ASHESI UNIVERSITY COLLEGE

TEXT TO SPEECH SYNTHESIS FOR GHANAIAN LOCAL LANGUAGES (Twi)

UNDERGRADUATE THESIS PROJECT

B.Sc. Computer Science

Derrick Odonkor

2016

ASHESI UNIVERSITY COLLEGE

TEXT TO SPEECH SYNTHESIS FOR GHANAIAN LOCAL LANGUAGES (Twi)

By

Derrick Odonkor

Thesis project submitted to the department Computer Science Ashesi University College

In partial fulfillment of Bachelor of Science degree in Computer Science

APRIL 2016

DECLARATION

I hereby declare that this thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……

Date: ……

I hereby declare that preparation and presentation of this [capstone type] were supervised in accordance with the guidelines on supervision of thesis laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date: ……

Acknowledgement

I appreciate the support from Professor Alan W. Black of Carnegie Mellon University and my supervisor Dr. Nathan Amanquah.

Abstract

Businesses in Ghana do communicate with each other through letters, telephones, fax, and emails. Due to the growth of technology, most of these businesses are providing their services online to make it convenient for customers. However, these mediums of communication have some limitations when it comes to language. In the instance where the parties involved in the communication speak and understand different languages, it becomes difficult to understand. Businesses would like to reach out to these people in a language they can understand. Statistics have shown that almost half of the Ghanaian Adult (15 and above) population are illiterates however they also need to do business as well to take care of their family. The challenge is that due to their inability to understand the message, either in text or audio, they tend to be left out in this age of technology growth. As a solution, this paper intends to develop a text-to-speech application system in Twi which can be used to develop prompts for Interactive Voice Response.

Table of Contents

Chapter 1

1.0 Introduction

The advancement in IT and the convenience of doing business online services has rapidly encouraged lots of people and businesses to learn and use IT in their day to day activities. Currently, people do interact with businesses through phone calls, web customer care, letters, fax, and in person. There are some challenges involved in these mediums of communication mentioned. The common ones occur when parties seeking to interact or transact business speak different languages, and will have to translate texts written in different languages. Futher, some people cannot read texts written in their language, so they usually hire readers to do that for them. This becomes a problem for companies whose target market struggle to read the information on the bill boards, flyers, and even struggle to understand the announcement or advertisement on the radio. To improve on this limitation and reach out to more people especially for the businesses, an extension will be to use interactive voice response (IVR) systems in their automated systems such as ATMs, call prompts, online instructor, voice prompt to check for bank balance, and voice prompt installed in cars to give directions. Though some of these IVR systems have been implemented one of the limitations are that the voice prompts are pre-recorded and fixed for one particular system. For instance, in the Telecom systems, the voice prompt that alerts the caller to record a message after a tone is pre-recorded and installed into the Telecom system. Also the voice that provides details of a user's bank account, and the voice that shows appreciation whenever a transaction is made at the ATM are all examples of pre-record statements. Such systems cannot be directly applied to other systems. However, new set of voice prompts have to be created for those

systems. This makes the process of creating the prompt quite expensive due to the fact that the actor, whose voice will be used for the recording, will have to be paid for the amount of time spent. Another database to store these wave sounds will be required, and if it has to be done for different languages the actors of various languages will have to be employed hence paid.

A large portion of the Ghanaian population do not read English hence does not feel comfortable using any form of IT for business, and education. Others sometimes prefer to listen to the local radio presentations. This because they feel more comfortable with it than that of one with a different language since they lack some form of education. In Ghana, the Youth (15-24years) literacy rate is 71% and the Adult (15 and over) literacy rate is 58% on average ("Effective Literacy Programmes L", 2016). This statistic shows that there are about 42% of adults who are left out when it come to IT advancement. For instance, these 42% illiterates, made up of both males and females, enjoy the services of Ghana Commercial Bank as compared to Ecobank and other banks such as Ecobank, Zenith bank, and Cal Bank because there is staff available at GCB branches to assist them with whatever they want, be it filling a form or cheque. Other banks have self service centers and processes with text in English making it difficult for these illiterates to catch up. Because of this large number of illiterates in Ghana, it will be best for businesses to reach out to them and provide their services in their own language. Further, businesses can employ IT to enable them serve more customers more efficiently and conveniently. However, as businesses automate, this is going to pose as a challenge that has to be solved. The 42% illiterates would find it convenient to interact in a local language by

voice (in fact lot of people do interact in the local language whenever they turn up in an office, event, or in meeting with peers). Such people can benefit from a local language Interactive voice response (IVR) systems. To increase flexibility and easy adaptation to these systems, it will be important to make use of text-to-speech (TTS) to reduce the cost of deployment and increase the likely hood of adoption. Using the Ghana Commercial Bank scenario, the bank can integrate an Interactive Voice Response in their system to help with some of the processes. For example GCB can use it in an automated machine to give basic instructions on how to fill a cheque or withdrawal form.

TTS is a system that converts a given text to a speech sound. The system may take the text from a database. The text goes through different phases to be processed then gives the speech sound as an output. TTS systems can aid persons who are visually impaired, mul-titaskers, readers of a large amount of text, and can be used in automated systems such as ATMs and any system that uses voice prompts.

1.1 Background

Over the years language has been one of the means of communication among people. Be it spoken or unspoken, language remains an essential tool used in sending information from one person to the other. Various languages such as English, French, and Italian contain different sounds and their articulation can change the meaning of the word even if spelt identically. This occurs if the right pronunciation is not achieved. That is, if a different accent (Example, British accent) pronounces a word of an unfamiliar language (Twi), the inaccurate pronunciation and intonation could be interpreted inaccurately by the hearers or audience. Example is that, a foreign who has not heard of Twi before will

pronounce the word “Nyame” differently from a native Ghanaian. This is because the “Ny” combination does not occur in English. Narrowing this problem down to the Ghanaian community, businesses are growing and expanding their medium through which they reach out to customers. Since these messages are in particular languages, it is important to carefully pronounce these words with care so as to convey the right message. A mis-communication of information due to these important components of speech such as the phonetic sounds can lead to the wrong interpretation of the message received.

