The ModelTalker System

H. Timothy Bunnell[1,2], *Jason Lilley*[1,2], *Chris Pennington*[1], *Bill Moyers*[3], *& James Polikoff*[1]

[1] Speech Research Lab, Nemours Biomedical Research, Wilmington DE, USA
[2] Department of Linguistics, University of Delaware, USA
[3] AgoraNet Inc., Newark DE, USA

{bunnell,lilley,penningt,polikoff}@asel.udel.edu, moyers@agora-net.com

Abstract

The ModelTalker TTS system has recently been largely rewritten to change its design to make full use of information derived from talker-specific HMMs that are trained in optimizing phonetic transcription and alignment. Because this system is substantially different than the system last used in a Blizzard challenge in 2005, we decided to participate once again. This allows us to both compare performance with the previous version on similar tasks, and with the latest cutting edge TTS technology. The current version of ModelTalker appears to be comparable with the previous version in segmental intelligibility and substantially improved in the naturalness of its synthetic output.

Index Terms: voice banking, speech synthesis, unit selection

1. Introduction

ModelTalker is a unit selection text to speech (TTS) system that has been developed in conjunction with a broader application suite for use in *voice banking*, a process in which users who are at risk for losing the ability to speak record a corpus of their own speech for later use in a communication aid or Speech Generating Device (SGD). Thus, the broad project goals are to allow users who typically have little or no knowledge of speech technology or acoustic phonetics to successfully record a corpus of speech that is large and varied enough to support concatenative synthesis.

Because many users in the target audience for this technology—primarily patients with neurodegenerative diseases such as amyotrophic lateral sclerosis (ALS)—already are experiencing mild dysarthria when their disease is diagnosed, it is impractical to consider recording the very large corpora (equivalent to several hours of speech) that are typically used for unit selection synthesis. Instead, the ModelTalker corpora typically comprise less than an hour of running speech and are structured to provide an extended form of diphone coverage for English. The use of relatively small corpora also dictates that the ModelTalker TTS system must be able to manipulate intonation and timing in its waveform generation stage, although its unit selection logic strives to minimize the need to use f0 and timing modification, and users are able to turn off signal modification entirely if they wish.

The first version of the ModelTalker TTS system was a participant in the 2005 Blizzard challenge[1]. That system performed relatively well in terms of word error rate (WER). In overall WER, it was the second best system tested that year. However, it performed poorly on the mean opinion score (MOS) tasks where it was generally the lowest ranked system. Thus, while the synthetic speech produced by ModelTalker was very good in intelligibility, it sounded very unnatural.

We attributed the poor MOS scores for ModelTalker to two factors, both of which stemmed from the fact that stimuli for the 2005 challenge were generated with full signal processing enabled in the system. First, ModelTalker was synthesizing both the intonation and timing of all speech. That is, rather than use timing and intonation estimates merely as targets in the unit search, the system imposed its values via PSOLA processing on the output speech. Several deficiencies in the prosodic models, particularly with respect to segment durations in consonant clusters led to somewhat unnatural sounding speech timing. The second problem was that we later found several bugs in the implementation of the PSOLA signal processing that led to harsh buzzing due to replicating very brief segments many times.

In the time since 2005, the ModelTalker system has undergone significant changes. Many of these were related to the recording program and process involved in capturing acceptable corpora from novice users in a home or clinic setting and to a redesign of the synthesizer front end. However, within about the last 18 months, we have also completely rewritten or significantly revised both our waveform synthesis engine and the software that automatically constructs a synthesis database from recorded corpora. These latter changes were intended, in part, to improve the naturalness of ModelTalker voices while maintaining or improving intelligibility as well.

In the following, we describe the current ModelTalker TTS system, concentrating on the changes that have been made to the database construction and unit selection process. Based on the overall Blizzard results, it appears that these changes have resulted in improved naturalness. It is more difficult to assess progress with regards to intelligibility, but certainly the system's WER is competitive with that of many other systems.

