A CORPUS-BASED APPROACH TO <AHEM/> EXPRESSIVE SPEECH SYNTHESIS

E. Eide, A. Aaron, R. Bakis, W. Hamza, M. Picheny, and J. Pitrelli
{eeide,asaaron,bakis,hamzaw,picheny,pitrelli}@us.ibm.com

IBM T.J. Watson Research Center
Yorktown Heights, NY 10598 USA

ABSTRACT

Human speech communication can be thought of as comprising two channels – the words themselves, and the style in which they are spoken. Each of these channels carries information. Today's most-advanced text-to-speech (TTS) systems such as [1],[2],[3],[4] fall far short of human speech because they offer only a single, fixed style of delivery, independent of the message. In this paper, we describe the IBM Expressive TTS Engine, which is able to add another channel by offering five speaking styles. These are: neutral declarative, conveying good news, conveying bad news, asking a question, and showing contrastive emphasis. In addition to generating speech in these five styles, our TTS system is also able to generate paralinguistic events such as sighs, breaths, and filled pauses which further enrich the style channel. We describe our methods for generating and evaluating expressive synthetic speech and paralinguistic effects. We show significant perceptual differences between expressive and neutral synthetic speech for each of our speaking styles. In addition, we describe how users have been empowered to easily communicate the desired expression to the TTS engine through our extensions [5] of the Speech Synthesis Markup Language (SSML) [6].

1. INTRODUCTION

Today's most-advanced text-to-speech (TTS) systems use a cost function to select segments from a recorded database; the selected segments are then concatenated to form the synthetic speech signal. In such systems, the database of speech segments is typically collected by recording a speaker reading a large script. The underlying speaking style of the speaker from which the database is recorded is reliably perceived in the output synthetic speech [7]. Usually, when building these systems, researchers strive for consistency, because pieces of waveforms from one part of the database are expected to join seamlessly with pieces of waveforms from other parts of the database. In addition to consistency in low-level parameters, such as loudness and distance from the microphone, the speaker is asked to maintain consistency in speaking style. Because a single style of speaking is maintained during the recordings, and that style will be largely preserved in the output speech, the target recording style is necessarily chosen to be a compromise which is expected to be appropriate in most applications. A typical example is a style which is somewhat neutral but with warm, friendly tendencies, such as that of a weather reporter. Unfortunately, any single style, including that of a friendly newscaster, cannot be appropriate in every context. In the case of conveying bad news, such as: *"That flight is sold out,"* a subdued approach would be much more appropriate.

A desirable quality of a TTS system is the ability to speak *expressively*, dynamically adapting the style of voice according to the message. Such a system would, for example, speak in an upbeat manner when conveying good news and in a more subdued tone when delivering bad news. Other examples of expressions that a TTS

Expression	Example
Good news	I have successfully reset your PIN.
Bad news	I am unable to verify your identity.
Confusion	I did not understand your request.
Contrast	This is a *round-trip* fare.
Apology	I cannot find you in my records.
Question	Do you confirm the sale of all shares?
Confidence	Your account balance is $8,432.50.
Greeting	Welcome to the IBM Help Desk.
Farewell	Thanks for calling. Goodbye.

system might need include confusion, showing contrastive emphasis, apology, asking a question, exuding confidence, greeting, farewell, etc. Specific examples of each of these expressions are shown in the table above. Note that some expressions are overlapping, such as bad news and apology. We may be able to use one system to convey several expressions, as long as the output does not sound inappropriate for the message; a key feature of an expressive TTS system is its ability to communicate in a manner that is appropriate for the message being conveyed. Generally, an authoritative-sounding voice may be best for an information-provision application such as a travel-planning system, but in a case in which the system does not understand the customer's request or cannot comply with it, a less authoritative expression would be more appropriate, to complement rather than conflict with the words expressing either confusion about the request or remorse about the system's inability to comply with the request. In such a case, the appropriateness of the expression is likely to contribute to the customer's satisfaction with the quality of the system. Efficiency may be an additional benefit here, as it may impose less cognitive load on the listener if the expression coordinates with the text of the message rather than detracts from it.