1.2 Significance of the Problem

This is a significant problem in the Ghanaian business society where people interact and transact services online, business systems often need to support multi local languages. This is because clients are getting convenience and comfortable with online services, however some people still get left out due to their inability to communicate in these foreign languages. Twi IVR systems will be useful because it is the most dominant language spoken in Ghana. Currently the Twi systems available are fixed format, pre-recorded content and cannot be transferred to another system if need be. This means that a new recording has to be made for a new IVR system. This will result in high cost and delay.

Furthermore, foreign visitors find it difficult to read local Ghanaian texts and due to their accent and the tonation of the words which they are not used to, meaning of the local language words might turn out to be different. This is because either the language sounds new to them or they might have learnt the local language by reading text and may enunciate with their foreign accent hence making it difficult to understand. Though the

Twi IVR system may sound not so natural it gives the visitors some basic idea of how words are pronounced.

Solutions such as text to speech systems and speech to speech systems have being developed to solve some of these problems such as difficulty in reading text on smaller screens. People who access information on their mobile can make use of this system and this is because reading on the small screen often strains the eye. Some benefits of text-to-speech systems are that they can be used in automated systems such as ATMs, customer care services, and text readers. They also can be used by people with learning disabilities due to dyslexia and other learning problems. Furthermore, people having issues reading large amount of text can have some sort of system that can read to them and the vision impaired and the mul-titaskers can also benefit from such systems.

The challenge of using a IVR as a solution in the Ghanaian local language societies is that current text to speech systems are developed with foreign accents and are designed for foreign languages. However, using a foreign accented text to speech engine for a local text does not sound natural to be of good help. Also the Ghanaian local language (Twi) has certain characters whose speech sounds are difficult to enunciate using to the same process used in creating a foreign language TTS. To the best of the author's knowledge, no Twi TTS exists. However the Twi keyboard and texts exists (Fletcher, 2016). Google has a Twi search engine (Google, 2016) but no Twi TTS yet.

The aim of this paper is to identify a way of developing a Ghanaian local language text-to-speech. Using the festival tool on the Linux platform and the PRAAT record and

labelling tool on the windows platform, this paper seeks to answer the following questions at the end of the research.

- What steps need to be completed to create a TTS for Twi?
- Would the quality of TTS for a Ghanaian local be good enough to be understood by a local language speaker?
- How does the output of TTS for Ghanaian language compare with using an English TTS engine on Ghanaian language text?

1.3 Overview of remaining chapters

In the subsequent chapters, chapter 2 outlines the details with the related work, various types of the TTS system, the technologies, tools, and processes involved in developing a TTS system, and an explanation of how it works. Chapter 3 details out the methodology and the necessary steps taken to achieve the research questions stated above.

Chapter 2 – Related Work

2.0 Background

A Text-to-speech synthesizer is a computer-based system that should be able to read any text aloud (Acero, 2000). It is a tool used to convert text to an acoustic sound. Investigations has been made into how text to speech systems can be used in an automated telephone banking system to offer additional details on customers' account transactions (Morton et al. 341-362). The results showed that customers welcome the added value of Text-to-Speech to the system but that it has to be minimized (Morton et al. 341-362). TTS is also a useful tool in the educational sector used to enhance the teaching capabilities (Rughooputh et al, 2008). This is because teachers use them in their teaching to help students understand the lecture. These benefits go a long way to help people finding it difficult to learn.

2.1 Types of TTS system

TTS can be accomplished in several ways. The methods include articulatory synthesis, formant synthesis, and concatenative synthesis (Acero, 2000).

2.1.1 Articulatory Synthesis

In the articulatory synthesis, the quality of the speech sound is high due to the direct modeling of the human articulatory behavior (Acero, 2000) namely how to model the positions of the tongues and jaws which help in speech production. Here the human vocal tract is modeled to produce sounds like that of humans. That means the positions of the speech articulators are adjusted to produce the speech sound. The down side of it is that it is difficult to model the human articulatory (Acero, 2000). By far it is the most

complicated with regard to the model structure and computational structure ("Speech Synthesis", 2016). Therefore, there has not been any successful implementation using the Articulatory Synthesis as compared to the other types of synthesis but in theory, it has the best high-quality synthetic output. It involves lots of computations and adjustments of the position the lips and the tongue, lung pressure, the tension of the vocal cords and so on ("Speech Synthesis", 2016). So currently everything is in theory, yet to have an implementation of it.

2.1.2 Formant synthesis

The formant synthesis is an intelligent way of producing speech but the output sounds robotic (Acero, 2000). It is basically an acoustic resonance of the human voice tract that are fixed frequency peaks, the frequency (period/wavelength) of waves represented by a peak (maximum energy) in the wave spectrum. This is sometimes known as the (dominant frequency) (Bowles, 2016), that makes a timbre of a voice consistent over a wide range of frequencies. This method does not sound natural enough to be used in systems such as the search engines, and ATMs. Computers are usually used to make these sounds.

2.1.3 Concatenative synthesis

Thirdly, the concatenative synthesis is basically joining various previously saved wave sounds to form a speech sound (recorded voice prompts that are saved into any storage space with a .wav extension). Different rules are used in this method and one of them is the syllable-rules where a text is divided into syllables and then based on the syllables their corresponding wave forms are sounded and joined together. The disadvantage of this method is that the size of database used to store the various wave-

forms of the corresponding phonemes grows larger. Phonemes are the basic form of language. According to (Karthikadevi), the production of the TTS for the Tamil language, using the concatenative method was the best as compared to the others. This is because the wave forms of the basic forms speech such as phonemes are recorded and combined to produce a speech sound. (Shreekanth et al), in building the TTS for Hindi, used the concatenative method in conjunction with the syllable-rules since the Hindi language is naturally syllabic. The results provided a smooth continuity in the flow of the speech sound. They conclude similarly that concatenation is the best among the various methods mention above.

