Building an English Unit Selection TTS System: Internals and a Practical Guide

B216282_4883

***Abstract*—This report explores the construction of a unit selection voice for text-to-speech (TTS) synthesis, focusing on the waveform generation stage. We develop a voice for speaking aloud dessert recipes, utilizing data collected from websites. We discuss a greedy algorithm for automatic script selection with diphone coverage maximization. We discuss the recording process, automatic corpus segmentation, and the inclusion of linguistic features and acoustic parameters in the unit selection algorithm. Two experiments with 39 non-expert listeners with 60% of native English speakers evaluated the impact of various factors on perceived naturalness and intelligibility. A comparison of the REAPER pitch marking and F0 estimation algorithm to Festival's equivalent algorithms found no significant preference for naturalness among listeners. The second experiment assessed the effect of data quantity and domain adaptation, concluding that additional in-domain data improve naturalness and intelligibility within the domain. Overall, our findings suggest that unit selection TTS systems can produce natural and intelligible synthesized voices with minimal effort, given sufficient data with modest audio quality and phonetic coverage.**

***Index Terms*—Speech Synthesis, Festival, Unit Selection, Speech Databases**

I. Introduction

In the era of rapidly advancing artificial intelligence and speech technology, text-to-speech (TTS) synthesizers play a crucial role in various applications, including virtual assistants, audiobook narration, and accessibility tools for the visually impaired. One of the most significant challenges in developing TTS systems is creating natural and intelligible synthesized voices that closely resemble human speech. Among various synthesis techniques, unit-selection-based TTS systems have shown great promise in producing high-quality voices which can be synthesized with data recorded with a modest recording setup and with a few hundred utterances with sufficient phonetic coverage.

Unit selection TTS systems build synthetic speech by selecting and concatenating small units of recorded speech, such as phonemes or diphones, from a large database of pre-recorded speech samples. The quality of the generated speech depends heavily on the quality and variety of the recorded samples in the unit selection database. As the voice is synthesized from recorded samples, many properties of speech such as prosody, segment duration etc. do not need to be explicitly modelled [1]. A well-designed unit selection voice can provide a high level of naturalness and intelligibility, as the natural segment duration and prosody is inherent in the unit selection database.

This report investigates the process of building a unit selection voice for TTS synthesizers, focusing primarily on the waveform generator stage of the synthesis pipeline. We present an overview of unit selection synthesis, discuss the main components of the unit selection algorithm, and explore techniques for optimizing voice quality. We further offer practical recommendations for data collection, processing, and algorithm design using Festival [1], an open-source, general-purpose, and multi-lingual TTS to create highly natural and intelligible synthesized voices using unit-selection-based TTS systems. Furthermore, we delve into the design and evaluation of a TTS system tailored for speaking aloud dessert recipes in the cooking domain, sharing insights from two experiments involving subjective and objective evaluations.

This report is organized as follows. Section II describes the process of voice building. The process of data collection, script design, recordings and labelling for recording speech database for concatenative unit selection is described in Section III. Section IV reports the various listening tests conducted and their results. Finally, we conclude with a summary and discussion of findings in Section V.

II. Building Synthetic Voice

The process of building a synthetic voice could be broken down into three major steps:

- **Text analysis:** From raw text to identified words and basic utterances.
- **Linguistic Analysis:** Finding pronunciations of the words and assigning prosodic structure to them: phrasing, intonation and durations.
- **Waveform Generation:** From a fully specified form (pronunciation and prosody) generate a waveform

The synthesized speech is implemented using **multisyn** unit selection algorithm [2] where a target utterance structure is predicted which forms a target specification. From the inventory of recorded units, a list of suitable candidates is proposed for each target unit. The best sequence of candidates is found by minimizing target and join costs. Figure 1 provides an overview of the unit selection TTS system.

A. Target Construction

For synthesizing unit selection speech, a target utterance structure or target specification is constructed. In earlier diphone synthesis systems, a lot of information such as pitch, duration, intensity etc. need to be predicted, which is not strictly required for unit selection synthesis. A lot of the mentioned information is inherently present in the recorded units, which will later become the part of unit selection database. Hence, only basic linguistic resources such as pronunciation lexicon, phrasing model etc. are required for unit selection synthesis.

