Text to Speech Synthesis using HMM-based Unit Selection and Concatenative Synthesis

Contents

1 Abstract

Text to speech synthesis is a rapidly growing aspect of computer technology and it plays an important role in various platforms of human computer interactions. In this project, we have implemented text to speech synthesis using Hidden Markov Models for unit selection and concatenative synthesis. We have used a "generic English" (Genglish) database which is simpler and has a smaller lexicon than spoken English, making it easier for us to test the efficiency of our algorithm. The aim of the project is to produce accurate speech samples that are intelligible while also sounding natural.We have made use of TTSBOX, a text to speech synthesis toolbox on Matlab [2].

2 Introduction to TTS Synthesis

Text-to-speech (TTS) synthesis refers to creation of natural sounding speech from arbitrary text. The goal of a text-to speech system is to automatically produce speech output from new, arbitrary sentences. The quality of a speech synthesizer is measured based on two primary factors – its similarity to normal human speech (naturalness) and its intelligibility (ease of understanding by the listener). A typical text to speech system has two parts – front end and back end parts. Front end part is responsible for preprocessing and symbolic linguistic representation of the text. The back end is responsible for conversion into sound.
The various methods for text to speech conversion are mentioned below [1]:

- *Formant synthesis* : In this method the formant frequencies are used to generate speech.
- *Concatenative speech synthesis* : Concatenative speech synthesis method involves production of artificial speech by concatenating prerecorded units of speech.
- *Articulatory Synthesis* : Articulatory synthesis refers to computational techniques for synthesizing speech based on models of the human vocal tract .

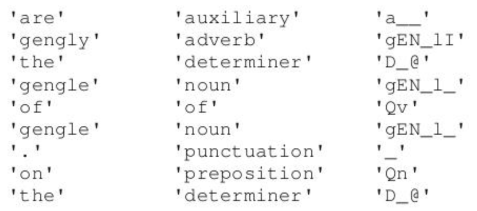
We have used the concatenative speech synthesis algorithm. The database being used is a "Generic English" database [2], or "Genglish" database in short. Genglish has no abbreviations, no arabic nor roman numbers, and no acronyms. For a deeper examination of Genglish sentences, we have created a MATLAB corpus file containing a set of 50 Genglish sentences (about 800 words) in which each word is listed with its spelling, part-of-speech category, and phonetization. Genglish has a simpler lexicon than English and has fewer words . Thus synthesis of speech is easier using phonetic units, but naturalness is a major challenge of this method .

3 Theory and Explanation of TTS Algorithm

Our TTS algorithm consists of two main parts: Training the model and Testing the Model. We have used a database of 50 annotated .wav files for training our system. We test our system by giving a new combination of words from the existing lexicon, and analyzing the result. We compare our final output with the output of Microsoft's SAPI5 TTS function, which we have taken as the benchmark for state-of-the-art TTS conversion.

3.1 Generating the Corpus

A Genglish corpus file is created containing a set of 50 Genglish sentences (about 800 words) in which each word is listed with its spelling, part of speech category and phonetization.Each word in the Genglish vocabulary is given a tag as to which part of speech category it is, like noun, verb, determiner, punctuation etc. Hence, when the text input is given to the Matlab code, it invokes a function which performs lexicon search and finds out the part of speech category of every word in the text input.



```
'are'       'auxiliary'     'a__'
'gengly'    'adverb'        'gEN_lI'
'the'       'determiner'    'D_@'
'gengle'    'noun'          'gEN_l_'
'of'        'of'            'Qv'
'gengle'    'noun'          'gEN_l_'
'.'         'punctuation'   '_'
'on'        'preposition'   'Qn'
'the'       'determiner'    'D_@'
```

Figure 1: Genglish Corpus

3.2 Preprocessing the Text Input

A TTS system converts written text to a phonemic representation, then converts the phonemic representation to waveforms that can be output as sound. It is often impossible to correctly pronounce a sequence of words in natural languages without prior knowledge of their part-of-speech, as well as of their hierarchical organization into groups, which itself also depends on the sequence of part-of-speech involved. The preprocessing for our genglish database consists of three modules: morphological analysis, contexual analysis and syntactic-prosodic analysis [2].