Just like the Hindi language, the Twi language contains some unusual combination of alphabets such as “tw” in “twen” (meaning to “wait”) which is difficult to process by an English language TTS. This is because “tw” combinations do not exist in the normal English phonemes combinations. Others combinations include “ky”, “ny”, “kw”, “dw” such as contains in “kyere”, “nyan”, “kwan”, “dwen” respectively. Nevertheless, this paper is going apply the syllable rule based and the concatenative method in the system production because the articulatory synthesis has not been successful in practice but seems to work in theory and simulation ("Speech Synthesis", 2016) and formant synthesis sounds too acoustic to be used in a TTS.

2.2 Analysis of a Word

To accomplish TTS sentences must be broken into words and words analyzed by their constituent components.

- **Speech sound** is the smallest discrete segment of sound in a stream of speech.

- **Phones** are basically the vowels or consonants of the alphabets. They are the basic unit of phonetic analysis and a speech segments that posses distinct physical and perceptual properties. To illustrate, given the phrase in figure 1, the individual letters are phones. This because they individually posse distinct sounds that will not change the meaning of any word.

- **Diphones** are basically a pair of phones adjacent to each other in a verbal sequence. They are usually the transition between two phones. With reference to Figure 1, "Th" in the word "This" is a diphone. In speech synthesis, when pre-recorded diphones (Th in this case) are combined together they sound more natural than combining just pre-recorded phones 'T' and 'h'. This is due to the variations of phones depending on its surrounding phones.

This is my skin

Fig 1. Example of a sentence

- **Syllables** are units of spoken language consisting of single uninterrupted sound. They influence how language is spoken in terms of the rhythm, prosody, and stress. They are also known as the building blocks of words. It is subdivided into onset, rhymes, coda, and nucleus.

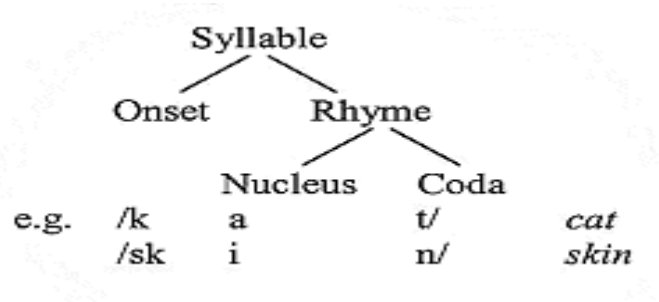

Fig. 2. Illustration of the various components of a Syllable

- **Onset**-The beginning sound of the syllable usually consonants in English. In fig. 1 above, "/k" is the beginning sound of the word "cat".

- **Rhymes**-This is the rest of the syllables after the onset. Inferring from fig.1, "a" and "/t" are the rhymes of the word "cat".

- **Nucleus** - This is the core or essential part of a syllable. Without it a syllable cannot exist. From fig. 1 "a" is the nucleus of the word "cat"

- **Coda**-This is the ending consonant sound of the syllable. Finally, "/t" is the coda of the word cat.

Therefore nucleus together with the coda forms the rhyme of a rhyme of a syllable.

- **Words** are combination of syllables. From the example above the word "skin" consists of "/sk", "i", and "n/". Combining these gives you the word "skin"

- **Phoneme** is a single unit of sound that has meaning in any language. Thus, its sound distinguishes a word from another. The difference in meaning of the words "skin" and "skit" is as a result of exchanging the phoneme /n/ and the phoneme /t/. Another difference in meaning of the words "sky" and "dry" is as a result of exchanging the phoneme /sk/ and the phoneme /dr/.

2.3 Processing Text in A TTS synthesis system

Developing a TTS involves a lot of time and voice processing. Currently a lot of research work is being done in this area to improve on the system to be able to generate natural sound from the system and also with emotions. The process involved in obtaining a speech sound from a TTS is illustrated in Fig. 2.