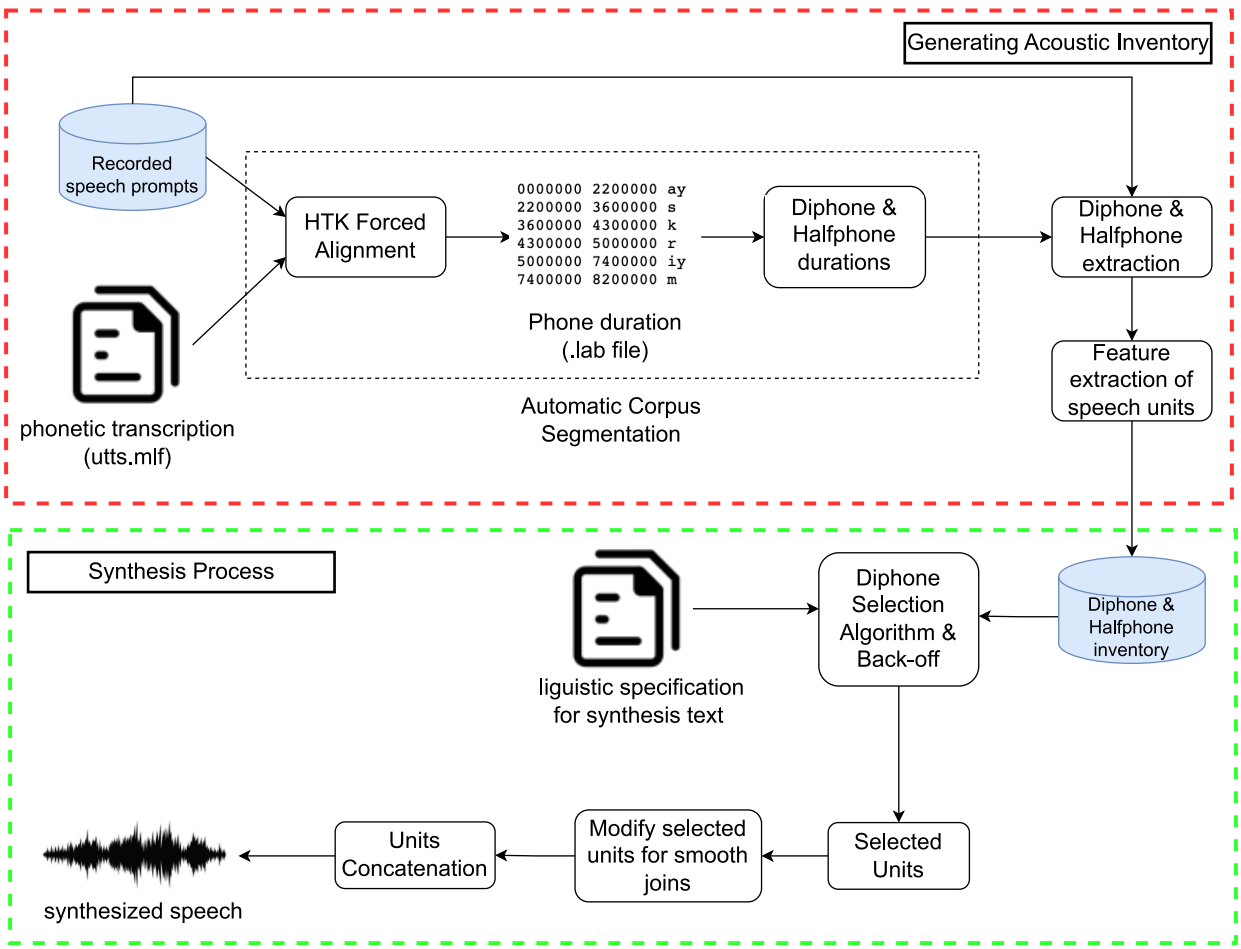

Fig. 1: Overview of unit selection TTS system.

B. Target Cost

The target cost measures how well a candidate unit from the database matches the desired target unit in terms of linguistic or prosodic features. The lower the target cost, the better the match between the candidate unit and the target unit. By adjusting the weight of the target cost, you can influence the importance of finding a good match between the desired and available units in the database.

$$C_{\text{target}}(u_i, t_i) = \sum_{k=1}^{K} w_k d_k(u_i, t_i), \tag{1}$$

where u_i is the candidate unit, t_i is the target unit, K is the total number of features, w_k is the weight for the k-th feature, and $d_k(u_i, t_i)$ is the distance function for the k-th feature. The weights adds a penalty if a feature does not match the target specification.

Because of the non-uniform distribution of phones, the total number of candidates for each target unit can be quite large. Pre-selection can be employed to limit the number of candidates for each target unit, in which case it acts as a filter for passing only the most suitable candidates for which join costs will be computed. This can consequentially speed up the search significantly by restricting the search space.

The distribution of diphones takes the form of long-tail Zipf-like distribution, which can be seen in Figure 3. This means that complete coverage of all the diphones is not guaranteed during the script design phases. Another possibility could be that the words are spoken in pronunciation which is different to that predicted during script design or the text to be synthesized has out-of-domain words. Hence, it is possible that we do not have any suitable candidate corresponding to a target unit. In such cases, some backing-off must be performed. Backing-off procedures include a possible list of manually written substitution rules which can be referred to in case of missing diphones such as vowel reduction for full vowels. Festivals also provide a `phone_substitution` list, which could be used for this purpose. Another way to implement backoff is to substitute with surrounding diphones to preserve the continuity of the phone sequence. We can also switch to half-phones as target units to provide suitable candidates and chose the best candidate which provides the lowest join cost.

C. Join Cost

The join cost measures how well two adjacent candidate units can be concatenated to form a smooth transition without any audible discontinuities. The lower the join cost, the better the concatenation between the two units. By adjusting the

weight of the join cost, you can influence the importance of achieving smooth transitions between the selected units.

$$C_{\text{join}}(u_i, u_{i+1}) = \sum_{k=1}^{K} w_k d_k(u_i, u_{i+1}), \tag{2}$$

where u_i and u_{i+1} are adjacent units in the sequence, K is the total number of features, w_k is the weight for the k-th feature, and $d_k(u_i, u_{i+1})$ is the distance function for the k-th feature.

The join cost has three subcomponents which are equally weighted: pitch, energy and spectral mismatches. The join cost is calculated by calculating Euclidean distances between 12 MFCCs of the potential join to compute spectral discontinuity. Similarly, pitch and energy mismatches are calculated using f0 and energy coefficients.