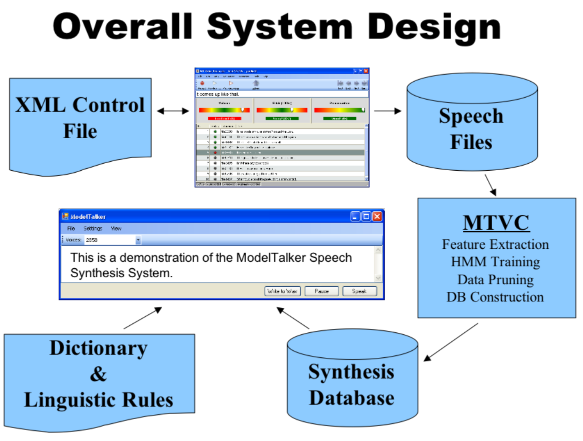
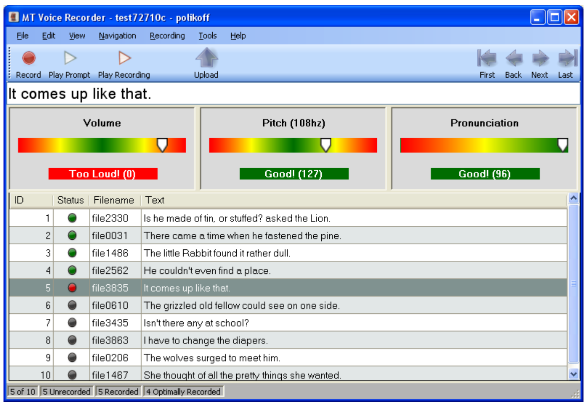


Figure 1. ModelTalker overall design to guide recording and generate a unit selection TTS voice from the recorded corpus.

2. ModelTalker System

The overall design of the ModelTalker system is illustrated in Figure 1. It comprises three distinct components: 1) the ModelTalker Voice Recorder (MTVR) program that is used to assist in recording corpora; 2) the ModelTalker Voice Constructor (MTVC) program that builds an appropriately structured synthesis database from a corpus of recorded utterances; and 3) the ModelTalker TTS system itself (hereafter, MT) which consists of a DLL or shared object library compatible with Windows, Mac OS X, and Linux, a SAPI 5.1 interface for Windows systems, and a light-weight user interface to the main functions of the primary library.

The MTVR program (Figure 2) is guided by an XML control file that lists, among other things, the utterances to be recorded and the preferred order in which to record them. This control file is updated by MTVR to maintain a record of device settings and the recording status of each utterance in



It comes up like that.

Too Loud! (0)

Good! (127)

Good! (96)

ID	Status	Filename	Text
1	●	file2330	Is he made of tin, or stuffed? asked the Lion.
2	●	file0031	There came a time when he fastened the pine.
3	●	file1486	The little Rabbit found it rather dull.
4	●	file2562	He couldn't even find a place.
5	●	file3835	It comes up like that.
6	●	file0610	The grizzled old fellow could see on one side.
7	●	file3435	Isn't there any at school?
8	●	file3863	I have to change the diapers.
9	●	file0206	The wolves surged to meet him.
10	●	file1467	She thought of all the pretty things she wanted.

5 of 10 | 5 Unrecorded | 5 Recorded | 4 Optimally Recorded

Figure 2. MTVR screen illustrating user feedback meters and primary controls.

the corpus. MTVR includes a calibration procedure designed to ensure that the recording environment is acceptable, that is, that the recording volume is set appropriately, and that the background noise level in recordings is sufficiently below that of low amplitude speech segments such as /h/ and /f/. The calibration procedure also collects statistics on the talker's speech characteristics to set limits on both the speaker-specific f0 and speech amplitude range. Once calibrated, MTVR guides the user in recording each utterance in the desired corpus. First, working from the English gloss, MTVR requests a phonetic transcription of the utterance from the MT front end. It then displays the written text of the utterance to be recorded, and plays an aural prompt of the utterance (either synthesized by MT, or a prerecorded prompt if available). After this prompt, the user initiates recording with a mouse button click or key press, speaks the utterance, and terminates recording with another button click or key press. The recorded utterance is then analyzed by pitch tracking and a forced alignment of the MT transcription to the speech signal. Results of these analyses are displayed in the form of meters on the MTVR interface as user feedback, and are used to determine if the recording was acceptable in amplitude range, f0 range, and pronunciation. The latter assessment is based on a confidence measure from the HMM forced alignment. If the recorded utterance is acceptable in amplitude, f0, and pronunciation, MTVR automatically advances to the next utterance and repeats this process. If the recording was not acceptable, users are advised to rerecord the same utterance.

