ASHESI UNIVERSITY

DEVELOPING A FUNCTIONAL NATURAL LANGUAGE PROCESSING SYSTEM FOR THE TWI LANGUAGE WITH LIMITED DATA

UNDERGRADUATE THESIS

B.Sc. Computer Science

David Sasu

2019

ASHESI UNIVERSITY

Developing a functional Natural Language Processing System for the Twi language with limited data

UNDERGRADUATE THESIS

Thesis submitted to the Department of Computer Science, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

David Sasu

April 2019

DECLARATION

I hereby declare that this Undergraduate 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 Undergraduate Thesis were supervised in accordance with the guidelines on supervision of thesis laid down by Ashesi University.

Supervisor's Signature:

……………………………………………………………………………………………

Supervisor's Name:

……………………………………………………………………………………………

Date:

……………………………………………..………………………………………………

Acknowledgement

This thesis was completed through the generous contributions of many people. As a result, I cannot acknowledge everyone individually here. However, I would like to recognize, with utmost appreciation and gratitude, some individuals for their specific contributions to the success of this theses.

Foremost, I would like to express my sincere gratitude to my advisor, Mr. Dennis Asamoah – Owusu for supporting, motivating and providing consistent feedback throughout this research. Mr. Asamoah - Owusu’s deep understanding of Natural Language Processing and Machine Learning unfailingly guided me in the development of all of the systems that had to be implemented for this research.

Further, I would like to thank Constant Likudie for supporting and advising me as I carried out this research thesis. His often timely and helpful advice helped to ensure that I stayed on-course during the development of this research.

Abstract

Language is a basic characteristic of human beings. It does not only play the vital role in the transmission of information, but it is also pivotal in the establishment of social connection and emotional bond. For centuries, machines had never possessed the ability to exchange ideas with mankind in an intelligent or intuitive way. However, in recent years, due to breakthroughs in the field of Artificial Intelligence and the rise of computational power, machines have made significant and quite impressive gains in the goal of understanding human language and interacting with it. The branch of Artificial Intelligence which is responsible for enabling machines to understand human language is known as Natural Language Processing. Natural Language Processing involves the utilization of statistical and mathematical models to create algorithms that can train machines to learn and understand human language. The major problem with the algorithms that are created in Natural Language Processing is that they require huge amounts of data to train. Unfortunately, this implies that Natural Language Systems cannot be created for languages that do not have large amounts of readily available data. These kinds of languages are called "low resource languages" and most Ghanaian languages, including the Twi language, fall into this category. This research would explore how a functional Natural Language System may be created for the Twi Ghanaian local language with limited language data.

Keywords:

Natural Language Processing; artificial intelligence; machine learning; low-resource languages

Table of contents

List of Figures

Chapter 1: Introduction and Background

Language is the primal form of communication between all human beings. It constitutes of all the different sounds, gestures and symbols that human beings use to express themselves and communicate with one another. It is of high importance because it not only preserves the entire history, culture and knowledge of the human society, but it also ensures the continued survival of humanity. Through language, human beings have the ability learn from one another and collaborate effectively to achieve any set task or goal. It is an extremely influential and powerful tool and depending upon how it is used, it can shape and define reality.

Although language may be expressed in many different forms, the most popular form through which it is expressed is speech. ‘Speech’ is a term which is used to describe the ability to express thoughts and feelings by articulate sounds [1]. According to the Ethnologue, which is regarded as the world’s most extensive language catalogue, there are approximately 7,097 languages that are being spoken in the world today and there are six major language families of the world, which are: Afro-Asiatic, Austronesian, Indo-European, Niger-Congo, Sino-Tibetan and Trans-New Guinea [2].

Even though communication through speech had always been reserved as a special activity that ensued only between human beings, the advent of unprecedented computing power and seemingly limitless technology, such as Artificial Intelligence, in today’s world has enabled mankind to share the activity of speech with machines.

Enabling computers to understand speech is a very important task because, with the increasingly high demand that is being placed on computers for the performance of more complex tasks, there is the need to develop a way in which these complicated requests could be made to the computer system in a very easy and natural way. Natural Language Processing (NLP) is the area of computer science research that enables computers to understand and manipulate natural language text or speech. It involves understanding how human beings learn and use natural language so that technological tools and systems can be built to utilize natural language to perform desired tasks. It began in the 1950s and has since then evolved. According to the research paper entitled “Natural Language Processing: a historical review”, Natural Language Processing has gone through 4 different phases of development since its inception. The first phase was driven by Machine Translation, the second phase was influenced by Artificial Intelligence, the third phase was the “grammatico-logical” phase and the fourth phase was heavily focused on lexical and corpus data [3].

