RECONCILING PRONUNCIATION DIFFERENCES BETWEEN THE FRONT-END AND BACK-END IN THE IBM SPEECH SYNTHESIS SYSTEM

Wael Hamza, Raimo Bakis, and Ellen Eide
IBM T.J. Watson Research Center,
Yorktown Heights, NY 10598, USA
{ hamzaw,bakis,eeide}@us.ibm.com

ABSTRACT

In this paper, methods for reconciling pronunciation differences between a rule-based front-end and the pronunciations observed in a database of recorded speech are presented. The methods are applied to the IBM Expressive Speech Synthesis System [1] for both unrestricted and limited-domain text-to-speech synthesis. One method is based on constructing a multiple pronunciation lattice for the given sentence and scoring it using word and phoneme n-gram statistics computed from the target speaker’s database. A second method consists of storing observed pronunciations and introducing them as alternates in the search. We compare the strengths and weaknesses of these two methods. Results show that improvements are achieved in both limited and unrestricted domains, with the largest gains coming in the limited-domain case.

1. INTRODUCTION

Concatenative speech synthesis systems that use large speech databases have proven successful in generating high quality speech output [2],[3],[4] especially when they are used to synthesize limited-domain text [5],[6]. Typically, the synthesis process comprises three major steps: front-end processing, prosody generation, and back-end processing. The front-end takes input in the form of raw text which contains numbers, dates, times, currency, abbreviations, and punctuations and converts it into a normalized format. It also performs some annotation such as word emphasis and prosodic phrasing. Then pronunciations are produced for this normalized annotated text. The prosody generation module performs prosodic information prediction such as pitch contour and segmental duration of the given sentence. All this information is passed to the back-end which in turn generates the corresponding speech signal. In the case of concatenative synthesis that uses a large speech database, this process is carried out through the concatenation of pre-recorded speech units selected from a large number of candidate speech segments by minimizing a cost function.

The issue of using rule-based vs. corpus-based approaches to perform the above processes, especially the front-end process, is still an open issue in the field of speech synthesis research [7]. However, in the IBM Expressive Speech Synthesis System, a significant improvement was realized by moving towards corpus-based prosody prediction techniques trained on data from the target speaker since, although the rule-based system is carefully designed and implemented, it does not take into consideration the specific prosodic patterns the target speaker might have. On the other hand, rule-based approaches might be superior in the case of insufficient training data.

A rule-based system produces pronunciations in the IBM Expressive Speech Synthesis System. Although the rule-based system rarely produces unacceptable pronunciations, the pronunciations produced are often different from the pronunciations the target speaker uses. For example, the front-end may produce “EE DH ER” as the pronunciation for “either,” while the speaker may have pronounced the word as “AY DH ER.” As described later in detail, this difference results in a lack of contiguous candidate speech segments to synthesize a given phonetic sequence.

In this paper, we present two methods of adapting the rule-based pronunciations produced by the front-end to match the target speaker’s pronunciations; we discuss the relative merits of each. The paper is organized as follows. First, in Section 2, an overview of the IBM Expressive Speech Synthesis system is presented. In Section 3, the problem under consideration will be described in detail; the proposed solutions and their results are presented in Section 4. Finally, we draw some conclusions in Section 5.

2. THE IBM EXPRESSIVE SPEECH SYNTHESIS SYSTEM – OVERVIEW

We recently introduced the IBM Expressive Speech Synthesis System. Details of the system may be found in other publications [1],[2],[8] and are summarized here. The system is concatenative, with context-dependent subphonemic synthesis units. It is capable of, in addition to the default, neutral speaking style, conveying good news, conveying bad news, asking a question, and showing contrastive emphasis appropriately.

During synthesis, the IBM system uses a rule-based front-end to convert the input text to a sequence of target phonemes. An acoustic-context decision tree is used to convert the target phonemes into target subphonemic units we will refer to as leaves. The tree improves the speed of the system and also forces contextual information to influence the choice of segments in the search by segregating all segments representing a given phoneme into context-dependent clusters. The tree has an undesirable property, however, that segments representing a given word may end up at leaves other than the leaf asked for by the search when synthesizing that same word, due to differences in the phonetic context of that word in the training and test, or due to differences in the expected and observed pronunciation of the word.
In selecting segments from which to form synthetic speech, only segments which map to the appropriate leaf of the acoustic decision tree are considered. Thus, even if a word appears in the database, segments from it are not necessarily considered when synthesizing that word, because of contextual differences. By allowing segments from leaves other than the one requested by the search, as described in the remainder of this paper, we are able to relax the context-constraint and trade it off with obtaining fewer splices and possibly retrieving segments drawn from the word to be synthesized.

