Development of TTS Engine for Indian Accent using Modified HMM Algorithm

S. S. Gantayat

Department of CSE, GMR Institute of Technology, Rajam, Andhra Pradesh, India
E-mail: sasankosekhar.g@gmrit.org

Abstract—A text-to-speech (TTS) system converts normal language text into speech. An intelligent text-to-speech program allows people with visual impairments or reading disabilities, to listen to written works on a home computer. Many computer operating systems and day to day software applications like Adobe Reader have included text-to-speech systems. This paper is presented to show that how HMM can be used as a tool to convert text to speech.

Keywords—K-means, Text-to-speech, Speech synthesis, HMM Algorithm.

I. INTRODUCTION

A text to speech system is composed of two parts, a front-end and a back-end. The front end has two major tasks. First, it converts the raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. This process is often called text normalization pre-processing or tokenization. The front-end then assigns phonetic transcriptions to each word and divides and marks the text into prosodic units, like phrases, clauses and sentences. The process of assigning phonetic transcriptions to words is called text-to-phoneme or grapheme-to-phoneme conversion. Phonetic transcriptions and prosody information together make up the symbolic linguistic representation that is output by the front-end. The back-end often referred to as synthesizer, it converts the symbolic linguistic representation into sound.

HMM-based synthesis is a synthesis method based on Hidden Markov Models, also called Statistical Parametric Synthesis, which is a widely used model for any Speech Recognition and Synthesis applications. In this system, the frequency spectrum (vocal tract), fundamental frequency (vocal source), and duration (prosody) of speech are modeled simultaneously by HMMs. In the proposed project work it is considered to improve the HMM technique to the Indian accents.

Statistical parametric speech synthesis based on hidden Markov models (HMMs) is now well-established and can generate natural-sounding synthetic speech. In this framework, we have pioneered the development of the HMM Speech Synthesis System, HTS (H Triple S). This research started by developing algorithms for generating a smooth parameter trajectory from HMMs. Next, to simultaneously model the excitation parameters of speech as well as the spectral parameters, the multispace probability distribution (MSD) HMM was developed. Using the logarithm of the fundamental frequency and its dynamic and acceleration features as the excitation parameters, the MSD-HMM enabled us to treat the sequence, which is a mixture of one-dimensional real numbers for voiced regions and symbol strings for unvoiced regions, in a probabilistic framework. To simultaneously model the duration parameters for the spectral and excitation components of the model, the MSD hidden semi-Markov model (MSD-HSMM) was developed. The HSMM is an HMM having explicit state duration distributions instead of transition probabilities, to directly model duration; it can generate more appropriate temporal structures for speech. These basic systems employed a mel-cepstral vocoder with simple pulse or noise excitation, resulting in synthetic speech with a “buzzy” quality. To reduce buzziness, mixed or multi-band excitation techniques have been integrated into the basic systems to replace the simple pulse or noise excitation and have been evaluated. These basic systems also had another significant problem: the trajectories generated from the HMMs were excessively smooth due to statistical processing, resulting in synthetic speech with a “muffled” quality. To alleviate this problem, a parameter generation algorithm that considers the global variance (GV) of a trajectory to be generated was developed.

II. MODULES

Module 1: Creation of Speech Corpus (dictionary).
Module 2: Labeling.
Module 3: Prototype HMM (Training of HMMs).
Module 4: Speech Waveform Synthesizer (Model Set).
A. Creation of Speech Corpus

CMU Arctic databases designed for the purpose of speech synthesis research. These single speaker speech databases have been carefully recorded under studio conditions and consist of nearly 1150 phonetically balanced English utterances. They are distributed as free software, without restriction on commercial or non-commercial use. The Arctic corpus consists of four primary sets of recordings (3 male, 1 female), plus several ancillary databases. Each database is distributed with automatically segmented phonetic labels. These extra files were derived using the standard voice building scripts of the Festvox system. In addition to phonetic labels, the databases provide complete support for the Festival Speech Synthesis System, including pre-built voices that may be used as is. Festival and Festvox are available at http://www.festvox.org. The Arctic speech corpus is available at http://www.festvox.org/cmu_arctic.

CMU Arctic is a set of single speaker databases that have been carefully recorded under studio conditions, packaged with associated information such as phonetic labels and pitchmark files. An Arctic “database” is a reading of the Arctic prompt set (plus associated files) by a single speaker in a specified style of delivery. This release of Arctic contains recordings by four separate speakers. When referring to the Arctic “corpus” we mean the entire collection of databases, including test sets. The databases have version numbers. As with computer code, version numbers indicate the level of maturity and stability. Numbers with a zero after the decimal point (e.g. version 1.0) are major releases intended to serve as a reference point for system development and evaluation. Minor releases are subject to change, allowing for more frequent additions, deletions, and improvements.

<table>
<thead>
<tr>
<th>Tomorrow</th>
<th>snow Probability</th>
<th>cloudy Probability</th>
<th>sunny Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>snow</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>today cloudy</td>
<td>0.2</td>
<td>0.6</td>
<td>0.2</td>
</tr>
<tr>
<td>sunny</td>
<td>0.1</td>
<td>0.1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

At every state tomorrow weather is depending upon today state Visual representation.