While the MTVR program has received a substantial amount of the effort involved in updating the ModelTalker system, it was not directly used for participation in the Blizzard challenge. For that, recordings of the two primary Blizzard English corpora were converted to resemble the final output of the MTVR program and these were used directly as the input to the MTVC program. For the conversion process, we first ran a standalone version of the pitch tracker used by MTVR on the 16 kHz Blizzard recordings to obtain the pitch marking information used by MTVC. This pitch tracker locates the onset of each pitch period (or an arbitrary epoch during voiceless segments) and associates a binary voicing decision flag with each onset marker. The onset markers are used for both extracting PSOLA epochs and also for the pitch synchronous analysis process used by MTVC. We then used MT to phonetically transcribe the English text of each Blizzard prompt with the 56-symbol transcription set used by MT. These phonetic transcriptions were then force-aligned to the Blizzard stimuli using an alignment tool trained on TIMIT data. The transcriptions themselves, along with some prosodic tags also generated by MT and associated filenames for the Blizzard stimuli were finally formatted to resemble the output typically derived from MTVR and that is used as the input to our MTVC program.

In the following two sections, we describe in detail the analysis process used by MTVC to create a synthesis database for MT and the unit selection process used by MT for concatenative synthesis.

2.1. MTVC

MTVC implements all aspects of the process of converting a collection of appropriately formatted recordings to a synthesis database for MT. This is achieved through the following sequence of steps.

2.1.1. Feature Extraction

The first processing stage within MTVC performs acoustic feature extraction and builds the primary acoustic parameter database using pitch synchronous analyses. For research applications and non-commercial testing purposes, the acoustic database is an indexed sequence of PSOLA epochs extracted from the raw speech files, however other waveform encodings (e.g., residual-excited LPC) can also be used. Regardless of the method used to encode data for waveform synthesis, MTVC also calculates pitch synchronous acoustic features for use in training speaker-specific HMMs. The primary feature set used by MTVC is based on a principal components decomposition of a 32-channel Bark-weighted filter bank analysis of each pitch period or voiceless epoch (hereafter we refer to the pitch synchronous analysis frames as simply frames, or epochs whether it is computed from a voiced or voiceless region of the signal). This analysis is similar to a standard Mel or Bark cepstrum analysis except that the cosine terms of the DCT are replaced with a series of speaker-specific eigenvectors. Indeed, there is often considerable similarity term-for-term between the eigenvectors and the terms of a cosine expansion, although we presume that the eigenvectors afford a more optimal speaker-specific solution.

In addition to (or instead of) the primary Bark-PCA analysis, MTVC can calculate several other feature sets. These include the time derivatives of the log spectral amplitudes in each filter channel, the time derivatives of the PCA coefficients, Line Spectral Pairs and their time derivatives, and a collection of coefficients based on source features. A control file and command line options to MTVC enable/disable each possible feature data set. Each active feature set is treated as a separate data stream for use in training continuous HMMs.

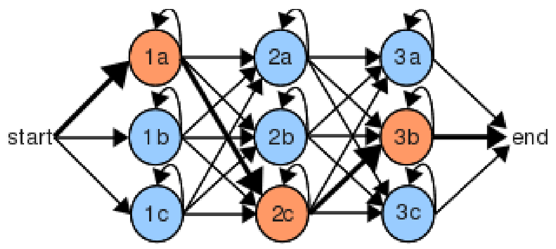


Figure 3. Example HMM structure resulting from MTVC training. Brown states connected by dark arrows indicate one possible path through the HMM.

2.1.2. HMM Training

For HMM training, MTVC uses a locally developed library of functions that allow monophone models to be trained with both continuous and discrete feature streams. The continuous acoustic feature streams extracted by MTVC may be augmented with several discrete features that capture linguistic properties including phonetic context, boundary level, stress, etc. Because these discrete linguistic features align at the phone or higher (e.g., syllable) level, they would be useless or worse if used in monophones with simple left to right state transition structures. To make use of the discrete feature augmentation, MTVC "grows" parallel-state HMMs as illustrated in Figure 3. That is, MTVC begins training with an initial strict left to right model (in Figure 3, the initial model would have 3 emitting states, but that is configurable) and then after a brief initial training period, splits some or all of the initial states to form parallel states that share connections with logically prior and subsequent states, but do not have mutual transitions. The new parallel state configuration is then trained some more, and states may then be split again. In the example illustrated in Figure 3 there have been two splits at each of three left-right serial locations, and there would thus be 27 distinct possible paths through the HMM. The brown shaded nodes with heavier connection arrows in Figure 3 illustrate one such path.