The join cost could significantly vary depending on the phonetic class of the candidate units. For example, joins are often inaudible when the phonetic class is fricatives. Similarly, we can also identify poor joins, for example, one of the most costly mismatches is incurred between a join of voiced and unvoiced speech, hence, we can have preventive checks for such joins.

There's often a trade-off between target cost and join cost. It's essential to find a balance between these two factors to achieve high-quality speech synthesis. A well-tuned system will prioritize both target and join costs to ensure an accurate unit selection and smooth concatenation, resulting in natural-sounding speech. The optimal weights for target cost and join cost can vary depending on the specific TTS system, the speech database, and the application's requirements and should be tuned on the basis of statistical or perceptual evaluations on the data from the target domain.

D. Unit Sequence Search

The target cost measures how well a candidate unit matches the target features of the input text, while the join cost measures how well the candidate units fit together acoustically. By combining these two costs, the unit selection algorithm can select the best-suited units and concatenate them to generate natural-sounding speech. The Viterbi search algorithm aims to find the optimal unit sequence $\mathbf{u}^* = (u_1^*, u_2^*, \ldots, u_N^*)$ that minimizes the total cost:

$$C(\mathbf{t}, \mathbf{u}) = \sum_{i=1}^{N} C_{\text{target}}(u_i, t_i) + \sum_{i=2}^{N} C_{\text{join}}(u_{i-1}, u_i) \tag{3}$$

Here $C_{target}(t_i, u_i)$ denotes the target cost between candidate unit u_i and target unit t_i and $C_{join}(u_{i-1}, u_i)$ is the join cost between candidate units u_{i-1} and u_i. The optimization to find the best unit sequence is done by using the Viterbi search algorithm [3].

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}}\, C(\mathbf{t}, \mathbf{u}) \tag{4}$$

E. Waveform Concatenation

Selected candidate units from the unit selection algorithm are concatenated in a pitch-synchronous, overlap-and-add manner. To ensure that concatenation is pitch-synchronous we need to predict accurate pitch marking which is essential for preserving the natural prosody of the synthesized speech. For unit selection in principle, we need to modify F0 and duration of the candidate units. However, if the unit selection database is big enough, we may not need any prosodic modification as the selected target may be close to the desired prosody.

III. Recording Speech Database

A vital component of unit selection speech synthesis is the design of the voice. Regardless of how well a system is executed, the generated speech can only be as excellent as the data comprising the voice inventory. Typically, speech synthesis datasets are recorded professionally by a single speaker in a controlled environment under supervision. The amount of data required for TTS depends on the chosen model. The type of data or domain is another important aspect which is given importance while building voice inventory for unit selection-based TTS. This section discusses the various design decisions taken to build the unit selection synthesizer which is capable of reading dessert recipes, which limits the domain of the synthesizer. Figure 2 provides an overview of the process of corpus creation, automatic script/prompt selection and recording prompts.

Building a voice consists of the following processes:

- Data collection
- Automatic script design
- Record the speech with the help of voice talent
- Automatic corpus segmentation and labeling of the recorded utterances
- Extract pitch marks
- Extract parameters for voice building
- Build a unit selection synthesizer
- Test and evaluate the voice

A. Data Collection

Data for the recording was collected from `tastykitchen.com`, which allows for the scrapping of the content on the website. Over 1000 dessert recipes were scraped and then they were split into sentences and the top 50,000 sentences were selected as sentence inventory, from which the prompts will be selected for recording.

The collected utterances consist of non-standard words (NSW), some of the common NSWs are mentioned below:

- abbreviation of units such as kg (kilogram), g (gram), cm (centimetre), mm (millimetre), °C (degree Celsius), °F (degree Fahrenheit), tsp (teaspoon), tbsp (tablespoon)
- dimensions - 20x10 tray (twenty by ten tray)
- Fractions: 1/4 (quarter), 1/2 (half)
- Ratios: 1:2 (one is to two)
- Unicode characters (Emojis, quotes, special characters etc.)

Some of the above-mentioned NSWs are specific to the domain of cooking and their expansion rules might be missing from the Festival's frontend which converts the `text` to `token`, which can then be used to generate phonetic transcription as per the chosen lexicon. If this step fails we might