Many automated telephone systems support conversations between callers and computers. The caller speaks to a speech recognition system, and the computer responds using text-to-speech software. One goal of the TTS designer is to maximize the communicating efficiency of the system -- both the quantity and accuracy of the information it can carry in a given time. Until recently, that mostly involved making the output speech more intelligible. At this point the intelligibility problem has largely been solved, so designers now seek other ways to increase the quantity and quality of information that TTS can communicate. As an example in which an expressive voice is more efficient than a non-expressive voice, consider the following interaction from a travel-planning system:

Customer: I'd like a flight from New York to Denver tomorrow morning.
System: I have a flight from New York to Denver tomorrow evening.

Without an ability to express the contrast between the requested morning flight and the provided evening flight, the customer is likely to attribute erroneously the mismatch to the system misunderstanding his request, and follow with a repeat request. If, instead, the system has an ability to specify to the synthesis component that “evening” be spoken in such a way as to express contrast, the customer is more likely to understand that the system has understood him and is providing the best alternative, thereby sparing the diversion and avoiding a lengthier interaction.

In this paper, we discuss enabling a concatenative text-to-speech system to speak expressively. We focus on a corpus-based approach. Concurrently we are exploring another approach to generating expressive synthetic speech, which relies on an intermediate prosodic phonological representation such as ToBI [8],[9]. That approach is described in detail in [10] for the generation of questions and contrastive emphasis. We are currently working on merging these two complementary approaches, to arrive at a unified expressive TTS engine. In Section 2 of this paper, we describe the corpora we use in our expressive speech system. In Section 3 we describe how we build statistical models of pitch and duration from the expression-specific data. In Section 4 we describe how we make use of the expressive data and expression-specific prosody models to generate expressive speech. In Section 5 we describe the addition of paralinguistic events such as sighs and filled pauses to the synthetic utterance. In Section 6 we describe our extensions to SSML which enable users to communicate desired expressions and paralinguistic events to the synthesis engine. Finally, in Section 7 we present results on the ability of our TTS system to speak expressively, and we discuss those results and future work in Section 8.

2. DATA COLLECTION

A viable method of generating expressive speech is to collect separate speech corpora in each desired expression, and synthesize using a system trained exclusively from the data of a given expression in order to produce speech in that style [7]. However, that approach is expensive, slow, and inflexible. In this section we discuss a variation of that approach which is much more practical to deploy. Specifically, we collect a set of databases, one in each of the desired expressions. The expression-specific texts were read by the same professional speaker who generated our neutral database. However, for each expression we need collect only enough data to train prosody models; we replace the prosody models built from the neutral database with those trained on data from a given expression.

The speaker read a small number of sentences for each desired expression. Each set of sentences was written so that the message of the text was consistent with the target expression. We collected data for the following expressions: conveying good news, conveying bad news, and asking a question. From the data for a given

expression, we build expression-dependent prosody models as described in the next section.

3. EXPRESSIVE PROSODY MODELS

To build a prosody model for each expressive state, an end pitch and a delta pitch for each syllable are predicted from a set of features gathered from the text. This method, including the features, is the same as for our neutral model; only the data from which the model is built differ.
Features include:

- Lexical stress of the current syllable
- Phrase-level stress of the current word, as predicted by the rule-based front-end processor
- Distance of the current word from the beginning of the current phrase
- Distance of the current word from the end of the current phrase
- Part of speech of the current word
- Type of the current phrase (yes/no question, ends-in-comma, ends-in-period, etc.)
- Pitch at the end of the current syllable as predicted by the rule-based front-end processor

In training the model to predict F0, for each syllable in a given training utterance, the feature vector associated with that syllable along with the feature vectors associated with the two syllables to the left and to the right are concatenated, and associated with an observation vector consisting of log(p) and Δp, where p is the pitch in Hertz at the end of the syllable nucleus. From these feature vectors and observations, a decision tree is built to maximize the likelihood of the observations.

During synthesis, the same features are assembled and dropped down the tree for each syllable in the sentence to be synthesized. The mean pitch and mean delta pitch at the resulting tree leaf are used to construct the target pitch contour. The estimated end pitch and delta pitch of the syllable are used to calculate a target pitch contour which, after smoothing [1], is used to evaluate the pitch target component of the cost function for each database segment under consideration for selection.