In the first phase, machine translation in NLP was implemented as a lookup, where each word was processed individually by checking its value in a given dictionary [3]. This approach was problematic because it led to the formulation of syntactic and semantic errors [3]. As a result of this, a large portion of this first phase of NLP was heavily focused on syntax processing strategies which could help mitigate the errors caused by the approach which was being used [3].

In the second phase, NLP was coupled with Artificial Intelligence and emphasis was placed on ‘world knowledge’ and the role it played in the construction and manipulation of meaningful representations [3]. The approach adopted in this phase of NLP involved the construction of knowledge bases to satisfy certain specific user input [3]. The real challenges faced by NLP researchers in this phase was developing a general purpose front-end and providing for the acquisition of application-specific knowledge to handle a user’s real needs in dialogue [3].

In the third phase, NLP systems were developed based on grammatical theory [3]. In this phase, NLP researchers developed a range of grammar types which could not only be computed but could also be parsed [3]. Therefore, the processing paradigm for the analysis of an input sequence was based on the interpretation of the syntax of the input sequence into a logical form [3].

In the fourth phase, lexicons were used to replace syntactic general rules in the development of NLP systems and there were significant advances in NLP technologies such as Speech Recognition [3]. This phase was also characterized by a rapidly growing interest in the development and provision of linguistic resources [3].

NLP is a very important piece of technology because it not only provides us with a more native way of interacting with computer systems, but in the grander scheme of things, it also possesses the ability to enable us to make meaning out of immense and seemingly unrelated pools of data. Some popular, real-world applications of NLP can be found in customer or personal virtual assistants, text summarization systems and chatbots.

Natural Language Processing systems need a lot of human language data to be able to work effectively and efficiently. Some languages have a lot of already recorded and easily accessible data which can be used to build NLP systems. However, there are many other languages that do not have a lot of recorded data and thus do not have effective and scalable NLP systems built for them. These languages are referred to as "low resource languages" and native Ghanaian languages, such as the Twi language, fall in this category.

The goal of this research is to therefore determine what it would take to build a functional NLP system for the Twi local language under the constraint of limited data. The functional NLP system that would be built for this research is a "Search Twi song" system. This song system would take in speech data, transcribe it and use the transcription generated to search for a song title from a repository of Twi song titles. This song system would then play the YouTube video of the song title that is eventually selected from the search.

1.1 Research questions

How to develop Natural Language Processing Systems for low-resource languages is a currently a very active area of Natural Language Processing research. As a result of this, Natural Language Processing researchers are consistently formulating better algorithms which can not only efficiently train on very small amounts of data but can also optimize the data selection process itself. Since this research is centred on the development of a functional Natural Language Processing system for the Twi Ghanaian local language with limited data, the research questions that would be explored and answered in this work are:

1) How far can we go with building an NLP system for the Twi language with limited data?

2) Can limited speech data collected from one speaker be used in the development of a functional NLP system?

3) Can limited speech data collected from multiple speakers be used in the development of a functional NLP system?

Chapter 2: Literature Review

2.1 Low-Resource Languages

A ‘Low-resource Language’ can be simply defined as a ‘language that has very limited annotated resources’ [4]. The biggest problem with low-resource languages in developing Natural Language Processing Systems is that the resources that are needed for these languages are extremely difficult to attain. Much of the language information and description of these languages are either unpublished , exist in only paper format or simply do not exist. In most scenarios, even when the language information of these languages exists in electronic format, it cannot be used simply because it is represented in a way that makes it either extremely difficult or impossible to use. As a result of all this, even raw text in a low resource language can be difficult to obtain and use [5].

Another problem with low-resource languages is that the orthographies for these languages may not be standardized. In such languages, the word boundaries, spellings of words and even the usage of the language itself can vary from one speaker of the language to another [6].

The third major problem with low-resource languages is that the dialects of these languages and the relationships between them are often unclear. That is, it is not uncommon when dealing with low-resource languages to find two different texts in the same language, from two different data sources, exhibiting substantial differences in language structure [4]. This posses a huge problem to Natural Language Processing researchers because in such a scenario, they would not know which tools to use for such texts.

