this prompt into a sentence, e.g. by using an XML tag as in "Everybody <prompt id=LIKES/> music", the synthesizer will look up the pre-scribed units for the word "likes", and only make these units (depicted by circles in the figure) available for the dynamic search.

For the text surrounding the prompt, the synthesizer will retrieve a full list of suitable variants from the database. A standard dynamic search is performed through the resulting lattice of speech units, making sure that the surrounding speech units will match up with the pre-scribed speech units.

Prompt sculpting can be used to build inventories of prompts for application specific, frequently occurring text. It is not suitable to optimize the synthesis of highly variable input text - for this purpose we provide a different way of interacting with unit selection, described in the next section.

### 4.2. Requesting alternative unit selection for words

A second application of interactive unit selection provides a user with the possibility to manipulate unit selection at run time, for specific words in the sentence, while leaving the quality of the surrounding words intact.

In this application, the user interacts with the system by means of an XML tag that specifies which unit selection alternative should be used for a word, or for a sequence of words. E.g. The input "Everybody <sayas alt=1> likes </> music." will tell the synthesizer to use the first alternative unit selection result for the word "likes", while all other words should use the default unit selection result.

We use the bias pruning component of the unit selection algorithm, described in section 2.3, to implement alternative unit selection. Alternatives are created by an iterative unit selection procedure, illustrated in Fig. 5. In each iteration, the bias pruning component will remove the speech unit sequences that were found in the previous iteration steps, and it will "freeze" the sequence of speech units for the words that have reached

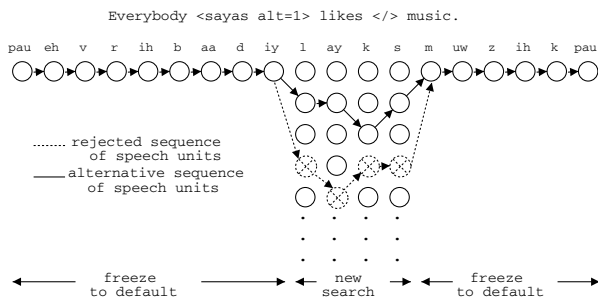


Figure 5: Alternative unit sequence selection.

their desired unit selection alternative (which is the default unit selection result, for words that are not enclosed by the XML tag). As a result, only the words that the user intends to change are changed. When the result is satisfactory, the XML tags that were used in the latest iteration become part of the text that has to be synthesized.

Requesting alternative unit selection for words, by means of XML tag insertion in text, is particularly suited to quickly author the output of the TTS system, for e.g. virtual news readers.

## 5. Future work

Interactive unit selection succeeds in correcting most of the unit-selection related synthesis problems that occur in our default synthesis system, providing very detailed information that can be used to improve our run-time algorithms.

By integrating diagnostic tools in an interactive unit selection application, we can investigate why an alternative sequence of speech units produces a better result than the sequence that was originally selected by our algorithms. Ultimately, we can learn from examples, produced by interactive unit selection, how we should modify our algorithms.

As an example, we could use machine learning techniques, similar to the ones described in [1], [12] and [13] to optimize the performance of our system. In [1] the optimization criterion is formulated in terms of minimizing the distance between synthetic and natural speech, in [12] and [13] cost function weights are modified to improve the correlation between the accumulated cost for synthetic sentences and perceptive evaluation scores. We can make use of the corrections we make by interactive unit selection to formulate another optimization criterion: change the default algorithm in such a way that the likelihood of choosing the corrected version of a speech unit sequence is higher than choosing the rejected version.

Other work involves the optimization of the user interface for interactive unit selection. Currently our interface still requires a basic understanding of phonetic and prosodic terms, and we would like to make it more intuitive to use.

## 6. Conclusions

In this paper we have made the point that the unit selection component of a corpus based speech synthesizer should not be treated as a black box, restricted to producing one single unit sequence for a target sentence. We have proposed an interactive unit selection environment, that can be used to make localized optimizations of unit selection for problematic sentences. The results can be seamlessly integrated in a running TTS system

- making off-line optimization of synthesis possible (prompt sculpting). It can also be used on-line to request alternatives to the default unit selection for specific words in an utterance.

## 7. References

- [1] A. Hunt and A. Black., "Unit selection in a concatenative speech synthesis system using a large speech database", In Proc. ICASSP, 1;373-376, Atlanta, Georgia, 1996.
- [2] M. Balestri, A. Pacchiotti, S. Quazza, P.L. Salza and S. Sandri, "Choose the best to modify the least: a new generation concatenative synthesis system", In Proc. Eurospeech, 5;2291-2294, Budapest, 1999.
- [3] M. Beutnagel, A. Conkie, and A. Syrdal, "Diphone synthesis using unit selection", In Proc. 3th ESCA/COCOSDA workshop on speech synthesis, Jenolan Caves, 1998.
- [4] N. Campbell and A. Black, "Prosody and the selection of Source Units for Concatenative Synthesis", in J. Van Santen et al. (eds.), Progress in Speech Synthesis, pp 279-292, Springer New York, 1996.
- [5] M. Chu, H. Peng and E. Chang, "A concatenative Mandarin TTS system without prosody model and prosody modification", In Proc. 4th ISCA workshop on speech synthesis, 241-246, Perthshire, 2001.
- [6] G. Coorman, J. Fackrell, P. Rutten, and B. Van Coile, "Segment selection in the L&H Realspeak laboratory TTS system", In Proc. ICSLP, 2;395-398, Beijing, 2000.
- [7] P. Rutten, G. Coorman, J. Fackrell and B. Van Coile, "Corpus based speech synthesis in the Lernout & Hauspie Realspeak TTS system", In Proc. IEE symposium on State-of-the-Art in Speech Synthesis, London, 16/1-16/7, 2000.
- [8] Y. Stylianou and A. Syrdal, "Perceptual and objective detection of discontinuities in concatenative speech synthesis", In Proc. ICASSP, Salt Lake City, 2001.
- [9] J. Wouters and M. W. Macon, "A Perceptual Evaluation of Distance Measures for Concatenative Speech Synthesis", In Proc. ICSLP, Sydney, 1998.
- [10] P. Rutten, M. Aylett, J. Fackrell and P. Taylor, "A statistically motivated database pruning technique for unit selection synthesis", In Proc. ICSLP, 1;125-128, Denver, 2002.
- [11] Rhetorical Systems Limited, Patent Application No. 0208813.6, "Method and Apparatus for Sculpting Synthesized Speech".
- [12] M. Lee, D. Lopresti and J. Olive, "A text-to-speech platform for variable length optimal unit searching using perceptual cost functions", In Proc. 4th ISCA workshop on speech synthesis, 75-80, Perthshire, 2001.
- [13] H. Peng, Y. Zhao and M. Chu, "Perpetually optimizing the cost function for unit selection in a TTS system with one single run of MOS evaluation", In Proc. ICSLP, 2616-2616, Denver, 2002.