3.2.1 Morphological Processing

We analyise the genglish corpus and derive a morphological lexicon, Part of Speech lexicon, Phoneme lexicon and grapheme lexicon. The morphological lexicon consists of all the words encountered in the corpus as well as their possible parts of speech. The PoS lexicon consists of all the PoS tags encountered in the corpus. The phoneme and grapheme lexicons consist of all the possible phonemes and letters respectively in the corpus.

3.2.2 Syntactic Prosodic Grouping

The primary goal of a syntactic prosodic parser is to segment utterances into smaller units to increase the listener's understanding of the text. Thus, in many state-of-the-art TTS systems, prosodic phrases are identified with a rather trivial chinks 'n chunks algorithm [5] . In this approach, a prosodic phrase break is automatically set when a word belonging to the chunks group is followed by a word classified as a chink. Chinks and chunks basically correspond to function (grammatical words) and content words (lexical words) classes respectively, with some minor modifications.
Example:
I asked them if they were going home to Bangalore.
Chinks: I, if , to
Chunks: asked them, they, were going home, Bangalore.

3.2.3 Contextual Analysis of the database with Hidden Markov Models for PoS tagging

A descrete *Hidden Markov model (HMM)* is a finite state machine which generates a sequence of discrete time observations. At each time unit, the HMM changes states at Markov process in accordance with a state transition probability, and then generates observational data in accordance with an output probability distribution of the current state. This statistical time-series model is widely use in speech recognition and synthesis tasks [7]. In our text-to-speech system, HMMs are used for bi-gram Part of Speech (POS) tagging and for phoneme to speech alignment (section 3).

Contexual analysis of the genglish database is done using the bi-gram model [2]. In this model, we assume that the part-of-speech tag of a word, only depends upon the POS tag of the previous word. Thus, neighbouring words do not depend on one another but rather on the underlying POS tags. Let X = x_1, x_2, .. x_t be the sequence of words and
Y = y_1, y_2, .. y_t be the corresponding tags.
To find the most probable tags of the sequence we have the equation:

$$y = \arg\max_y p(y|x) = \arg\max_p p(x, y) \tag{1}$$

Here, y are the observable outputs and x are the hidden states.
By Bayes' Theorum, $p(x|y) = p(x_1, x_2, ...x_t|y_1, y_2, ..., y_t)$

But according to the Markov Property, the output probability of the current state only depends upon the previous state. Thus,

$$P(X|Y) = \prod_{i=1}^{n} p(x_i|y_i) \tag{2}$$

The prior computation of all emission and transition probabilities is required. This can be done by counting appearances of words and tag combinations in a corpus.The probability that category c_i emits word w_i is approximately given by the number of times w_i appears as c_i, divided by the total number of words with part-of-speech category c_i:

$$P(w_i|c_j) \approx \frac{n(w_i, c_j)}{n(c_j)} \tag{3}$$

Similarly, the bigram transition probability between categories c_j and c_i is approximately given by the equation:

$$P(c_i|c_j) \approx \frac{n(c_i, c_j)}{n(c_j)} \tag{4}$$

3.3 Training the Model

There are two parts of the model that need to be trained by minimising a cost function: the phonetisation module and the speech-to-text alignment module.

3.3.1 Phonetisation

The phonetisation (or letter-to-sound, LTS) module is responsible for automatic detection of phonetic transcription of the incoming text. Our algorithm trains Classification and Regression Trees (CART) to model how a given input text sequence can map to different phonetisation, given certain contextual features. The decision features used in the node-splitting algorithm are the letter being currently phonetised, the letter to the left and right of the letter being currently phonetised, and the PoS of the current word. The output of the CART is a phonemic symbol.

3.3.2 Text Alignment

Segmenting speech into phonemes is not an easy task, even when phonemes are known (in which case this operation is termed as alignment). Done by hand, it takes forever; done by machines, it is never completely reliable. We have used an HMM-based text-to-speech alignment system developed at TCTS Lab [4] and produced corresponding .seg files, the content of which is easy to understand: each line mentions a start, an end (in sample), and a phoneme name.Special care is taken to avoid errors due to coarticulation by choosing an optimal cost function.