2.2 Previous approaches to low-resource languages in Natural Language Processing

Previous approaches to developing Natural Language Processing Systems for low-resource languages can be put into two major categories. These major categories are: (1) approaches that focus on a small set of languages, and (2) approaches that are applied to a very large set of languages simultaneously [4].

The first category generally focusses on a single language or a very small set of languages. In this category, data about the text or speech in the language(s) of interest is first collected and subsequently used in the development of an NLP system. This approach to low-resource languages does yield useful results, but it usually requires the aid of an expert and it is not immediately applicable to other languages [4].

In the second category, the language data of many low-resource languages are collected simultaneously, organised, shared and used in the development of NLP systems. An example of this approach to the development of NLP systems for low-resource languages can be seen in the Crúbadán project that was executed by Scannell [4]. In this project, Web search queries were crafted to return Web-pages that contain specific low-resource languages. The Crúbadán project has enabled the development of corpora for approximately 2000 different languages and these corpora have led to the development of many different tools and resources for low resource languages including thesauri [7]. However, a drawback of Scannell’s approach is that it relies on the manual effort of expert volunteers, without which none of the tools and resources would have been created [4].

2.3 Automatic Speech Recognition Systems

Automatic Speech Recognition refers to the process of deriving the transcription of an utterance given the speech waveform [8]. Therefore, an Automatic Speech Recognition System (ASR) is a computer program that maps an acoustic signal to a string of words [9]. Automatic

Speech Recognition Systems were initially developed in the 1950s for individuals in the disabled community and much of the research that was being conducted in the development of this technology was initially funded by the National Science Foundation and the Defense Advanced Research Projects Agency [10].

There are three main approaches to speech recognition when building automatic speech recognition systems and they are: Acoustic-Phonetic approach, Pattern Recognition approach and Artificial Intelligence approach [11]. The Acoustic-Phonetic approach is also known as the "Rule-based" approach. This approach uses the knowledge and rules of phonetics and linguistics to identify individual phonemes, vowels, sentence structures and sentence meanings. The Acoustic-Phonetic approach usually performs poorly when used in the development of automatic speech recognition systems because of the difficulty in the expression of phonetic and linguistic rules [11].

The Pattern Recognition approach can be broken down into two main steps. These two main steps are: (1) training of speech patterns and (2) recognition of speech patterns by way of pattern recognition [11]. In the first step of the pattern recognition approach, labelled training examples via a formal training algorithm are formed for matching in the future [12]. However, in the second step of this approach a test pattern is formulated from an input signal and this unknown test pattern is compared with each sound reference pattern that was formed during training in the first step of the approach, with the end goal being to identify the sound reference pattern that is most similar to the unknown test pattern [11].

The Artificial Intelligence approach is a combination of the Acoustic-Phonetic approach and the Pattern Recognition approach. In this approach, an expert system is usually implemented with neural networks to classify sounds [11].

Automatic Speech Recognition Systems are usually built based on pattern recognition systems and they demonstrate the following features: Voice Signal Pre-processing, Feature

Extraction and Pattern Matching [12]. Voice Signal Pre-processing represents the first phase of speech recognition technology. In this phase, a voice signal is first input into a speech recognition system with the help of a microphone and it is subsequently processed and transformed into an electrical signal. After the input voice signal has been processed, its features and characteristics are extracted. In this phase of feature extraction, the important characteristics of the input voice signal are extracted and used in the generation of a voice template. After the phase of feature extraction, the process of pattern matching is initiated. During pattern matching, the extracted features of the input voice signal are compared with the characteristics of other voice templates, which are stored in a Model Reference Library within the Speech Recognition System, in the bid to find a match [12].

2.4 Automatic Speech Recognition Systems for low-resource languages

Due to the vast amount of work and research done in the development of automatic speech recognition systems for low-resource languages in recent years, it is almost unfeasible to conduct a detailed review of all the proposed approaches in the literature. As such, this section presents only an overview of a few of the proposed systems but a more complete review of the development of automatic speech recognition systems for low-resource languages can be found in [13].