This parallel state architecture along with the use of discrete linguistic features affords an alternative to triphones, prosodically conditioned models, or both. Rather than having separate models (with possibly tied states) to represent a phone in one phonetic or prosodic context, this approach represents different contexts as different paths through the states of a single monophone model.

One area of continued research in our lab is the question of the best splitting algorithm for growing parallel state models, and the best stopping criteria. For that reason, the current version of MTVC incorporates several splitting algorithms and stopping criteria. For the 2010 Blizzard challenge, we tried just a few alternatives and screened the result by listening to SU sentences to select the result that seemed best.

MTVC requires an initial alignment of phonetic transcriptions to the speech corpus, for which we use a speaker independent TIMIT-trained aligner. This alignment information is used to estimate the initial HMM parameters that are then refined by MTVC using embedded retraining with Viterbi alignment along with the growth of the parallel state structure as described above. The end result of HMM training in MTVC is state-level alignment and indexing of all recorded speech using the final parallel-state monophone HMMs.

2.1.3. Data Pruning

After HMM training is completed, MTVC performs a series of checks to detect and eliminate segmentation errors. Of course, segmentation errors may be due to either a lack of agreement between the phonetic transcript and what was actually spoken, or to poorly aligning an accurate transcript. One level of data pruning is achieved by rejecting segments that deviate significantly (in a statistical sense) on a variety of dimensions from overall averages for the segment. MTVC uses duration, amplitude, proportion of voiced epochs, and log likelihood as dimensions for this pruning process. Segments that are identified as outliers on any of these dimensions are flagged as such and become unavailable for use in synthesis.

An additional experimental mode of pruning utilizes an HMM-based confidence measure to determine if segments are to be flagged. This measure is based on comparison of the forced phonetic alignment to an open phoneme recognition pass in which all possible phones and phone sequences are considered. In this mode, possible phone sequences may optionally be conditioned by bi-gram probabilities. This experimental confidence measure was used as the primary pruning step in generating Blizzard voices. However, more recent internal testing suggests that the original statistical pruning approach often has greater sensitivity to segmentation errors. The MTVC pruning strategy is another area of continued study and development in our laboratory.

2.1.4. Database Construction

The final stage in MTVC processing is to assemble the data files that comprise the synthesis database for MT. The version of MT that was used for the Blizzard challenge required that the following data files be built by MTVC: 1) a raw PSOLA database consisting of every windowed epoch from the original corpus; 2) the HMM models that resulted from training; 3) A file of statistics that provide duration and pitch information as conditioned by prosodic factors; and 4) indexing data that provides information about all selectable units and the location of the associated waveform epochs within the raw PSOLA database. No attempt is made to encode or otherwise compress the waveform data. In fact, because PSOLA requires segments to be windowed over a region amounting to twice the epoch duration, and because we store each epoch separately for simplicity, the raw waveform database for MT required roughly twice the storage space of the original speech data.

2.2. MT Unit selection & synthesis engine

The previous version of the unit selection and synthesis engine in ModelTalker [1, 2] indexed all biphone sequences in the speech corpus and selected segment splice locations within each phone of the biphone sequence as an integral part of the unit selection Viterbi search. This strategy was derived as a natural extension to a diphone synthesis system, wherein there are multiple versions of each possible diphone from diverse prosodic and segmental contexts and where decisions about diphone boundary locations were postponed until synthesis time. However, this approach discarded a substantial amount of acoustic-phonetic information that was gained in aligning speaker-specific HMMs to the speech corpus to estimate segmentation boundaries.