Analogously to the F0 contour generation, we use a decision tree model to produce a duration target for each phone to be synthesized. A set of features is derived from the text for each phone, including:

- The phone identity, as well as that of the two phones to the left and to the right of the current phone
- Voicing (voiced/unvoiced) of the current phone and of the two phones to the left and to the right of the current phone
- Broad class of the current phone (vowel, semi-vowel, fricative, nasal, plosive) as well as of the two phones to the left and to the right of the current phone
- Total number of syllables in the current word
- The number of syllables preceding the current phone's syllable in this word
- The number of syllables after the current phone's syllable in this word
- Lexical stress of the current syllable
- Phrase-level stress of the current word, as predicted by the rule-based front-end
- Number of words between the current word and the beginning of the phrase
- Number of words between the current word and the end of the phrase
- Part of speech of the current word
- Type of the current phrase

These features are then paired with the observation log(d), where d is the duration of the current phone. From the feature vector and observation pairs, a decision tree is constructed to maximize the likelihood of the observations assuming a Gaussian distribution at each node of the tree.

In synthesis, feature vectors are determined from the text to be synthesized in the same manner as was used for training the decision tree. The feature vectors are then dropped down the tree; the mean of the duration of all training vectors mapping to that leaf is then used as the target duration for the phone to be synthesized.

4. SYNTHESIZING EXPRESSIVE SPEECH

In addition to building prosody models from each expression, we include the small set of segments from each of the expressions in the search, motivated by the fact that prosody alone does not fully convey the desired expression, as shown in [11]. In order to include the expressive data in the search and bias the search towards choosing segments from the appropriate expression, we first tag each occurrence of each unit in the expressive databases with the underlying expression with which that database was collected. The expression-tagged units are then pooled with the neutral data, which have been tagged as "neutral."

In addition to tagging the segments with the expression from which they came, we currently construct an expression-cost matrix, which specifies the cost of choosing a segment from expression i in synthesizing expression j, for all pairs of expressions i and j. We construct this matrix by hand using trial-and-error to tune the costs. With the prosody models trained as described in

Section 3, the databases tagged, and the cost matrix in place, we are ready to generate expressive speech.

In synthesis, the desired expression's prosody models are used. All segments from all expressions are considered, with the penalties for using a segment from expression i to synthesize expression j comprising an additional component in the cost function compared to the single-expression synthesis system. Should we increase the size of the expressive databases, we would expect the cost of substituting one expression for another would need to be increased. However, in the current embodiment, the expressive databases are small, and the quality of the synthesis is improved by allowing neutral segments, as well as segments from other expressions, in the search. We trade-off the degree to which the desired expression is conveyed by the spectral qualities of the segments chosen to comprise the synthetic utterance against the smoothness and overall quality of the synthesis.

The above discussion assumes that the desired expression is known; determination of which expression to use when is beyond the scope of this study. In our system, we have extended SSML to include our set of target expressions; here we assume the user provides the system with marked-up text which specifies the desired expression. This markup will be described more fully in Section 6.

5. PARALINGUISTIC EVENTS

When humans converse, in addition to speech they produce non-speech sounds which can be cues that impart additional information beyond that which is carried by the words alone. Breaths, coughs, sighs, chuckles, and hums all modify the message being conveyed and subtly add information. For example, a sigh is a sign of distress or unhappiness, whereas a chuckle indicates light-heartedness. In a TTS system, such paralinguistic events efficiently provide cues as to the state of a transaction, such as a sigh succinctly signaling that no flight at the requested price could be found, or that a compromise may be necessary. Such paralinguistic events also make the synthetic speech sound more natural.

In order to augment the synthetic speech signal with these paralinguistic events, we first listed all of the events we were interested in being able to generate. We then composed a script which would be easy for the speaker to understand, such as "*mmm*, that bread smells wonderful," and, "Excuse me, *throat clear*, I have something to say." We recorded each event in its carrier sentence and then excised these events by hand. The excised events form the portfolio of events we are able to generate.