Kathol *et al.* [14] developed a method for creating an automatic speech recognition system for a low-resource language called ‘Pashto’ in spite of the lack of orthographic information for the language and the lack of an already existing corpus of data for the language. To obtain orthographic information for Pashto, Kathol *et al.* had to transcribe the acoustic data relating to the language directly into a phonemic representation of the language. Kathol *et al.* also had to generate a usable corpus for Pashto from the manual transcriptions of a news series called

“Voice of America”, which was broadcasted in Pashto. After obtaining the corpus and orthographic information of Pashto, Kathol *et al.* developed a language model for Pashto using the minimum discriminative information clustering algorithm. The algorithm generated a clustering tree in which the root represented the whole vocabulary of Pashto and every other node represented a class that consisted of all of the words in the descendant nodes of that particular node. The speech recognition system that was built by Kathol *et al.* had an initial Word Error Rate (WER) of 21%. But however, after changing the orthographic points in the data used, the WER of the speech recognition system ranged from 19.4% to 29.7%. The major drawback of Kathol *et al.*’s approach was that the linguistic resources that was used in the creation of the speech recognition system were not of high depth, quality or reliability.

Unlike Kathol *et al.,* Rygaard [15] approached the problem of the lack of orthographic information of low-resource languages in the development of an automatic speech recognition system by exploring data augmentation methods. The form of data augmentation that was used by Rygaard in the creation of an automatic speech recognition system is known as Speech Synthesis. Rygaard’s method involved first using Text-to-Speech software to synthesize the language data of low-resource languages which may be found on the web. Rygaard then used the synthesized audio obtained from the Text-to-Speech software to train acoustic models for automatic speech recognition. In this approach, the process of training and synthesis was done using a Hidden Markov Model (HMM) – based speech recognition system called HTS and the speech recognizers of the speech recognition system were built using an open-source tool called Kaldi. Rygaard tested out this approach by synthesizing a movie subtitles dataset and a blog utterances dataset, and training language models on these two distinct datasets. After testing, it was discovered that the performance of an ASR system that was trained using 4.4 hours of synthesized movie subtitles dataset improved by 0.81 absolute points. It was also discovered that, training with 11.3 hours of synthesized blog utterances, led to a 0.93 absolute reduction in the WER of an ASR system. One major drawback of Rygaard's approach is that it is heavily dependent upon the language data of low-resource languages which can be found on the web. For low resource languages that have very little or no language data on the web, this method cannot be applied effectively.

Liu *et al.* [16] developed a method for building a phone recognition system for a low-resource language from the mismatched transcriptions of speech in the target language. Mismatched transcriptions of speech in a target language refer to the transcriptions provided by people unfamiliar with the language, using English letter sequences [16]. The speech data that was used in the execution of this approach was extracted from the publicly available Special Broadcasting Service Australia (SBS) radio podcasts hosted in 68 different languages. Liu *et al.*'s approach involved first building a baseline multilingual system that was trained using native transcriptions from several different languages (not including the target language), and then adapting the parameters of the acoustic model of this system using only probabilistic phone transcriptions in the target language derived from mismatched transcriptions [16]. This approach provided up to 25% of relative improvement in phone error rates when tested on an unseen evaluation set [16].

The Feed-forward Neural Network Language Model (ff-NNLM) has also been used to develop an ASR system for low-resource languages. It is an n-gram language model where the posterior probability distribution of the following word is computed for a given n-1 history using a neural network model [17]. Gandhe *et al.* [17] used this approach to conduct experiments which proved that NNLMs are more effective than standard n-gram language models , such as the a modified Kneser-Ney (mKN) smoothed trigram language model, in the development of ASR systems for low-resource languages. In Gandhe *et al.*'s approach, ff-NNLMs were trained using a projection layer of 100 and a hidden layer of 150 units, and an mKN smoothed trigram language model was trained using the SRILM toolkit [17]. Gandhe *et* *al.* then performed speech recognition using the Janus recognition toolkit, which can apply n-gram language models, NNLMs and combined n-gram + NNLM models during both speech recognition decoding and lattice re-scoring [17]. The results that were obtained after Gandhe *et al.*'s experiments indicated that ff-NNLMs performed better than the mKN smoothed trigram language model when applied to either a limited-data training set or a full-data training set.

Some approaches to developing ASR systems for low-resource languages involve an end-to-end speech recognition system based on the attentional model, which was originally proposed for machine translation and developed by Chorowski *et al.* [18]. Other approaches also involve a bidirectional recurrent neural network with a connectionist temporal classification loss function, which was developed by Maas *et al.* [19] and Graves *et al.* [20], to generate the transcription of a low-resource language character by character.