From this segmented speech corpus, we have built a speech unit database, in which we have stored, for each available unit, the minimum information needed to compute its match to a given phonemic target. They are :

1. A String of characters:
   (a) The name of the current phoneme.
   (b) The name of the left phoneme.
   (c) The name of the right phoneme.
   (d) The part-of-speech of the current word
   (e) The index of the current prosodic phrase
   (f) The number of words to the right in the sentence.
2. The index of the sentence containing the phoneme (related wav file names are given by this index)
3. The start sample for the current phoneme in the related wav file.
4. The end sample for the current phoneme in the related wav file.

3.4 Real-time Text to Speech

When a text input is given to the system, it goes through the following steps:

1. Preprocessing the sentence: dividing it into words and tokens.

2. Finding the most optimum part-of-speech tags: Once emission and transition probabilities are estimated for the HMM (section 2.2.3), obtaining the best sequence of tags for a given sentence reduces to selecting the best sequence of part-of-speech tags for the sentence, i.e., the one with highest probability (given the sequence of words and the bi-gram model). This corresponds to finding the best path in a lattice. We use the brute force algorithm to do this.

3. Breaking down the input sequence into chinks and chunks.

4. Phoneme Classification using the trained CART: Finally it implements a Viterbi algorithm for finding the best sequence of units, the one that minimizes an overall selection cost. The target cost is defined in a very crude way : it is 1 if the linguistic context features of unit and target match, and 0 otherwise. In other words, we have given the same weight to all linguistic features. The concatenation cost is simply set to 0 for consecutive units, and to 1 for others. No acoustic distance is computed.

5. Automatic Unit Selection [6]: Given a phoneme stream and a target prosody for utterance, our unit selection algorithm selects the optimum set of acoustic units from the phonetic corpus that best match the target specified (diphone). To do this, we optimise two cost functions : Overall target cost – defined as the sum of elementary target costs of the candidate units and the initial target, and the Concatenation cost – defined as the sum of elementary concatenative costs between successive candidate units.

6. Corpus Based Concatenative Synthesis [6] : is done based on information such as a phoneme, the previous phoneme and next phoneme, index of the part of speech and start and end sample . A simple concatenation is done which extracts selected diphones from the speech corpus and assembles them into synthetic speech. It is well known that this operation tends to produce audible mismatches .A minimalist

treatment of phase mismatches is implemented here, by computing the cross-correlation between units in order to shift them for least possible mismatch.

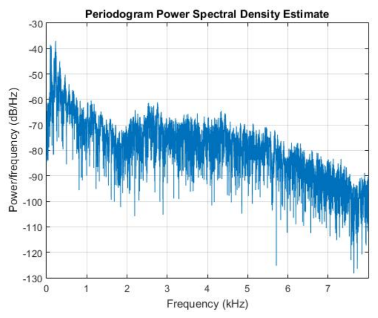
4 Results

The waveform generated using Hidden Markov Models is compared with a standard to observe how effective concatenation is in generation of speech. The standard that we have used for comparison is the state-of-the-art Microsoft Speech API5 which has a ready made database and functions for text to speech synthesis. The waveforms shown in figure 2 are compared with respect to their power spectral densities, spectrograms and time domain waveforms.

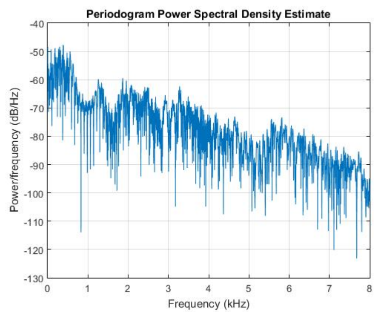
We can see that both the waveforms are comparable in formants and inherent structure. The differences that exist are due to the fact that both these speech samples are generated from different databases, with voices of different timbre and pitch. Thus it can be concluded that the waveform generated by our algorithm is comparable to that generated by the Microsoft Speech API and is quite intelligible.