In synthesis, we insert these events, when desired, into the synthetic speech stream. Having multiple tokens of many of these events enables us to choose randomly among several examples. Doing so improves the naturalness of the speech when more than one occurrence of a given event is required, because repeating the same recording of a phenomenon such as a breath several times is easily recognized by a listener as being artificial.

A user of the TTS system is able to include a paralinguistic event in the audio stream by marking up the text through using an extension of SSML, as explained in the next section.

6. EXPRESSIVE MARKUP

In our TTS system, we rely on markup as the means for a user to specify the desired expression for each utterance. The markup is interpreted by the TTS engine and dictates the choice of prosody models, as well as the cost of each segment in the search, as was described in Section 4.

In order to facilitate this interface between the user of the TTS system and the expressive TTS engine, we extended SSML according to the guidelines we proposed in [5].

The use of markup as an interface between the user and the engine enables our expressive TTS engine to be easily integrated into a dialog system. In that case, knowledge of the internal natural-language-generation state implies an appropriate expression, whose specification is then passed to the TTS system by augmenting the automatically-generated text prompt with markup specifying an appropriate expression for the text.

Independent of the approach taken by a synthesizer to generate expressive speech, conveying the desired expression to the synthesizer is necessary for efficient and appropriate speech-based interactions between human and machine. In a unit-selection-based synthesizer, one could collect databases spoken in different expressive states in order to generate synthetic speech with an expressive content, as in this paper, or one could use signal processing to adapt neutral speech to achieve a desired expression. Either way, though, the desired expression needs to be specified to the synthesis system through markup. Ideally, that markup should be hierarchical, so that users with different areas of expertise can interface with the engine at different levels of abstraction, in the manner which is most natural and convenient for them. For example, film directors can specify emotions, while

speech scientists can interface with the system by specifying pitch and duration contours.
As an example of the benefits of the hierarchical view of the extended SSML language, consider the case of contrastive emphasis. In our prosodic phonological approach to expressive speech described in [10], we use our hierarchical extensions of SSML to specify emphasis. The emphasis tag is translated into a series of ToBI symbols which typically correspond to emphasized speech. Those symbols, which sit in a lower level of the hierarchy than does "emphasis," are then passed to the TTS engine.

The multilayered framework for specifying expressiveness creates a rich, annotated text to be used by the synthesizer. The expressive speech synthesizer deals with tags or tag layers using one of the following three alternatives:

1. Use the tags directly in the speech synthesis process.
2. Translate tags from one layer to tags in a lower-level layer using tag translators, which are the set of rules or systems that map tags in one layer to corresponding tags in another.
3. Ignore tags not supported.

Option 1 allows the synthesizer to use various layers of tags to provide better quality speech output. In the case of a concatenative synthesizer having a set of expressive databases available, say, one for each of a set of desired expressions to be synthesizable, the high-level tag is passed directly to the synthesizer.

Option 2 enables the design of new tag-layers along with their interfaces and using them with legacy synthesizers by developing appropriate translators to translate new tags to tags belonging to layers understood by the synthesizer. In the case where expressive speech is achieved via signal processing on a neutral database, abstract specifications would be transformed into physical ones. Option 3 allows extension to the repertoire of styles possible by a TTS engine while preserving backward compatibility with a legacy synthesizer.

In order to reinforce the expression being conveyed by a given text, a developer or client application may desire a particular paralinguistic event to occur at a particular point in the audio stream. This ability is enabled through the use of markup, in a manner very similar to the specification of the desired expression itself. For example, a developer could specify:

<prosody style="bad news"> Well <sigh/> the cheapest flight is more than your allowed maximum. </prosody>

which would indicate to the TTS engine that a sigh appropriate in a bad news context should be placed between the words "well" and "the" in the audio. Markup specifying these events is a convenient way for a developer to achieve these types of events in the audio coming from the TTS engine.

7. RESULTS

In order to verify that the method of generating expressive speech described in this section was effective, we performed a separate listening test for each expression. Each test was administered to 32 native English speakers, 16 male and 16 female. Each listener heard 30 pairs of sentences, where one member of the pair was synthesized from our single-expression system, and the second member of the pair was the same text synthesized from the expressive TTS engine. The order of the systems heard by the listener was randomized, so that half of the time the listener heard the default system followed by the expressive system, and half of the time the listener heard the expressive system followed by the default system.