2.5 Low-resource speech data collection for the development of Automatic Speech Recognition Systems

Raw speech signals can be cheaply collected for many low-resource languages through radio broadcasts, fieldwork or public speech [18]. However, the raw speech waveform does not carry much information on its own and this is why field workers usually collect some annotation for a low-resource language [18]. Transcription is usually used for low-resource languages having some writing system [18].

For most previous approaches, low-resource speech data had been collected for the development of speech recognition systems through the use of mobile applications. Vries *et al.* [18] introduced a smart-phone-based data collection tool, Woefzela, to collect speech transcriptions with the focus on quality control [18]. They collected almost 800 hours of speech on their South African data collection project in the bid to demonstrate the ability of smart-phone devices to cheaply and efficiently collect data [18]. Bird *et al.* also introduced a smart-phone application called Aikuma to collect parallel speech between a low-resource language and a higher resource language for the purpose of language preservation [18]. For the initial experiment, Bird *et al.* managed to collect approximately 10 hours of parallel speech from indigenous communities in Brazil and Nepal [18]. The BULB (Breaking Unwritten Language Boundary) project also employed an extension of Aikuma to collect more than 100 hours of parallel speech with monolingual re-speaking for higher speech quality of several African languages at reasonably low cost [18].

Although mobile applications provide a cheap and easy way to collect low-resource language data, one of the main downsides that they present through their use is the risk related to data security and privacy. The data that is collected regarding low-resource languages may be stolen or corrupted when stored on mobile devices or even transmitted between them.

2.6 Text similarity algorithms for Text Search Systems

Text similarity algorithms are sequences of instructions that are followed to measure the degree of likeness of given texts. These algorithms are able to assess the similarity between texts by taking into consideration the lexical and semantical structure of given texts [21]. Words are similar lexically if they have the same character sequence [21]. On the other hand, words are similar semantically if they are used in the same way, used in the same context or share the same type [21]. This research would focus on text similarity algorithms that assess the similarity between given texts on a lexical basis.

With regard to lexical text similarity algorithms that can be implemented for text search systems, the literature shows that the Damerau-Levenshtein, Jaccard Similarity, Cosine Similarity and Ratcliff-Obershelp Similarity algorithms are all examples of good text similarity algorithms [21]. However, in this research the Ratcliff-Obershelp algorithm is used to implement the search system. The Ratcliff-Obershelp algorithm is able to compute the similarity between any two given strings by first doubling the number of common characters between the two strings and then dividing the result obtained by the total number of characters in the two strings [22].

2.7 Summary of related literature

Thus far, the literature shows that the major problems with developing ASR systems for low-resource languages are that: (1) the linguistic resources for low-resource languages are generally difficult to obtain. (2) The orthographic information of most low-resource languages is not standardized. (3) The dialects of most low-resource languages and the relationship between them are often unclear.

With regards to the above problems related to the development of ASR systems for low resource languages, the literature shows that approaches such as: (1) obtaining the linguistic resources of low-resource languages from unconventional sources such as news broadcasts or websites, (2) creating and using synthetic low-resource language data and (3) using mismatched transcripts, can help address the issue of where to obtain resources for the development of ASR systems for low-resource languages. The literature also makes mention of the fact that mobile-phone applications are the most common means through which low-resource language data is currently collected and it also warns of the potential risk of data loss or comprise that could occur through the use of the mobile devices.

Regarding the issue of which language model is best suited for the development of ASR systems for low-resource languages, the literature shows that NNLMs perform better than standard n-gram language models when used in the development of ASR systems. The literature also shows that text similarity algorithms can be executed on the basis of the lexical or semantical similarity between any given texts. However, the literature also reveals that, when dealing with text similarity algorithms that measure textual similarity on a lexical basis, algorithms such as: the Damerau-Levenshtein algorithm, the Ratcliff-Obershelp algorithm, the Cosine Similarity algorithm and the Jaccard Similarity algorithm are all high performing algorithms that can be used to implement text search systems.