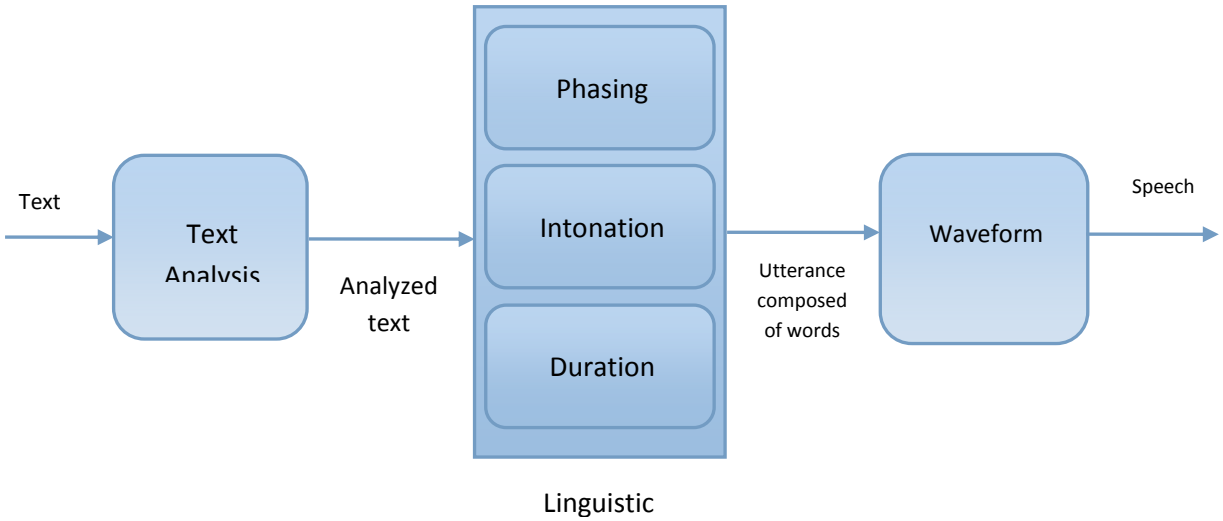

Fig. 2. Processing text in a TTS system

First of all, text to be converted to speech goes into the **analysis stage** where the system has to identify what kind of text has been entered by the user. Supposing the text provided is "I was born in 1992". The system at the analysis stage needs to figure out whether the user meant "I was born in nineteen ninety two" or "one nine nine two" or "one thousand nine hundred and ninety two" based on the context. Consequently, a sentence such as "I need $1992" will be output difference from "I was born in 1992" although they both have the same number included. After the system identifies which it is, it is then passed on the **Linguistics analysis**. Here, the text goes through the **phasing**

stage where all texts that are not in text form are converted to text such as converting the numerical value 1992 to a text form depending on the linguistic analysis obtained. The difference between the two stages is that the analysis stage identifies the type of text based on context and the linguistics converts the text to a readable text.

Next, the converted text is parsed to the **intonation stage** where one of the methods (Articulaory, Formant, or Concatenative synthesis) for speech production is used. For instance the concatenative synthesis may be used to break down the text into their various phonemes and then these phonemes are matched to their corresponding wave sounds that have been previously recorded and stored in a database. This is done using an efficient algorithm that matches the phonemes to the recorded sound. When this is done the text is then refined in the **duration stage** where based on the text's position in a sentence, phrase or a standalone word, the utterance is produced and a waveform is generated as an output to the user using the speakers connected. Fig. 2 shows a graphical representation of the process.

2.4 Phoneme and Speech Generation Creation

To be able to build the system in section 2.2, the following processes have to be followed;

a) **Recording**

This is where a lot of words that contains the various phonemes, diphones, and syllables are recorded. This is done using a microphone and recording software or any recording device that can filter as much noise possible. The purpose of recording these words is to be able to extract the phonemes from them. A phoneme at the beginning of a

word will sound different from that same phoneme in the middle or at the end of a word. So recording these words will provide the variations needed for the system. Table 1. Below shows a few English phones which appear in the diphones.

Table 1. Sample phonemes and position in words

Phones	**Words**
aa	fAther, wAshington
ae	fAt, bAd
ah	bUt, hUsh
ao	lAWn, dOOr, mAll
aw	hOW, sOUth, brOWser

Through this stage, the voices that are required to produce sound output are extracted. Different versions are recorded to gain the stress, pitch, and other details of a particular phoneme. This is done by recording words that contain these phonemes. Recording these phonemes and others can take hours or even days to do because of the number of possible combinations to be recorded the actor would have to take rest breaks. If there are some mistakes during the recording, it has to be re-recorded.

b) **Labeling**

After the recording is done, labels added to the .wav sound generated so that the system can use it as reference when needed. That is, after the sound is recorded it has to be stored with a particular name that relates to it so that it becomes easy to identify that particular sound when needed. There are no specified rules during the labeling process. Any naming convention the developer wants to use is allowed so long as duplicates are not created in the process.

c) **Matching rule**

This is where the developer writes rules to reference the text entered by the user to the wave sound corresponding to the constituent phonemes and diphones. The rule is a matching pattern that helps to refer to the wave sounds when requested. This is important in the concatenation phase of the speech development because it is one of the methods that merge different sound wave forms to generate a speech sound. Example is

```
(# [c h] C = k)
```

The above means that for any word beginning with “ch” followed by a consonant, replace the “ch” with “k”

d) Concatenation synthesis

In this stage, the final wave sound is generated by concatenating all the phonemes, and diphones needed for the word pronunciation. These are concatenated together to form the final speech sound.

e) **Speech production**

Finally the speech is uttered for the user through an audio output system such as speakers.

This paper intends to use the concatenative synthesis as a method to develop and create the speech sounds. This is because it will be useful and much efficient to concatenate the wave files and generate the intended sound though the database may grow large.

Chapter 3 Methodology

3.0 [] of Steps

The purpose of this paper is to build a text to speech engine for a local Ghanaian language to read out Twi words supplied by a user. To test it, Twi phrases will be passed to the system for the system to read out the phrase to the user. Using festival (Black & Lenzo, 2016), a speech synthesizer created at Carnegie Mellon University, to create the synthesized voice will be helpful because it contains functions to cater for the speech analysis and linguistic stage. Also the paper will be exploring the PRAAT tool for recording the various phonemes in Twi and also use it to label them accordingly.

The recording will be done using the windows version of the PRAAT application. This is because the Linux version of PRAAT has compatibility issues with the current hardware of the computer being used. Since festival is programmed for Linux distribution, this project will be using a Linux distribution (Elementary OS) for the deployment of the festival application.

Sample phonemes will be recorded for just a few words about 30-40 words to verify the workings of the system. This will take some time due to the synthesizing process which can take some time because it is processor intensive and is dependent on the PC processor speed and memory. Next, the necessary changes and tweaks to the system will be made till the desired output is not achieved. Such as re-labelling some waves which failed.