5 Conclusion and Future Scope

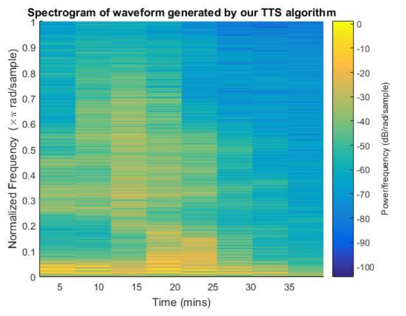
Text to speech synthesis is a rapidly growing aspect of computer technology and it plays an important role in various platforms of human computer interactions. The most debated trade-off considered while discussing TTS accuracy is that of intelligibility versus naturalness. Most text to speech techniques have reached the acceptable level of intelligibility. However there is still lot of work being done to achieve naturalness of speech. The Concatenative method provides more natural and individual sounding speech, but the quality with some consonants may vary considerably and the controlling of pitch and duration may be in some cases difficult, especially with longer units. Labelling of phonetic units is also quite difficult. With the advent of neural networks, advancements in this field are taking stride at a steady rate. Future scope for this study is to be able to handle numbers, acronyms, abbreviations, uppercase titles, how to handle out-of vocabulary words and spelling mistakes, while maintaining the naturalness of speech.



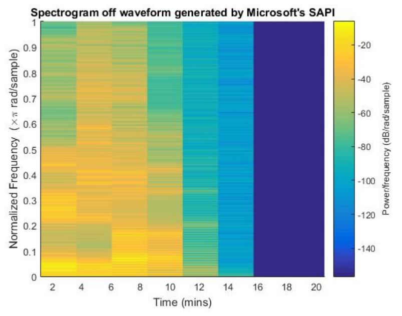
(a) Power Spectral Density of HMM based TTS



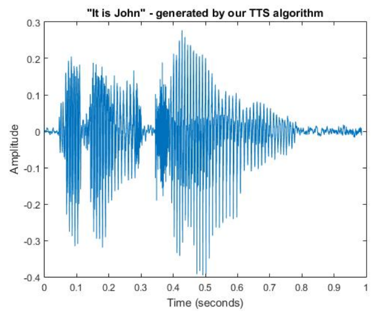
(b) Power Spectral Density of SAPI 5



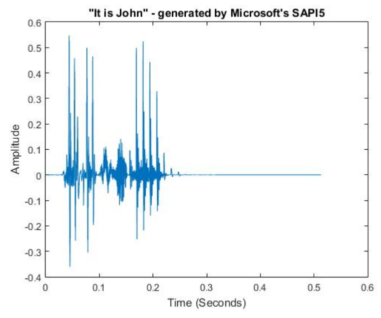
(c) Spectrogram of HMM based TTS



(d) Spectrogram of SAPI5



(e) Waveform generated by HMM base TTS



(f) Waveform generated by SAPI5

Figure 2: Analysis of Output

References

[1] Desai Siddhi, Jashin M., Verghese Desai Bhavik, ”Survey on Various Methods of Text to Speech Synthesis”, in *International Journal of Computer Applications* (0975 – 8887),2017

[2] T. Dutoit and M. Cernak, ”TTSBOX: a MATLAB toolbox for teaching text-to-speech synthesis,” Proceedings. (ICASSP ’05). *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005., Philadelphia, PA, 2005, pp. v/537-v/540 Vol. 5

[3] TTSBOX documentation published by the TCTS Lab , Faculté Polytechnique de Mons.

[4] F. MALFRERE, O. DEROO, T. DUTOIT, and C. RIS, “Phonetic Alignement: Speech-Synthesis-based versus Viterbi-based,” *Speech Communication*, vol. 40, no. 4, pp. 503–517, 2003.

[5] Paul Taylor, ”Text to Speech Synthesis”, University of Cambridge Publications.

[6] A. J. Hunt and A. W. Black, ”Unit selection in a concatenative speech synthesis system using a large speech database,” 1996 *IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, Atlanta, GA, USA, 1996, pp. 373-376 vol. 1.

[7] Yamagishi, Junichi. ”An introduction to HMM-based speech synthesis.” Technical Report (2006).