The present version attempts to make better use of all the information that is gained in training and aligning speaker-specific HMMs to the speech corpus by adopting the HMM state as the basic concatenation unit. This approach is similar in various respects to a number of previously described systems (e.g.,[3-5]) that use subphonemic waveform concatenation units. The system described by [4] is perhaps most similar in specifically defining concatenation units to be HMM states, however, there are also a number of significant differences. First, the parallel-state HMM architecture used here allows context sensitivity to be represented within the framework of monophone HMMs rather than triphones. Additionally, the clustering approach used to train triphone models as described by [4] is an agglutinative process, whereas the approach employed here progresses by state splitting. Finally, whereas [4] stored only a single prototypic region of waveform associated with each HMM state, essentially preselecting all units (c.f., [6]) the present approach stores all waveform instances associated with each state, allowing much longer stretches of originally recorded speech to be recovered from the database when appropriate for the utterance to be synthesized, but at the expense of a much more elaborate on-line selection strategy, more like that described by [3].

The more or less standard unit selection strategy, e.g., as described by [6], involves a Viterbi search over a set of candidate units, each of which is assigned a target cost based on some set of features and concatenation or join costs associated with concatenating units. The Viterbi search then finds the specific sequence of units that minimizes the combined target and join costs. In MT, the basic concatenation units may be as small as the portion of waveform aligned to a single HMM state, but with the constraint that we allow at most one segment-internal splice. This constraint results in dividing each HMM phone model into potentially multiple head/tail pairs of units. For example, with a simple 3-state model that does not have parallel states, nor allow state skipping, there would be two head/tail pairs of units to consider: the pair in which the head consisted of only state 1 and the tail consisted of states 2 and 3, and the pair with head consisting of states 1 and 2 and tail consisting of state 3. Thus, the possible units for selection are a function of the HMM architecture used by MTVC, and the constraint that only one segment-internal splice is allowed. MT allows this domain to be further constrained by specifying that only certain locations within the logical left to right sequence of states may contain splices. For instance, if using four-state models without skipping or parallel states, one could specify that the only acceptable splice location would be between the second and third states. This would effectively constrain MT to mimic the structure described by [3] in which segments are divided into left and right half-phones.

To organize the search process efficiently, MT first builds a search graph based on the desired phone sequence, the HMM structure defined from MTVC, and any splice location constraints. This graph contains one node for each possible head and tail unit as described above. Each node is then populated with all possible candidate units for that node, the candidates are ranked by target cost, and pruned to at most some predefined maximum number of candidates. In the version of MT used for the Blizzard challenge, candidates were further ranked and pruned into a series of tiers, with the top tier corresponding to candidates that were perfect matches to the desired target unit in triphone context, stress, boundary level, and pitch accent. The prosodic features were based on the syllable from which the unit was drawn. The second tier level corresponded to units that matched the desired triphone context but differed on one or more prosodic features from the ideal target. Subsequent tiers ranked candidates per biphone context, and so forth.

Table 1 lists the features used in calculating target costs. Most of the features listed in Table 1 are relatively self-explanatory, however, a few merit further description. For the phonetic context, we use a function that provides a phonetic distance between any two segments in the MT phonetic inventory. The phonetic distances between the left and right context segments of the candidate segment and those of the target segment context are summed to calculate the target cost that is multiplied by the cost weight. We have experimented with several phonetic distance metrics including those based on perceptual data, linguistic features, and physical acoustic properties. For the Blizzard challenge, our distance measure was based on the number of shared phonetic features and sonority.

Table 1. Features used to calculate target costs.

F0	Weight applied to normalized deviation in average segment F0 from the F0 predicted by prosodic models.
Duration	Weight applied to normalized deviation in segment duration from the duration predicted by prosodic model.
Stress	Weight applied to deviations from desired segment stress.
Boundary Level	Weight applied to deviation from desired boundary level.
Phonetic Context	Weight applied to phonetic distance between target and candidate context segments.
Pitch Accent	Cost associated with selecting a unit that differs in pitch accent from the target.
DHMM Features	Weight assigned to observation probability of candidate state(s) given contextual factors.
Robustness	A weight that favors “robust” segments.

Two other target features are somewhat unique to MT. The DHMM features provide observation and transition probabilities for each parallel state of a complex HMM given discrete contextual factors such as phonetic context, and prosodic features. The robustness feature was added specifically for use in Blizzard. Segments were flagged as robust if they exceed average amplitude and duration for the segment by 1 standard deviation or more. This feature was added to provide a possible enhancement for noise masking conditions.