Table 2. PC and Virtual Box specifications

PC Specification	
OS	Windows 7 Professional
Version	6.1, Build 7601
Board or PC & model number	Lenovo T420
Processor	Intel(R) Core(TM) i5-2430M CPU @ 2.40GHz (4 CPUs)
Memory	3.9 GiB
System Model	4177R3U
Storage	54.1 GiB
Virtual box Specification	
OS	Elementary OS
Version	0.3.2 " Freya " (32-bit)
Built-on	Ubuntu 14.04 ("LTS")
Virtual Box Version	5.0.2 r102096
Processor	1 CPU
Memory	1024MB
Display	128 MB video memory
Storage	15GB Virtual size and 8.62 Actual size

3.1 Finding the Pronunciation of a Word

For this project a lexicon which contains all possible words, is created in a syllabic structure. This is the first point of reference for every text received. In festival, the lexicon takes both the word and a part of speech to find its pronunciation (Black & Lenzo, 2016). This is done to make the process of finding the pronunciation of the word efficient. This occurs in the **phasing stage.** In this file, every line contains some words in a particular format conforming to that of festival. Thus, every line contains a word, the

part of speech of that word (thus, whether a noun, a verb, an adjective, or an adverb), and how that word is broken down which is the pronunciation of that word. Below are some examples;

("walkers" n (((w oo) 1) ((k @ z) 0)))

(“present" v (((p r e) 0) ((z @ n t) 1)))

As part of the pronunciation, the stress is defined as well. In festival, this is denoted by using “1” or “0” to indicate whether high or low tones (Black & Lenzo, 2016). This approach can be changed to suit the project’s structure. After a lexicon is defined, the letter-to-sound rule is developed as a supplement to the lexicon. So that, should the text entered by the user not exist, the system with then use these set of rules to generate the speech sound. The letter-to-sound rule is basically a set of rules that can be used to match letters or phonemes in a word to their corresponding wave files. This is done by hand normally and saved in a text file. An example of the format for creating a letter-to-sound used in festival is shown in Fig. 3.

```
(LEFTCONTEXT [ITEMS] RIGHTCONTEXT = NEWITEMS)
(# [c h] C = k)
```

Fig. 3 Sample Matching-Rule

This means that any ‘ch’ at the start of every word followed by any consonant should be rendered as the k phoneme (Black & Lenzo, 2016). The letter-to-sounds can be built from scratch using the procedure provided in Listing 1. This can be found in chapter 13.5 of the festival manual documentation (Black & Lenzo, 2016). This can be done for Twi as

well by replacing the non-existence combination with the corresponding letter that sound similar in English. Example

```
(# [k w] v = q)
```

The above means that if a word begins with “kw” followed by a vowel, the “kw” should be replaced with “q” as in “Kweku” and “Kwadjo”.

- Pre-processing lexicon into suitable training set
- Defining the set of allowable pairing of letters to phones.
- Constructing the probabilities of each letter/phone pair.
- Aligning letters to an equal set of phones/_epsilons_.
- Extracting the phonemes suitable for training.
- Building a Classification And Regression Trees (CART) models for predicting phone from letters (and context).
- Building additional lexical stress assignment model (if necessary).

List 1. Letter-to-sound procedure (Black & Lenzo, 2016).

All except the first two stages are automated (Black & Lenzo, 2016). After building the letter-to-sounds rules, all possible phonemes are recorded and saved to a database.

When the user enters a text, the system first checks if the word exists in the lexicon that was initially created. If it does, the system picks the pronunciation part and then, using some functions, it locates the corresponding wave sound from the database. To illustrate, suppose the lexicon has the entries in Listing 2 and the user enters the word “present”. The system first goes through the lexicon file to lookup the word “present”.

After finding it, system selects the pronunciation part of it. In this example the system will select (((p r e) 0) ((z @ n t) 1)). After that the system, using an in-built function, it matches the individual pronunciation to the corresponding wave files by selecting them from the database that was created after the recording.

```
("walkers" n (((w oo) 1) ((k @ z) 0)))

("present" v (((p r e) 0) ((z @ n t) 1)))

("monument" n (((m o) 1) ((n y u) 0) ((m @ n t) 0)))
```

List 2: Sample lexicon file

After getting the individual pronunciations, the system concatenates the individual sounds to form the complete speech sound.

In the case where the word does not exist in the lexicon, the system tries to generate the sound by using any letter-to-sound rule defined in the system. Letter-to-sound is used to support the lexicon when the word cannot be found. Usually, this generates something very close to what is needed.