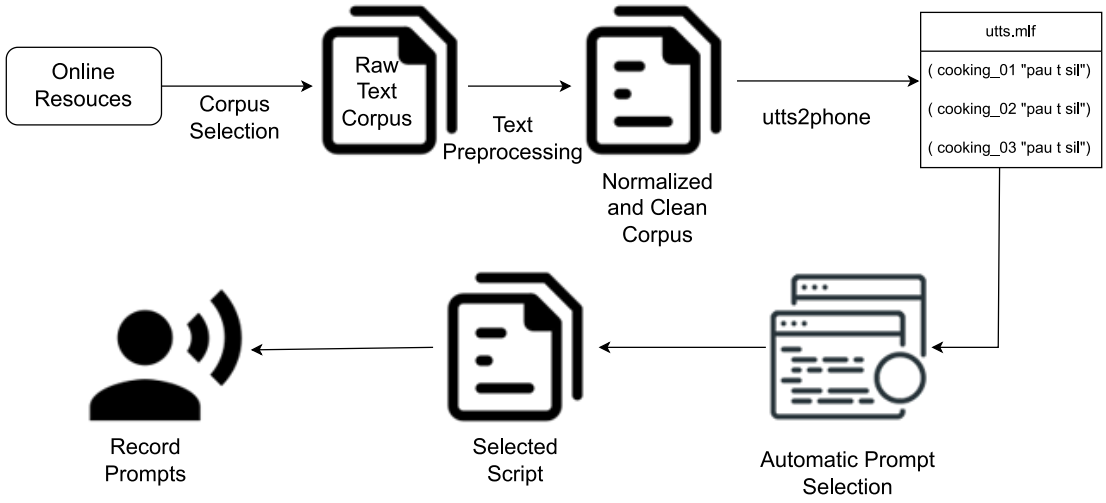

Fig. 2: Overview of corpus creation, automatic script selection and recording

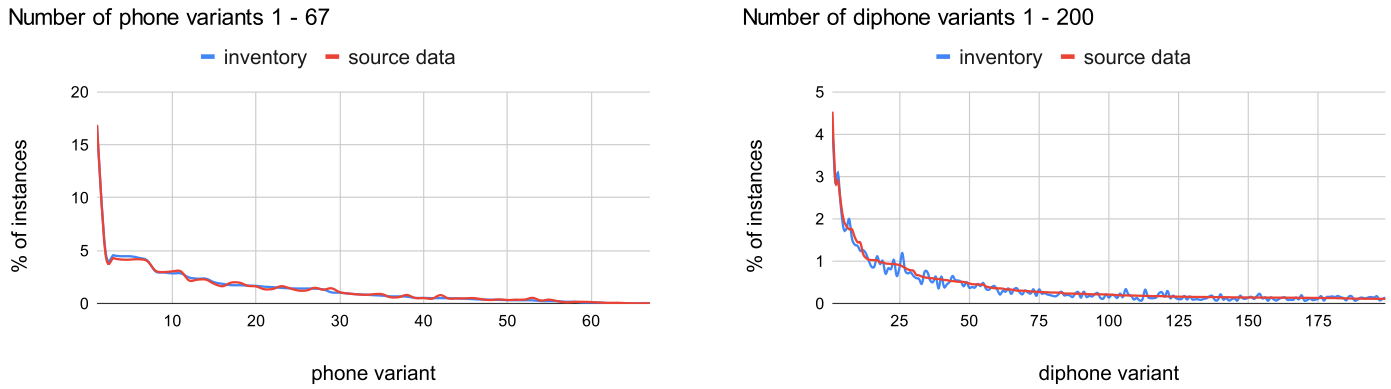

Fig. 3: Distribution of all the phoneme distribution (left) and the most frequent diphone variants (right) in the inventory for Cooking - GAM data

Data	Coverage Statistics		
	Number of Phones	Number of Diphones	Theoretical Diphone Coverage
ArcticA (GAM)	66/78	1093	25.09%
Cooking (GAM)	67/78	1143	25.46%
Cooking (RPX)	60/62	1073	29.81%

TABLE I: Phone and diphone coverage in source data (corpus)

not get the correct tokens for the utterances and there will be a mismatch in pronunciation of which we pronounce and which is generated by the festival frontend and this will lead to failure in the alignment step. We must ensure that the various terminologies are expanded such that they can be converted into correct tokens by the festival front end.

B. Automatic script selection

The speech database was designed for English unit selection synthesis using diphones. The script design followed the idea of the English CMU ARCTIC databases [4] [5]. Starting with our initial text corpus of 50 thousand utterances, we selected utterances in the script that satisfied the following conditions:

- be at least 10 letters
- be between 5 and 15 words
- only contains characters from the english alphabet or any of the english punctuation symbols
- starts with a capital letter
- end with a punctuation symbol

The choice of pronunciation dictionary/lexicon determines the phone set which will be used by the TTS system. Festival provides a pronunciation dictionary for General American English (GAM), British English (RP) and Scottish English (Edinburgh). As the speaker is a non-native English speaker, however, the choice of pronunciation dictionary was made on the basis of the closeness of speech to these three accents, which was identified that the pronunciation of the majority of words is closer to General American English (GAM). Hence, for the purpose of converting utterances to phones, the festival's `uniliex-gam` phone set was used for voice-building purposes unless specified.

Festival's pronunciation dictionary provides good coverage of words, however, there might be out-of-vocabulary (OOV)

words, for which it may use `letter-to-sound` rules, which may not be correct. Hence, all the utterances, which consist of OOV words with frequency < 20 were not selected for further prompt selection stage. We found a total of 26 OOV words whose pronunciation was not present in `unilex-gam`, we added them manually to the dictionary by identifying similar-sounding words from the dictionary and using parts of their pronunciation such that we have consistent pronunciation for OOV words.

To ensure that we are able to record the most possible context-sensitive combination of phones, TTS scripts generally maximize some sort of phone coverage. Among the various metrics which could be used to ensure the coverage of the TTS script such as phone, diphone, triphone etc. coverage [4]. For prompt selection, we have a two-stage algorithm where in the first stage we greedily select sentences to have the best diphone coverage. The first stage ensures that we have coverage of at least 1 unique diphone in our script (including rare diphones). We aim to collect 600 utterances from the corpus which provides maximum diphone coverage. The first stage of the greedy algorithm ranks the utterances with maximum diphone gain normalized by the number of unique diphones in the utterance. We found this way of normalization selects sentences which are closer to the length distribution of the original corpus. The first stage was completed with 278 utterances covering 100% diphone coverage with at least a single diphone of each type. The second stage ensures that we have a coverage of different prosodic variations of common phrases from the domain, which will help with selecting context-sensitive units which will help improve the naturalness of the synthesized voice. The second stage of the algorithm then completes with 322 utterances, covering the utterance with the maximum gain in unique common phrases normalized by the number of words in the utterance.