3. THE PRONUNCIATION MISMATCH PROBLEM

Although the rule-based pronunciations produced by our front-end are usually acceptable, mismatches between those pronunciations and the pronunciations in the target speaker’s database often occur. This mismatch is most costly when the text to be synthesized is similar to that in the training data, because in that case, if there were no mismatch, long contiguous chunks would likely be used to construct the synthetic speech. Such contiguous chunks of speech would lend themselves well to bypassing all signal processing, thus retaining the naturalness of the original recording [1]. To illustrate this issue, consider synthesizing a complete sentence from the training data. Assuming the prosodic targets match the observed prosody, we would expect to retrieve the entire sentence from the training corpus. Unfortunately, the rule-based system often produces pronunciations different from the target speaker’s database pronunciations, which forces the system to select segments from different database sentences, introducing splices in the synthesized speech and potentially reducing its quality. In limited-domain synthesis the training sentences closely resemble the sentences to be synthesized, so that sequences of words to be synthesized are likely to be found in the corpus. Mismatches in front-end and back-end pronunciations are especially costly here, because the long chunks of contiguous speech potentially obtainable are nonetheless unutilized. One solution to this problem in limited-domains is implemented in the IBM Phrase Splicing System [5], in which a dictionary of phrases in the database, along with the pronunciations used by the target speaker are stored. During limited-domain text-to-speech synthesis, this dictionary is searched for the required phrases and pronunciations are produced by copying pronunciations from the dictionary. Although this is an effective solution, it works only on the exact phrases identified by the application developer.

A similar mismatch problem arises when the front-end system phrases a given sentence in a way which leads to insertion of silences in different places from those in which the speaker actually paused. In this case, the contexts of the phonemes near the silences depend on their presence or absence, and therefore this also results in more splices in the output speech, potentially reducing its quality.

In the next section, two methods of compensating for the mismatch between the front-end and the back-end are described.

4. COMPENSATING FOR THE PRONUNCIATION MISMATCH PROBLEM

In this section we introduce two methods of compensating for the mismatch between front-end and back-end pronunciations. The first, presented in Section 4.1, relies on word n-grams to predict silence positions, and relies on phone n-grams to select from possible alternate pronunciations. Once the silence and pronunciations are chosen, the altered output of the front-end is passed to the back-end. This approach has the advantage that it could generalize to new words, but has the disadvantage that a hard decision is made before the search.

A second approach, discussed in Section 4.2, compiles a list at build time of all pronunciations of each word observed in the training corpus. During synthesis, that list is used to augment the segments considered for each unit to be synthesized. This approach has the advantage that it makes soft decisions and allows the search to select the lowest-cost solution. However, it has the disadvantage that it cannot change pause positions, and it does not generalize to unseen words.

4.1 Pronunciation reconciliation using n-grams

One way to solve the problems discussed in Section 3 is to correct the output from the front-end module to closely match the speaker’s database. In this subsection we describe a method for automatically making those corrections.

At build time, we compute n-gram statistics at both the word and phoneme level. A special word “SIL” represents silence. This word is introduced in the training corpus script in places where the speaker originally paused. The word sequence, including silence insertion as well as the phonetic sequence can be easily extracted at build time. This example of a training sentence with pauses identified is illustrated in Figure 1.

4.2 Pronunciation reconciliation using an alternate pronunciation

In this subsection we describe another method for compensating for the pronunciation mismatch between the front-end and back-end. In this approach, a pronunciation dictionary is compiled at build time that includes all possible pronunciations for each word in the training corpus. When constructing synthetic speech, the system will then choose the pronunciation that best matches the expected output of the front-end. This approach is especially useful in limited-domain synthesis where the training sentences can be expected to closely resemble the sentences to be synthesized.

1 Forcing the speaker to produce pronunciations similar to the rule-based produced ones is impractical and negatively affects the speaker’s comfort during recording.
During synthesis, the system uses the required word sequence to construct a word lattice as shown in Figure 2, removing the pause locations suggested by the front-end, and inserting an optional silence between each two consecutive words. The resulting lattice is rescored using the word n-gram language model obtained in the build step. Silence locations suggested by the rule-based system are optionally given less cost than other silences to promote the rule-based phrasing. The best sequence of words and silences is selected and passed to a pronunciation lattice generation step.

Using the sequence of words with inserted silences, a pronunciation lattice is constructed using pronunciations produced from the rule-based system. The lattice is expanded to include multiple pronunciations for the given words, where additional pronunciations are obtained from a pronunciation dictionary which has been optionally tailored to match the pronunciations used by the database speaker. The resulting lattice is rescored using the phonetic sequence n-gram statistics. Optionally, the pronunciation paths suggested by the front-end can be given less cost to bias the system toward using those pronunciations. The best phonetic sequence path is selected. This process is illustrated in Figure 3. The resulting phonetic sequence is passed to the back-end processor for segment selection. This method is well suited for implementation in a Finite State Transducer synthesis framework [9].