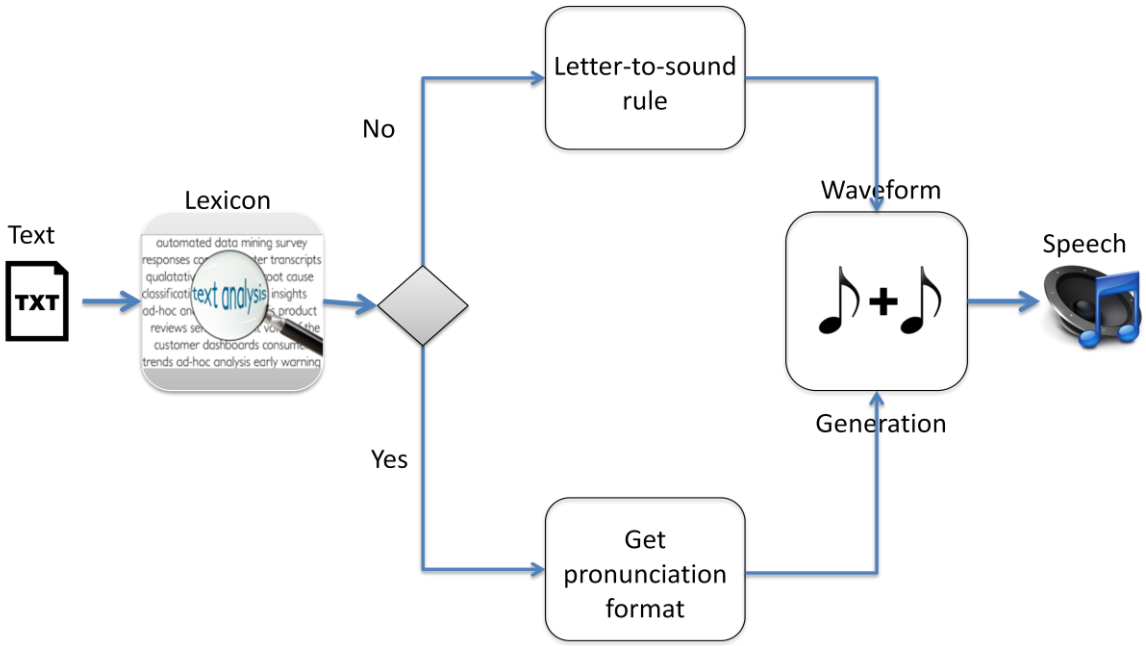

Fig. 3 Graphical Representation of Finding a Word's Pronunciation

Chapter 4 Implementation

This chapter talks about how the TTS was implemented. It describes the various processes needed to accomplish the intended goal.

4.0 Installation Process

The process of installing the necessary tools needed to be able to build a voice is described below.

4.0.1 Downloading Edinburgh Speech Tools, Festival, and Festvox

For Festival to work, it requires the speech tools. In order, Speech Tools, Festival and Festvox each need to be installed separately. The files came as compressed **.gz** files and were extracted to a folder before the rest of the actions were carried out.

The following was the actions taken to prepare the tools. It assumes the user is their home directory.

a. Created a directory for the downloaded files. Used the following command to create the directory in the home directory.

```
mkdir  -p -m 0700 ~/installed/festival
```

b. Created a directory for the installation

```
sudo mkdir -p /usr/local/festival
```
hit enter

Then entered the root/admin password.

c. Downloaded the latest Speech Tools, Festival, and Festvox from the link below and moved them into the directory ~/installed/festival

http://festvox.org/packed/festival/

d. Changed into the *~/installed/festival* directory and extracted all the files into the `/usr/local/festival` directory. Below is a one example. Follow for the rest of .gz files

`sudo tar xzvf ./festlex_POSLEX.tar.gz -C /usr/local/festival`

Entered the root/admin password

e. After extracting the files the directory looked like this;

`/usr/local/festival/festival`

`/usr/local/festival/speech_tools`

4.0.2 Installing downloaded tools

Before the tools are installed onto the pc make sure the files were successfully extracted. If there were errors it might be an incomplete download or the extraction was interrupted at some point. To resolve these issues, re-download and extract the files.

If there were no errors then one can continue with the following instructions.

i) First became root and changed into the **/usr/local/festival/speech_tools** directory using the commands below.

sudo -s hit enter

Entered the root/admin password

To change to the directory, do

`cd /usr/local/festival/speech_tools` hit enter

ii) Ran the following command which created the configuration file since it did not exit

`./configure`

One needs to make sure that gcc and g++ compliers are installed. One can check by trying to compile a simple hello world cplusplus (.cpp) code. If the system

complains about the lack of existence on a gcc complier or any C language complier, then it has to be installed. Alternatively, simply type gcc or g++ and observe the system response.

Next, ran

```
make
```

which compiled the speech tools.

At the time of installation an error about “lcurses” and “lncurses” was encountered. This was resolved by “libncurses5-de” library by running the following command in terminal.

“`sudo apt-get install libncurses5-dev`”

After that the ‘make’ command was ran again to make sure it works.

Finally for the speech_tools the command below was ran again;

```
make test
```

This will print the results; ‘Test ok’. If anything else is printed then go back and make sure no step was missed.

iii) If the compilation is successful, change the directory to the festival directory, thus

```
cd /usr/local/festival/festival
```

iv) Run

```
./configure
```

Then;

```
make
```

v) If the POSLEX, CMU, and OALD lexicons, and the don, kedlpc16k, and rablpc16k voices, were installed one can run 'make test' now

Make test might complain about some other voice however one can go to the festival url and download the required voices or can continue anyway.

vi) Created a symlink so that festival can run without its path. The following code was used to create the symlink.

```
'ln –s /usr/local/festival/festival/bin/festival
                    /usr/local/bin/festival'
```

vii) Now one have to make sure festival is installed and configured properly.

Run '`festival`' in terminal

If it gives an error then there was a mistake in your symlink command. One will have to check that.

viii) If it works one should see something like figure 3 on the terminal.

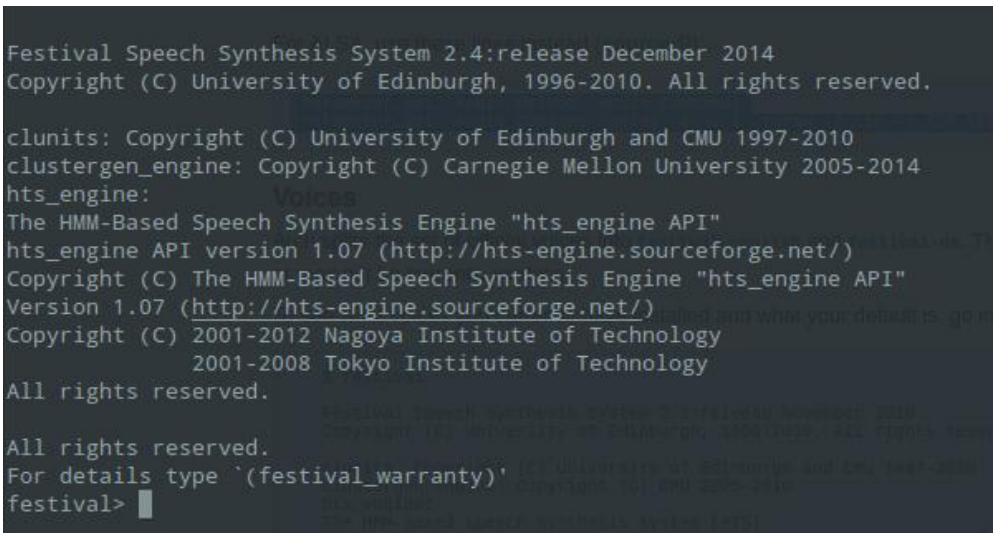

Fig. 3. Festival running console

viii) To test the program, a sample text was passed to it.

```
(SayText "Hello")
```