In order to test the ability of the expressive TTS engine to speak good news, we composed a test set of 30 utterances which were conveying good news, such as "Congratulations, you have the winning ticket." Each of these sentences was synthesized by the default and by the good-news system. Listeners were asked to specify which member of the pair of stimuli for each sentence sounded more like good news.

Similarly, in order to test the ability of the expressive TTS engine to speak bad news, we composed a test set of 30 utterances which were conveying bad news, such as "I'm sorry, I cannot locate your order." Each of these sentences was synthesized by the default and by the bad-news system. Listeners were asked to specify which member of the pair of stimuli for each sentence sounded more like bad news.

Finally, to test the ability of the TTS system to generate questions, we composed a set of 30 yes/no questions, such as "Do we have time to go to the park?" Listeners heard each of these questions as synthesized by the default and expressive TTS engines, and were asked which of the two stimuli for each question sounded more like a question.

Results for each of the three expressive states tested are shown in the table below. All of the results are statistically significantly better than the chance result of 50%.

Expression	Percent Correct
Bad news	70.2
Good news	80.3
Yes/no questions	84.6

As indicated in the table, we were able to effectively synthesize all of the expressions under consideration effectively, with the greatest success in Y/N questions. Good news was somewhat more successfully synthesized than bad news, although both of those expressions performed significantly better than chance. Neither pitch nor duration modification was performed to achieve the prosodic targets in the output speech in generating any of the above expressive states.

8. DISCUSSION

We have described the IBM Expressive TTS System, which, in addition to the default, neutral style, is capable of conveying good news, conveying bad news, and asking a question appropriately.

In our current system, markup is generated by the user or client application and is used as the vehicle for conveying the desired expression to the TTS engine. As no markup language is sufficiently rich to facilitate the specification of a desired expression, we extended SSML hierarchically, to enable convenient specification of expressive speech. In a more advanced system, rather than relying entirely on the user or client application to supply the markup, an appropriate expression could be detected automatically from the semantic content of the text; the synthetic output could then be generated to reflect that expressive state.

Similarly, given the user-specified or automatically-inferred expression desired, paralinguistic events could be automatically inserted into the synthetic audio to reinforce the desired expression.

We plan to merge the corpus-based approach to generating synthetic speech described in this paper with our alternate, prosodic phonological approach soon. We expect that the marriage of these two approaches will yield even more resounding differences between the default and expressive systems. We also intend to add more expressions to our system.

9. REFERENCES

[1] Eide, E. et al. *Recent Improvements to the IBM Trainable Speech Synthesis System.* Proc. ICASSP 2003, Hong Kong. Volume 1, pages 708-711.
http://www.research.ibm.com/tts

[2] Black, A.W. and K. Lenzo. Building Voices in the Festival Speech Synthesis System. http://www.festvox.org

[3] http://www.research.att.com/projects/tts/demo.html

[4] http://www.rhetorical.com/cgi-bin/demo.cgi

[5] Eide, E., et al. *Multilayered Extensions to the Speech Synthesis Markup Language for Describing Expressiveness.* Proc. Eurospeech 2003. Geneva, Switzerland.

[6] Speech Synthesis Markup Language Version 1.0. W3C Working Draft. December, 2002.
http://www.w3.org/TR/speech-synthesis

[7] Eide, E. *Preservation, Identification, and Use of Emotion in a Text-to-speech System.* IEEE Workshop on Speech Synthesis. September, 2002. Santa Monica, CA, USA.

[8] Silverman, et al. *ToBI: A Standard for Labeling English Prosody.* Proc. ICSLP, 1992, Banff, Alberta, Canada.

[9] ToBI online summary.
http://www.ling.ohio-state.edu/~tobi

[10] Pitrelli, J. F. and E. M. Eide. *Expressive Speech Synthesis Using American English ToBI: Questions and Contrastive Emphasis.* Proceedings ASRU. December 2003. St. Thomas, U.S. Virgin Islands.

[11] Bulut, M., S. Narayanan, and A. Syrdal. *Expressive Speech Synthesis Using a Concatenative Synthesizer*. Proc. ICSLP 2002, Denver, CO, USA.