The resulting scripts' phone and most frequent diphone distribution are shown in Figure 3 which shows that the distribution of phones and diphones in our script/inventory is similar to our source data (corpus). The average length of sentences is 9 ± 3 words which are suitable for recording purposes. Table I shows the phone and diphone coverage for the CMU Arctic A [4] and domain-specific dessert-recipes (cooking) scripts with GAM and RPX pronunciation lexicon. The greedy algorithm for automatic script generation tries to achieve maximum coverage of the total number of diphones present in the source data (corpus).

C. Making the recordings in the studio

Usually for building high-quality voice, speech is recorded in a recording studio with soundproofing designed to limit resonance, reverberation etc. However, for the purpose of this voice, the recording environment is a standard university accommodation room with modest soundproofing. Bose Quiet Comfort 35 headset with noise cancellation was used for recording. The distance between the microphone and the speaker's mouth was constant at all times.

The single-speaker speech database is recorded using SpeechRecorder [6], a tool designed to aid the recording of speech databases. The user was allowed to record the multiple versions of each sentence and the best instances were used. The recording sessions were conducted in multiple sessions to ensure that the speaker is able to speak in a consistent prosody through the recording of the database. Any mistakes in utterances due to wrong pronunciation or incorrect pauses or intonation are recovered by re-recording those utterances or correcting the corresponding transcription to avoid re-recording. For this work each sentence was recorded and saved as a separate WAV file with the following considerations:

- Sentences are pronounced with natural intonation with full diacritics such as to match the written form of the script.
- WAV files were recorded with 48 kHz/16-bit, single channel format, which is then downsampled to a sampling rate of 16kHz.
- Endpointing was not used as the silences at the start and end of each utterance are short and consistent.

We recorded CMU Arctic A [4] and domain-specific dessert-recipes (cooking) scripts with average utterance duration of 4.5s and 5s respectively.

D. Automatic corpus segmentation

After the script is recorded, the next step is the phonetic labeling of the recorded speech. One of the challenges is to find out the precise location of phone boundary. As we are using diphones, we do not need to place labels at the phone boundaries but instead midway between phone boundaries which works for the majority of segment types. Marking phone boundary midway between phone boundaries has the advantage that when we join phones midway, the spectrum is locally static compared to the phone boundaries and minute inaccuracies in the phone boundaries will still result in diphone boundaries which fall with the static region. Additionally, this allows us to use less precise methods for labeling phones such as automatic labeling and take advantage of the quantity of data instead of precisely tagged human labeling.

Automatic labeling is done by first converting the script to appropriate phonetic label sequences using a pronunciation dictionary. This poses two challenges. Firstly, does the phoneme sequence account for the accent of the speaker and the effect of connected speech such as vowel reductions, co-articulation etc. The second challenge is to precisely match the recorded speech to predicted phone sequences. These challenges are handled by using an accent-specific lexicon and using the linguistic analysis phase of TTS process such as adding information of closure, release, inserting pauses and silences to allow for better alignment of phonemes and recorded speech. The pronunciation variation is handled by adding alternative pronunciations of the words, which is modeled by a separate HMM framework as separate paths with the same start and end words.

The next step in automatic labeling is to align the phonetic transcription to recorded utterances. This is done by using HMMs in **forced alignment** mode, where instead of recognising the phones in the utterance, where we give correct phones and phone substitutions in the correct sequence to choose

from. Trivially, the recognition will always be correct, but as a by-product, the Viterbi algorithm produces state-observation alignment. Here, the training and testing data is the same i.e. the complete speech data. This is achieved using standard left-to-right monophone HMMs with 3 emitting states. We use 12 MFCC coefficients with energy, their deltas and delta deltas (a total of 39) features as speech parameters. A 10 ms window size with a 2 ms shift is used to generate observations which are aligned with phone states. The observations are modeled with GMMs which are gradually increased during the training process using HTKs standard *mixing up* procedure. Finally, forced alignment using the final models is used to predict labels for the recorded speech. This process produces consistent alignment and is computationally very fast. We get phone, and diphone durations as a result of this process and we can segment candidate diphones units which make up the unit selection database.

E. Extracting pitch marks

Here we use `make_pm_wave` script provided by the Festival which filters an incoming waveform with a low and high pass filter, then uses autocorrelation to find the pitch mark peaks with the specified `min` and `max` values for pitch periods. For the unvoiced section of a waveform, it fills with default pitch marks. An insufficient range of pitch period may result in too many or too few pitch marks. As the concatenation is done on pitch marks, incorrect pitch marks will leads to incorrect sections of waveforms getting concatenated will result in artefacts in the resulting speech signal. Finally, the generated pitch marks are time shifted to align pitch marks with the largest peak in each period; this is done using `make_pm_fix` script.

F. Extract parameters for voice building

After candidate units are extracted to create the speech database, the next step is to extract the features from the candidate units to be used for the unit selection algorithm. This includes building utterance structures to store all the linguistic information required for the target cost. The linguistic information such as phonetic string, a tree structure that connects those phones with their parent syllables and words, phonetic timestamps obtained by forced alignment etc. are added to the utterance structure. Other information required by join cost such as fundamental frequency, F0 and MFCC coefficients are is also extracted from candidate units and stored in separate files per utterance.