There was no sound and the system complained of 'Linux: can't open /dev/dsp,. This was resolved by running 'locate siteinit.scm' in terminal to locate the file 'siteinit.scm'. If the file does not exist, one should make sure the other voices and downloaded and extracted then restart the process again from point (iii). If the file exists navigate to that directory one sees. Run the command below to edit the file.

```
gedit siteinit.scm
```

The following was added to the bottom of the file.

```
(Parameter.set 'Audio_Method'Audio_Command)
(Parameter.set 'Audio_Command "aplay -q -c 1 -t raw -f s16 -r $SR
$FILE")
```

After that, save and then restart from point (vii).

Make sure that all this is done as root else one will get a permission denial error. This simply means one do not have the permission to make any form of changes to the file.

Next was to create the TTS using the steps defined in the previous chapter. First the procedure was tested with the English language then with the Twi language.

4.1 Creating TTS for English language

Based on the tutorial on Grapheme-based Synthesizer (Black and Lenzo), the environment variables are set to where the Edinburgh Speech Tools, the FestVox distribution and NITECH's SPTK were installed.

```
export ESTDIR=/home/awb/projects/speech_tools
export FESTVOXDIR=/home/awb/projects/festvox
export SPTKDIR=/home/awb/projects/SPTK
```

Using the English language as an example, the working directory for the project was created.

```
mkdir ~/data
mkdir ~/data/ash_en_dee
cd ~/data/ash_en_dee
```

The naming convention used were, the institution, language and speaker.

After, the following command was setup to setup the directory that was created.

```
$FESTVOXDIR/src/clustergen/setup_cg ash en dee
```

Next was to prepare the utterances and record them. To accomplish this task, a text file containing a large amount of text was used. The text file contained 1,100,849 words, 2,350 pages. The word frequencies of all the words in the text file was first found. The command to accomplish this is:

```
$FESTVOXDIR/src/promptselect/make_nice_prompts find_freq TEXT1.txt
TEXT2.txt ..
```

The text files maybe pathnames.

Next stage is to build a Festival lexicon for the most frequent words. By default the number of words selected for the lexicon is 5000. This can be overridden by adding an argument at the call of the command.

```
$FESTVOXDIR/src/promptselect/make_nice_prompts make_freq_lex
```

OR

```
$FESTVOXDIR/src/promptselect/make_nice_prompts make_freq_lex 500
```

The 500 at the end of the command tells the festival system to only build a lexicon of 500 words instead of 5000. An error occurred complaining about an “awk” complier. Gawk was installed to resolve the issue. The command below was used in terminal

```
sudo apt-get install gawk
```

When this is done, run the frequence lexicon command again.

After the lexicon is generated, sentences in the text file are processed to identify “nice” utterances. The definition of nice according to Festvox (Black and Lenzo), is a reasonable length of a sentence which has only words in the frequency lexicon, no strange punctuation, capitals at the beginning, and punctuation at the end, and a few other heuristic rule conditions. These “nice” utterances are written to data_nice.data file. The following command was used to generate the “nice” utterances.

```
$FESTVOXDIR/src/promptselect/make_nice_prompts find_nice TEXT0 TEXT1 ...
```

This will only search for the first 100,000 utterances (Black and Lenzo), which can be changed if, need be. This experiment maintained the default. 2029 nice sentences were generated.

The next stage is to identify those nice utterances that have the best coverage (Black and Lenzo) since at this point no lexicon and phonetics are available to select from. The algorithm used here is a greed algorithm where is it takes in a number to define the number of utterances it is looking for. The algorithm is then applied multiple times till the defined number is reached (Black and Lenzo). The default is 1000 utterances. Since this experiment is based on a CLUSTERGEN voice, 500 utterances are enough.

```
$FESTVOXDIR/src/promptselect/make_nice_prompts select_letter_n
```

This generated the 1081 nice prompts into `data.done.data`.

Finally, this stage extracts the vocabularies of the selected prompt set (Black and Lenzo). This will be needed when building the pronunciation lexicon and the speech database to contain the recorded wave files.

```
$FESTVOXDIR/src/promptselect/make_nice_prompts find_vocab
```

This whole process can be done using the command below;

```
$FESTVOXDIR/src/promptselect/make_nice_prompts do_all TEXT0 TEXT1 ...
```

4.2 Creating TTS for Twi language

In creating the TTS for the Twi language the same process was followed. A few changes were made to the working directory. This time the directory looked like this:

```
mkdir ~/data
```

```
mkdir ~/data/ash_tw_dee
cd ~/data/ash_tw_dee
```

The rest of the commands were followed till the very end just like the TTS for English. The experiment encountered an issue when finding the word frequency. The problem was that the system could not identify the ɔ and ɛ twi characters. The system recognized them as empty strings and so anywhere it finds those characters it takes the first part of the word before the characters in question and leaves the rest as invalid. Example; "wɔbɛbara" will end up to be "w" only discarding "ɔbɛbara" due to the first "ɔ" in the word.

There was a work around to resolve this by replacing all "ɔ" and "ɛ" with "c" and "z" characters respectively. This was done because the "c" and "z" characters do not exist in the Twi alphabets. The process of doing these replacements was termed as the Minion Algorithm.

After the algorithm was applied the rest of the commands were followed to the same point as the English TTS.

4.3 Limitations

Next stage is to record the prompts, however the experiment halted at this point due the following limitations.

1. Poor documentation

   The documentation on the festival applications was difficult to understand. There is no much work done with the festival system so finding information about was difficult.