Once all candidate units have been attached to nodes of the search graph and given target cost values, the graph is searched for the path that yields the minimum summed target and transition cost normalized by the number of nodes traversed in the path (skippable states and segments can result in paths of different lengths being considered). The join costs associated with this search are listed in Table 2.

Table 2. Features used to calculate join costs.

F0	Cost associated with F0 discontinuity.
State PDF	Cost applied to the Hellinger Distance (or KL Divergence) between state PDFs at a splice point.
Spectrum	Raw spectral discontinuity cost.
Transition Probability	Cost associated with the normalized state-to-state transition probability at a splice.
RMS Ampl.	Amplitude discontinuity cost.
HMM Path	Cost associated with the within-phone path probability that would result from a splice.
Discontinuity	Additional cost of selecting any unit out of its original acoustic context.

Several of the join costs used by MT are commonly used by most if not all unit selection systems. For example, join costs associated with the difference in f0, RMS amplitude, and spectral structure across the join are very commonly used in one form or another. MT uses two forms of spectral join cost. One is based on the KL divergence of the PDFs associated with the states to be joined. The other is the Euclidean distance between the magnitude spectra associated with the acoustic analysis frames at the edges of the units.

To better represent the dynamic structure across a join, most differences are calculated as a composite of the difference between the last epoch of the left unit and the epoch before the first epoch of the right unit plus the difference between the first epoch of the right unit and the epoch after the last epoch of the left unit. Thus, the join costs are based on differences between epochs of the potential synthetic output and the epochs they are replacing in the original recorded speech.

Two additional join costs are noteworthy in the MT implementation. One is the path join cost, which is based upon the mean spectral difference in states along the paths for the two segments in which an internal splice is considered. The second is the transition probability cost. For possible splices within a segment, this is merely the transition probability between the prospective spliced states normalized by the total non-self transitions from the left state. Neither of these costs is applied to joins at the edges of segments.

3. Blizzard tasks

Voices were built using MTVC for the two English hub tasks and for two spoke tasks. The standard American English ModelTalker dictionary was modified for these tasks to adjust for the talkers' RP pronunciation. About 225 words were modified (e.g., *been, again, schedule*), where significant vowel differences could lead to poor segment training and unpredictable inaccuracies in phonetic labels. The other primary change from the standard AE usage of MT was to make the final RX (the MT symbol for the syllable final /r/ allophone) segment skippable for both HMM training/alignment, and for synthesis. This allowed MTVC to skip the RX segment in words where the RP pronunciation resulted in simply schwa. Similarly, in the synthesis engine search for units, MT was able to explore and use utterances in which the nominal RX was skipped.

For the two spoke tasks ES1 and ES2, we explored a number of possible alterations for both the unit selection and signal post-processing. Most of these did not perform well and were not used in the final voices. For example, an attempt was made to generate a peak-enhanced version of the EH1 inventory for use in task ES2 using procedures similar to those described in [7], however preliminary testing of the resulting database suggested that it did not improve, and may have reduced intelligibility of the voice in both quiet and noise. The stimuli finally submitted for this task differed from those submitted for the EH1 task only in that the robustness feature was heavily weighted for unit selection.

For the ES1 task, we attempted to map the eigenspace of a voice generated with EH1 stimuli onto the eigenspace obtained by analysis of the first 100 sentences of the EH2 stimuli. This mapping did not succeed well enough to recognize the resulting voice as that of the EH2 talker and was dropped from the challenge. We interpreted the instructions for the ES1 task to require that specifically the first 100 sentences of the ARCTIC inventory must be used to generate the voice. Had it been possible to select any 100 sentences from the ARCTIC set, we might have attempted a simple unit selection inventory for this task.

4. Results

Overall, MT was around the middle of the pack for intelligibility as measured on SUS listening tasks with about 15% WER for all listeners and conditions, and a little more than 10% WER for native English speakers on the two hub tasks.

On naturalness and similarity measures, the MT voices were also in the average to high-average range of the voices tested. The mean similarity score for MT voices over all conditions was equal to or better than 12 of the 17 systems tested. For naturalness, MT voices scored almost exactly in the middle of the group, with an MOS that was equal to or better than 9 of the 17 systems.