Each state corresponds to one observation, Sum of outgoing edge weights is one
Observables states, \( \{1, 2, \ldots, N\} \)

Observed sequence, \( \{q_1, q_2, \ldots, q_n\} \)
1st order Markov assumption
\[
P(q_n = j \mid q_{n-1} = i, q_{n-2} = k, \ldots) = P(q_n = j \mid q_{n-1} = i)
\]
We denote the time instants associated with the state changes as \( t=1,2,3,\ldots,n \) and actual state at time \( t \) as \( q_t \).

**B. Markov Model: Sequence Probability**

- Question: What is the probability that the weather for the next 7 days will be “sun-sun-rain-rain-cloudy-sun” when today is sunny?

  \( S: \) rain, \( S: \) cloudy, \( S: \) sunny

\[
P(O \mid \text{model}) = P(S_3,S_3,S_3,S_1,S_1,S_2,S_3) = P(S_3)P(S_3|S_3)P(S_3|S_3)P(S_1|S_3)P(S_1|S_1)P(S_3|S_1).
\]

\[
= 1 \times (0.8) \times (0.8) \times (0.1) \times (0.4) \times (0.3) \times (0.1) \times (0.2) = 1.536 \times 10^{-4}
\]

**C. HMM Algorithms**

There are three algorithms used in HMM

1. Viterbi Algorithm,
2. Baum-Welch Algorithm,

**1. Viterbi Algorithm**

- **Purpose** – An analysis for internal processing result. The best, the most likely state sequence for Internal segmentation
- **Alignment of observation and state transition**
- **Dynamic programming technique**

Key idea – Span a lattice of \( N \) states and \( T \) times, to keep the probability and the previous node of the most probable path coming to each state \( i \) at time \( t \)

**Introduction Phase:**

\[
\delta_0(i) = \pi_ib_i(x_1), \quad \psi_0(i) = 0
\]

**Recursion Phase:**

\[
\delta_{t+1}(j) = \max_{1 \leq i \leq N} \delta_t(i)a_{ij}b_j(x_{t+1}) \quad \psi_{t+1}(j) = \arg\max_{1 \leq i \leq N} \delta_t(i)a_{ij}
\]

**Termination Phase:**

\[
P^* = \max_{1 \leq i \leq N} \delta_T(i) \quad q^*_T = \arg\max_{1 \leq i \leq N} \delta_T(i)
\]

**Path backtracking Phase:**

\[
q^*_T = \psi_{T-1}(q^*_{T-1}), \quad t = T-1, \ldots, 1
\]

![Fig 4. The Viterbi Algorithm for Isolated Word Recognition](image)

**Fig 4. The Viterbi Algorithm for Isolated Word Recognition**

**2. Baum-Welch Algorithm**

1. The Baum-Welch algorithm can be used to train an HMM to model a set of sequence data.
2. The algorithm starts with an initial model and iteratively updates it until convergence.
3. The algorithm is guaranteed to converge to an HMM that locally maximizes the probability of the training data given the model.

**Baum-Welch Reestimation (EM)**

- Iterative procedures that locally maximizes
- Convergence proven
- MLE statistic estimation

The following table shows a one run of the Baum-Welch algorithm:

- ‘\( \alpha \)’ is considered as start and ending symbols of the sequence string and through multiple iterations the probability values will be converges.

![Fig 6 Example of Baum-Welch Algorithm](image)

**Fig 6 Example of Baum-Welch Algorithm**

**3. K-Means Clustering**

HMM kind of algorithm is well known to be sensitive to its initialization point. The problem of this initialization point choice is addressed in this paper: a model with a very large number of states which describe training sequences with accuracy is first constructed. The number of states is then reduced using a k-mean algorithm on the state. This algorithm is compared to other methods based on a k-mean algorithm on the data with numerical simulations.

The clustering algorithm partitions a dataset into a fixed number of clusters supplied by the user. Hidden Markov Model (HMM) based clustering method, which identifies a
suitable number of clusters in a given dataset without using prior knowledge about the number of clusters. Initially, the dataset is partitioned into windows of fixed size based on the HMM log likelihood values. This provides a framework for identifying the most appropriate number of clusters (windows of varying sizes). After determining the number of clusters, the data values are then labelled and allocated to clusters.

D. Features of Audio

The following features will be extracted from the audio file for training the HMM to get the speech.

1. Energy Entropy Standard Deviation (std)
2. Signal Energy Std by Mean(average) Ratio
3. Zero Crossing Rate Std
4. Spectral Rolloff Std
5. Spectral Centroid Std
6. Spectral Flux Std by Mean Ratio

III. CONCLUSION

In this paper it is shown how HMM can be used to generate speech from the text file. HMM uses dictionaries, labelling and the three different algorithms like K-means, Viterbi Algorithm and Baum-Welch Algorithm to train and after that the speech will be generated from the text files. The HMM can be used for image processing and forecasting also.

REFERENCES