IV. Listening Tests and Results

A. Listening Test Design

Following are the types of subjective listening tests which were conducted as part of the evaluation:

- *Ranking of voices* in the order of naturalness, which is a comparison-based test. Most natural will be ranked 1st.
- *Mean Opinion Score (MOS).* A single test included listening to randomly ordered utterances, and listeners were asked to give a score of 1 (worst) to 5 (best) to rate their opinion of the utterance.

Recently, automatic speech recognition (ASR) is increasingly used to evaluate the intelligibility of text-to-speech synthesis (TTS) and results have shown such evaluation of intelligibility is consistent with human evaluation and often more reliable [7]. Whisper [8], as state-of-the-art in ASR has shown near-human performance. We use `openai/whisper-large-v2` from `huggingface` for generating transcription and consequently word error rate is calculated as an objective measure of intelligibility.

For evaluation of different synthesized voices, we selected 10 test utterances, which were not used in the synthesis of voice. These test utterances can be further divided as follows:

- *In-Domain:* 5 utterances are selected from dessert recipes.
- *Near In-Domain:* 3 utterances are selected from general cooking recipes which were not dessert recipes.
- *Out of Domain (OOD):* 2 utterances are selected from CMU Arctic B [4].

The number of test utterances was kept to 10 as the listeners were volunteers and it is expected that they are likely to abandon the listening test if the overall duration is kept longer than 10-15 minutes. 42 test responses were collected and after sanity checking 39 test responses were taken in the subjective evaluation of voices. Out of 39 tests, 60% of the listeners were native speakers of English.

B. Experiment 1: Effect of pitch marking and F0 estimation

As discussed in III-E incorrect pitch marking can lead to incorrect sections of waveforms of the target units getting concatenated which can result in artifacts which can have an adverse effect on the perceived naturalness of synthesized speech. Hence, better pitch marking should result in more natural speech. The F0 contour plays a vital role in conveying prosodic information and contributes to the naturalness of the synthesized speech. Unit selection TTS systems may face challenges in generating smooth F0 contours, especially when concatenating speech units with different F0 characteristics. Signal processing techniques, such as F0 smoothing and modification, can help mitigate these issues but may introduce artifacts if not applied carefully.

Festival's F0 estimation which uses `SRPD` (Super Resolution Pitch Determinator) [9] algorithm. We find that `REAPER` [10] consistently performs better than `SRPD` as shown in [11] which provides a detailed study of several pitch detection algorithms in simulated and noisy environments. We also did a subjective evaluation by comparing the pitch marks predicted by Festival's pitch marking algorithm and `REAPER`. Figure 4 shows an example of one such comparison, where we found that for the voiced regions `REAPER` is generally predicting pitch markings closer to the highest peak per epoch. For this reason, we chose `REAPER` to compare the effect of better or worse pitch marking and F0 estimation on the perception of naturalness of speech and objective evaluation of intelligibility.

The results of subjective evaluation of the naturalness of voices generated by Festival and `REAPER`, Figure 5 show the mean opinion scores (MOS) provided by listeners. We can

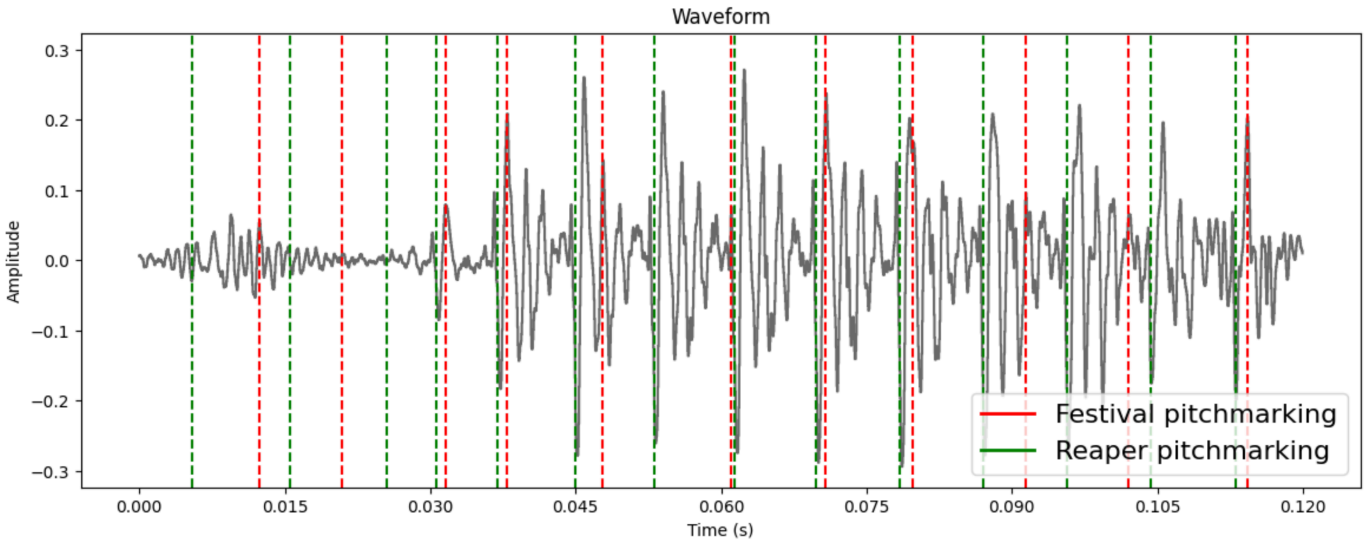

Fig. 4: Visualization of pitch marking generated by Festival and REAPER.

observe that the human voice is consistently given the highest naturalness and the MOS score for `REAPER` is slightly better than the Festival voice. We further do a significance testing, with Wilcoxon signed rank test [12] at p-level $p < 0.05$. We get a Wilcoxon p-value of $0.4488 >> 0.05$ which demonstrates no significant differences between the MOS scores. This indicates that there is no perceivable difference in the naturalness between the two voices. We also perform an objective evaluation of the synthesized voices as shown in Table II. We find that the results are consistent with the MOS scores for naturalness, however, given that we only have 10 utterances for testing, the difference in WER is deemed insignificant.