In order to assess the n-gram based method outlined in this subsection, a male voice was built using the process described in Section 2. In addition to the regular database sentences, financial news domain sentences were added to the build to serve as domain-specific data in the database. Two sets of test sentences were prepared. The first set contained news sentences representing general unrestricted sentences while the second set contained financial news sentences serving as limited-domain sentences. Both sets are synthesized using the regular synthesizer with and without the proposed method.

We measured splice percentages on a development set to determine the appropriate length of the n-gram. We found 5 to be the best n-gram length for phonemes, while in the case silence insertion, we found that using an n-gram of 2 was optimal.

For unrestricted synthesis, only a 3% improvement in the number of splices in the output speech is realized. However, on the domain-specific data, a 31% reduction in splice counts was achieved by using this technique.

One strength of the above approach is that, since decisions are made outside the segment search which requires a known-length sequence of leaves in our system, sequences of varying lengths may be considered. Additionally, this method generalizes to words not originally recorded. However, it makes a hard decision about pronunciations before the segment search. In the case where the number of phonemes in the alternate pronunciations are the same as the number of phonemes predicted by the front-end, we can directly include all possible alternates in the search, rather than making a hard decision. That approach is explored in section 4.2.

### 4.2 Pronunciation reconciliation using soft decisions

A second method of reconciling the mismatch between pronunciations generated in the front-end and those encountered by the back-end is described in this section.

At the heart of the method is a dictionary of alternate pronunciations at the segment occurrence level. To construct the dictionary, for each word in the training database, an entry is created which specifies the word, the segment occurrences which
In synthesis, for each word to be synthesized, the word is looked up in the dictionary and the list of segment occurrences corresponding to that word are retrieved. For each leaf to be synthesized, the list of candidate occurrences over which segment search is performed is augmented with the list of occurrences coming from the alternates dictionary, which is ordered according to contextual similarity with the context to be synthesized. First in the list are occurrences which match both left and right word contexts. Next are the occurrences which match both left and right phone contexts. Next are the occurrences which match either the left or the right word context but not both, and finally are the occurrences which match either the left or the right phone context, but not both. The list of alternates considered is truncated at a user-specified number.

Alternate segments which survive the truncation are considered as additional candidates in the segment search.

Using this method of reconciling the pronunciations generated by our rule-based front end and the pronunciations uttered by our speaker stored in the back-end, we were able to reduce our splice count by 16% on general speech, and by 34% on in-domain sentences. As this method outperformed the method described in Section 4.1 in terms of splice reduction, we ran a listening test using the method of pronunciation reconciliation described in this section to validate its worth. The test material was presented to a set of native North American English speakers, 12 male and 12 female. Stimuli consisted of 30 “neutral” sentences generated from each of 3 sources: our previous system, which did not allow alternate pronunciations, our new system which did contain alternate pronunciations, and natural speech from the speaker who spoke our database. We divided the test sentences into two categories: in-domain and out-of-domain. In-domain sentences contain things such as weather reports and travel information, where similar (although not identical) sentences were found in our original script, while out-of-domain sentences were general and bore less resemblance to the utterances in the database. All listeners heard all stimuli, with order randomized, and were asked to rate the overall quality of the speech they heard on a 7 point scale. We measured our performance on the in-domain and out-of-domain sentences separately. While we made good progress on out-of-domain sentences (closing 9.7% of the gap between our system and natural speech), we made even more striking improvements on sentences which were similar in content to the sentences in our database (closing the gap between our system and natural speech by 17.1%).

5. CONCLUSION AND FUTURE WORK

In this paper, we presented two methods of adapting pronunciations produced by the front-end of our TTS system to match the pronunciations of the target speaker. The first method rescoring a multiple pronunciation lattice using n-gram statistics trained from the target speaker pronunciations. Results show an insignificant improvement in unrestricted synthesis but a significant improvement in the limited-domain synthesis. The second method consisted of compiling at build time a dictionary of the segments corresponding to examples of each word in the training script. During synthesis, those segments are introduced as additional candidates in the segment search.

We found the second method to outperform the first method. Thus, the ability to change pause locations and variable-length sequences as well as to generalize to unseen words introduced by the first method was outweighed by the chance to overcome errors in the hard decisions by the phonetizer and acoustic tree, which is the strength of the second method.

In the future we plan to use the word n-gram approach of repositioning silences in conjunction with the soft-decision approach of the second method to further reduce our splice counts and improve the quality of our synthetic speech.

6. ACKNOWLEDGEMENT

The authors would like to thank Stanley Chen of IBM Research for his help using IBM Finite State Machine toolkit used in this work for lattice construction, n-gram statistics computations, and lattice rescoring.

7. REFERENCES