Of the reference systems in the present challenge, MT is most similar to the Festival system in its general design. It is thus of interest to compare scores on MT voices with scores obtained with Festival. This comparison is presented in Table 3, which shows the WER scores for Festival and MT on the three tasks where MT voices were used in SUS tasks. There are two WER scores listed in each cell of Table 1. The score over all listeners, and the score for Native English speakers only are presented as OVERALL/NATIVE. As Table 3 shows, the two system performed comparable in all conditions except for Native listeners with voice EH2 where MT seems to have a higher WER than Festival.

Table 3. Comparison of Festival and MT on SUS tests.

Task	System	
	Festival	**MT**
EH1	0.23/0.108	0.20/0.106
EH2	0.23/0.113	0.23/0.135
ES2	0.65/0.59	0.66/0.59

5. Discussion

The ModelTalker TTS engine and associated database construction tool (MTVC) have undergone significant changes within the last 18 months. The TTS engine has been completely rewritten, and the database construction tool has been significantly revised/extended to shift from a predominantly biphone-based system using discrete HMMs for segmentation to a purer non-uniform unit selection system employing a newly developed library for mixed continuous/discrete HMM training and alignment. While many of the system changes are still experimental and undergoing continued testing and revision, the 2010 Blizzard challenge seemed an ideal way to assess progress on MT, both compared to current state-of-the-art TTS systems, and more importantly, to the performance of the previous MT system in the 2005 Blizzard challenge.

Arguably the most useful comparison between MT and other systems in the present challenge is with the reference Festival system. Overall performance of these two systems is quite similar on nearly all measures. This is particularly encouraging given that we discovered a number of bugs in the calculation of target and join costs subsequent to creating Blizzard stimuli and believe that the results obtained here for MT represent more of a performance floor than ceiling for the approach implemented in MT.

In the 2005 Blizzard challenge, the MT system was comparatively strong in intelligibility as measured by SUS listening tasks. The MT system did not rank as high in the present challenge; however, this difference in rank appears to be more due to significant improvements in TTS technology generally than to any decline in the intelligibility of the MT system. Indeed, the WER scores for MT in the present challenge are very comparable to those of the system tested in 2005, despite the fact that voices in the present challenge were recorded by talkers of the British English RP dialect, a variant for which MT is not optimized.

Naturalness was a notable weakness of the MT stimuli generated for the 2005 challenge. One of our main goals in modifying MT has been to address this weakness, and in this regard, we are quite encouraged by the results of the 2010 challenge, which show that MT voices are competitive with other laboratory systems of similar design in both naturalness and perceived similarity to the original talker.

6. Conclusions

Participation in Blizzard 2010 has been a valuable exercise that has allowed us to assess and document progress towards the next major revision of the ModelTalker system.

7. Acknowledgements

Development of the current version of ModelTalker has been supported by grant R42-DC006193 from NIH, with additional support from NIDRR grant H133E080003, and Nemours Biomedical Research.

8. References

1. Bunnell, H.T., et al. *Automatic personal synthetic voice construction*. in *INTERSPEECH-2005*. 2005. Lisbon, Portugal.
2. Bunnell, H.T., S.R. Hoskins, and D.M. Yarrington. *A biphone constrained concatenation method for diphone synthesis*. in *SSW3-1998*. 1998. Jenolan Caves House, Australia.
3. Beutnagel, M., A. Conkie, and A.K. Syrdal. *Diphone synthesis using unit selection*. in *SSW3-1998*. 1998. Jenolan Caves House, Australia: ISCA.
4. Donovan, R.E. and P.C. Woodland, *Improvements in an HMM-based speech synthesizer.* Proceedings Eurospeech 95, 1995: p. 573-576.
5. Hauptmann, A.G., *SPEAKEZ: A first experiment in concatenation synthesis from a large corpus*. Proceedings Eurospeech 93, 1993: p. 1701-1704.
6. Zen, H., K. Tokuda, and A.W. Black, *Statistical parametric speech synthesis.* Speech Communication, 2009. **51**(11): p. 1039-1064.
7. Bunnell, H.T., *On enhancement of spectral contrast in speech for hearing-impaired listeners.* J Acoust Soc Am, 1990. **88**(6): p. 2546-56.