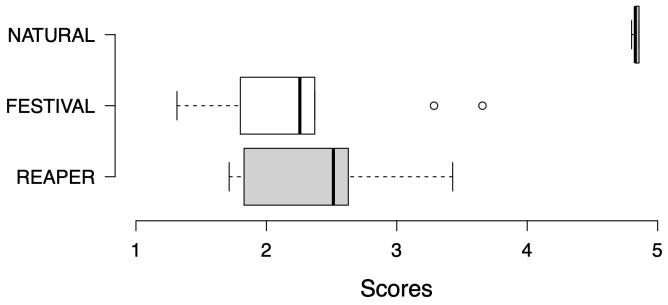

Fig. 5: Boxplot of MOS scores on the naturalness of synthesized voice using different pitch marking and F0 estimation algorithms

C. Experiment 2: Domain adaptation of voice

In this experiment, we would like to understand the effect of data size and domain of training data used to synthesize voice on the perception of naturalness by listeners. In corpus driven TTS system, domain adaptation is achieved by adding data from the target domain and synthesizing voice on the new data. [13] Also, the synthesized speech on test data from a different domain is comparatively less natural and intelligible because

	Voices		
	REAPER	Festival	Human
WER	6.50%	7.10%	2.50%
MOS	2.45 ± 0.61	2.30 ± 0.70	4.83 ± 0.02

TABLE II: Subjective and objective evaluation results for exploring the effect on naturalness and intelligibility of better pitch marking and F0 estimation algorithms

of differences in prosodic requirements such as duration of words, intonation and also missing phones and diphones from the different domain. [1]

We synthesized three voices `Cooking (278)` , `Cooking (600)` , `Cooking+Arctic (1193)`. The voices `Cooking (278)` and `Cooking (600)` are from the dessert recipes domain with 278 and 600 utterances in the training script; 278 utterances provide 100% diphone coverage and `Cooking (600)` is synthesized with 322 additional data as discussed in Section III-B. `Cooking+Arctic (1193)` is synthesized with 600 utterances from dessert recipes and 593 utterances from CMU Arctic A [4] with data from out-of-copyright books from the Gutenberg Project; which means that additional data is out-of-domain.

Figure 6 shows the results of subjective evaluation of comparative naturalness voices on the test utterances grouped by the domain of data. Table III shows the results of the objective evaluation results of intelligibility. Based on the results of listener preferences can conclude that for in-domain data the naturalness is in order `Cooking (600)` > `Cooking (278)` > `Cooking+Arctic (1193)` and adding more data helps improve naturalness and intelligibility only when the additional data is from the same domain. We can also conclude that additional OOD data helps synthesized voice adapt better for out-of-domain data as listeners show a significantly higher preference for `Cooking+Arctic (1193)` for OOD test

Voice	Test Data Type (WER)		
	In Domain	Near In Domain	Out of Domain
Cooking (278)	6.50%	8.10%	17.40%
Cooking (600)	3.20%	5.40%	13.00%
Cooking+Arctic	11.30%	7.90%	11.50%

TABLE III: Results for comparison of effect of data quantity and domain on intelligibility

utterances; however we cannot conclude that the resulting voice is of good quality as WER for OOD utterances is significantly higher than in-domain and near-in-domain cases as shown in Table III, but comparatively lower than other voices.

V. Discussion and Conclusion

In this report, we investigated the process of building a unit selection voice for text-to-speech (TTS) synthesizers, with a primary focus on the waveform generator stage of the synthesis pipeline. We began with an overview of unit selection synthesis, discussing how a well-designed system could produce high-quality voices that are natural and intelligible. We examined the main components of the unit selection algorithm, including target construction, target cost, and join cost, and how they impact the selection of candidate unit sequences. We also discussed the Viterbi search for identifying the best candidate unit sequence and how the waveform of the optimal sequence is concatenated to result in natural and intelligible synthesized speech.

Our TTS system was designed to speak dessert recipes such that it sounds natural and intelligible to humans. We collected data by scraping websites, cleaned and normalized it for recording purposes, and employed a greedy algorithm for maximizing diphone coverage. We generated a script of 600 utterances and recorded them alongside CMU Arctic A data. We then detailed the speech recording process, including setup and processing steps. Automatic corpus segmentation, using forced alignment mode of HTK-based HMM training, aligned phone sequences of utterances and recorded speech, segmenting diphones to prepare candidate units for the unit selection database. We explored building utterance structures that include linguistic features, duration information, pitch marking, F0, MFCC coefficients, and their effects on the unit selection algorithm.