2. Lack of phonetic materials on the Twi language

3. The festival system could not identify the ɔ and ɛ alphabets

Chapter 5 Tests and Results

Using a text file with 1,100,849 words, 2029 nice sentences were generated. Out of which 1081 nice prompts were generated and 2615 vocabularies. For the Twi, before the Minion Algorithm was applied to the text file with 14019 words, 700 frequent words, only 2 nice sentences, and 10 vocabularies were found. The 700 frequent words generated were not actually words but some combinations of letters which was not what was desired. After the algorithm was applied to the same text file 14019 words, 384 frequent words, only 384 nice sentences, and 1089 vocabularies were found. This proves that the algorithm that was applied works.

Also a few sentences were recorded for the English TTS using the PRAAT tool. However a few phonemes were extracted but could not continue the project due to the amount of time needed to record, label and extract sound waves.

5.1 Future works

To complete these tasks the following will have to be completed.

- Record and label the utterances
- Build utterance structures for recorded utterances
- Extract F0, voicing and mcep coefficients.
- Build a CLUSTERGEN voice
- Build an HMM-state duration model
- Testing

After this is done then the next stage is to follow the same principle to build a Ghanaian local language version, thus Twi.

Chapter 6 Conclusion and Recommendation

The objective of this paper was to understand the necessary steps needed to create a Ghanaian version of the TTS. Another aim was to compare the quality of the Ghanaian TTS to that of the English version. This is to prove whether the Ghanaian TTS sound the same as the English version when a Twi text is passed to it.

Though the experiment was halted due to lack of time and other hindrances such as the limitations of the festival application, the basic understanding of how to create a voice was achieved. That is, the experiment will prove successful if completed according to the layout structure. Currently, for future works, the "nice" sentences that was extracted from the text file need to be recorded and the various phonemes also need to extracted from the recorded voice. After which the rest of the processes will be handled by the festival application by passing the necessary parameters to the functions.

This system will be very useful to the Ghanaian society. As the Text-to-Speech field keeps growing, there will be many ways to integrate it into the Ghanaian system so that it will benefit everyone. Businesses will also server majority of the population as they bridge the gap between the literate and the illiterate.

Reference

Acero, A. (2000). An overview of text-to-speech synthesis. *Speech Coding, 2000. Proceedings. 2000 IEEE Workshop on*, pp.1-. doi:10.1109/SCFT.2000.878372

Arun, K., Shreekanth, T., & Udayashankara, V.(2014). Development of Speech Database for Hindi Text-To-Speech System Considering Syllable as a Basic Unit. *International Journal of Advanced Research in Computer Science and Software Engineering on, vol., no.4, pp.531-549*

Black, A., Taylor, P., & Caley, R. (2016). Festival Speech Synthesis System. Cstr.ed.ac.uk. Retrieved 27 March 2016, from http://www.cstr.ed.ac.uk/projects/festival/manual/festival_toc.html

Black, Alan W, and Kevin A. Lenzo. "Grapheme-Based Synthesizer". *Festvox.org*. N.p., 2016. Web. 17 Apr. 2016.

Effective Literacy Programmes › L. (2016). Unesco.org. Retrieved 30 March 2016, from http://www.unesco.org/uil/litbase/?menu=13&programme=124

Fletcher, Nathan. *Chale Keyboard*. Ghana: Ketecode, 2016. Print.

Google. (2016). *Google.com.gh*. Retrieved 2 May 2016, from https://www.google.com.gh/

Hazel, M., Nancie, G., Diarmid, M., Fergus, M., Andrea, A., & Mervyn, J. (2011). Usability assessment of text-to-speech synthesis for additional detail in an automated telephone banking system. *Comput. Speech Lang. vol. 25, no. 2*, 341-362. doi:10.1016/j.csl.2010.05.008

Jonathan, A.(1995). *Proceedings of the National Academy of Sciences of the United States of America. vol. 92, no. 22*, pp. 9946-9952

Karthikadevi, M., Srinivasagan, K.G.(2014). The development of syllable based text to speech system for Tamil language. *Recent Trends in Information Technology (ICRTIT), 2014 International Conference on*, pp.1-6, 10-12. doi:10.1109/ICRTIT.2014.6996126

Lawrence, R.(2003). *Science*: New Series, *vol. 301, No. 5639*, pp. 1494-1495

Mario, M., & Philip, G. (2010). Toward language-independent text-to-speech synthesis. *WSEAS Trans. Info. Sci. and App.* 7, 3, 411-421.

Narendra, N. P., Sreenivasa Rao, K., Krishnendu, G., Ramu, R. V., & Sudhamay, M. (2011). Development of syllable-based text to speech synthesis system in Bengali. *Int. J. Speech Technol. vol. 14, no. 3*, 167-181. doi:10.1007/s10772-011-9094-4

Rafieee, M.S., Jafari, S., Ahmadi, H.S., & Jafari, M. (2011). Considerations to Spoken Language Recognition for Text-to-Speech Applications. *Computer Modelling and Simulation (UKSim), 2011 UkSim 13th International Conference on*, pp.304-309. doi:10.1109/UKSIM.2011.64

Rughooputh, S., & Santally, M. (2008). Integrating text-to-speech software into pedagogically sound teaching and learning scenarios. *Education Tech Research Dev Educational Technology Research and Development*, 131-145.

Speech Synthesis. (2016). *http://www.cs.tut.fi/*. Retrieved 26 March 2016, from http://www.cs.tut.fi/kurssit/SGN-4010/puhesynteesi_en.pdf

Yimngam, S., Premchaisawadi, W., & Kreesuradej, W.(2008). State of the Art Review on Thai Text-to-Speech System. *Computer Science and Information Technology, 2008. ICCSIT '08. International Conference on*, pp.194-198. doi:10.1109/ICCSIT.2008.158