We designed listening tests and analyzed the results of two experiments with 39 non-expert listeners, of which 60% are native English speakers. The first experiment evaluated perceived naturalness using the robust REAPER pitch marking and F0 estimation algorithm compared to Festival's algorithms. Subjective evaluation of naturalness using mean opinion scores revealed a slightly higher preference for the REAPER voice, but significance testing showed no significant preference. Objective evaluation with the Whisper ASR system found a 0.60% better WER for the REAPER voice, but given the test utterances' size, the difference was deemed insignificant. The

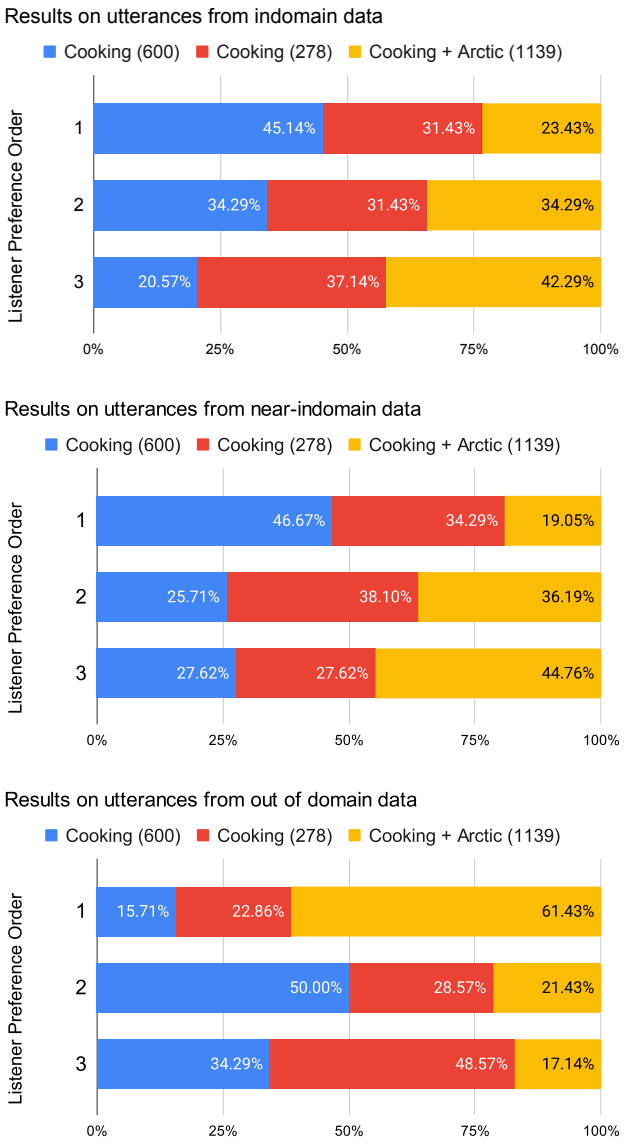

Fig. 6: Results of listener ranked preference of synthesized voice showing the effect of data quantity and domain on perceived naturalness. Lower ranks show higher preference.

second experiment assessed the effect of data quantity and domain on synthesized voices' domain adaptation. Listener preference testing for naturalness by ranking of voices and ASR-based automatic transcription was used for subjective and objective evaluations, respectively. We concluded that additional in-domain data improved in-domain and near in-domain naturalness and intelligibility, where voices synthesised with one-fourth of in-domain data performed better than those using additional out-of-domain data.

In conclusion, unit-selection-based TTS systems can create natural and intelligible synthesized voices with minimal effort. Ensuring data quality in terms of recording setup and phonetic coverage for context-sensitive diphone candidates can significantly enhance synthesized voice quality. With a large enough unit selection database, we can generate voices covering a

majority of prosodic variations within the domain, resulting in natural and intelligible speech with minimal waveform processing during candidate unit concatenation compared to other concatenative synthesis methods.

REFERENCES

[1] R. A. J. Clark, K. Richmond, and S. King, "Festival 2 - build your own general purpose unit selection speech synthesiser," in *Speech Synthesis Workshop*, 2004.

[2] R. Clark, K. Richmond, and S. King, "Multisyn: Open-domain unit selection for the festival speech synthesis system," *Speech Communication*, vol. 49, no. 4, pp. 317–330, Apr. 2007.

[3] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (from version 3.3)*, 01 2004.

[4] J. Kominek and A. W. Black, "The cmu arctic speech databases," in *Speech Synthesis Workshop*, 2004.

[5] A. W. Black and K. A. Lenzo, "Optimal data selection for unit selection synthesis," in *Speech Synthesis Workshop*, 2001.

[6] CSTR, The University of Edinburgh, "Speechrecorder." [Online]. Available: https://www.cstr.ed.ac.uk/research/projects/speechrecorder/

[7] J. Taylor and K. Richmond, "Confidence Intervals for ASR-Based TTS Evaluation," in *Proc. Interspeech 2021*, 2021, pp. 2791–2795.

[8] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022. [Online]. Available: https://arxiv.org/abs/2212.04356

[9] Y. Medan, E. Yair, and D. Chazan, "Super resolution pitch determination of speech signals," *IEEE Trans. Signal Process.*, vol. 39, pp. 40–48, 1991.

[10] D. Talkin, "Reaper: Robust epoch and pitch estimator," https://github.com/google/REAPER, 2013.

[11] D. Jouvet and Y. Laprie, "Performance analysis of several pitch detection algorithms on simulated and real noisy speech data," *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 1614–1618, 2017.

[12] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 196–202, 1945.

[13] M. Chu, C. Li, H. Peng, and E. Chang, "Domain adaptation for tts systems," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 2002, pp. I–453–I–456.