Chapter 3: Approach and Methodology

The main hypothesis of this research is that a functional NLP system can be developed for the Twi local language with limited data. In evaluating the above proposition, speech data was first collected from selected Twi-speaking volunteers and used to build an Automatic Speech Recognition System. After the Automatic Speech Recognition System was built, a Search System was implemented to allow users to speak a Twi song title and have the video of the Twi song that they mentioned play on YouTube.

3.1 Collecting Speech Data

The data that was used for this research is made up of 279 Twi language voice recordings that were collected from selected Twi-speaking students and faculty of Ashesi University. The total duration of all the collected Twi language voice recordings was 0.0775 hours (4.65 minutes), with each Twi language voice recording having an average duration of 0.0002778 hours (0.016668 minutes). In this initial stage of data collection, each selected research participant was asked to repeat 9 different Twi sentences which were mentioned to them. The Twi sentences that the research participants were asked to speak can be found in the Appendix section of this research paper.

As each research participant spoke out the Twi sentences, their voices were recorded with a voice recorder and the record audios were stored in a .wav format. After the recorded audios were obtained, they were split into two distinct datasets. With one dataset containing recorded audios provided by one speaker and the other dataset containing recorded audios provided by multiple speakers.

3.2 Building the Automatic Speech Recognition System

The speech recognition process which was implemented in the ASR system can be represented in 3 stages. These three stages are: (1) feature extraction or signal processing, (2) acoustic modelling or phone recognition and (3) decoding [9].

In the feature extraction stage, an input acoustic waveform is sampled into frames (usually of 10, 15, or 20 milliseconds) that are transformed into spectral features [9]. Each frame is represented by a vector of 39 features representing the spectral information of the frame as well as information about energy and spectral change [9]. The feature representation used in this research is the MFCC, the mel frequency cepstral coefficients.

In the acoustic modelling phase, the likelihood of the observed spectral feature vectors is computed given linguistic units (words, phones or subparts of phones) [9]. In this stage, Gaussian Mixture Models were used to compute the likelihood of the feature vectors.

In the decoding phase, the sequence of acoustic likelihoods which were generated from the acoustic modelling phase together with a Hidden Markov Model dictionary of word pronunciations is combined with a language model to output the most likely sequence of words [9].

A flow chart diagram of the ASR system is shown in Figure 3.1 below:

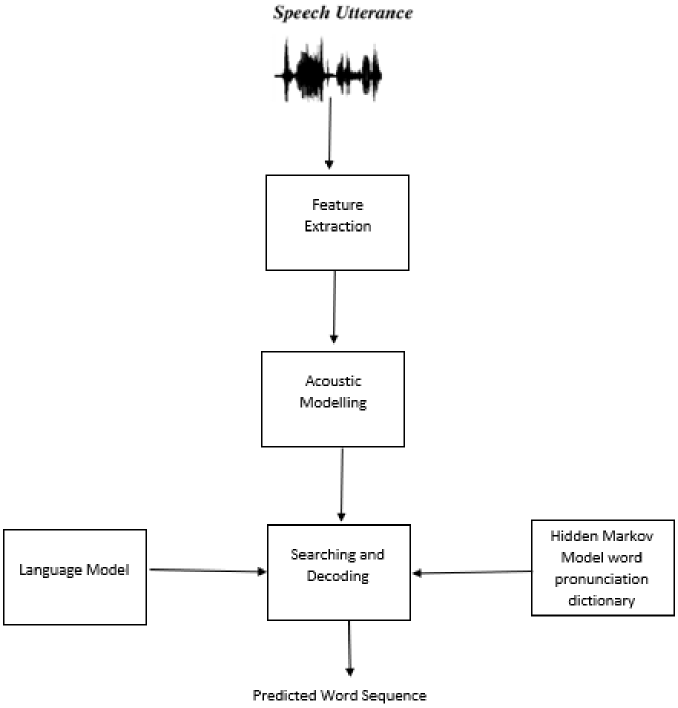


Figure 3.1: Flow chart diagram of the ASR system [9]

3.2.1 Machine Learning algorithm used in developing the ASR system for the Twi language

The machine learning algorithm which was used to implement the ASR system in this research is called Context Dependent Hidden Markov Models with Gaussian Mixture Models (CD HMMs GMMs).

3.2.2 Hidden Markov Models

A Hidden Markov Model (HMM) is a probabilistic sequence model that computes a probability distribution over possible sequences of labels and chooses the best label sequence for a given sequence of units (words, letters, morphemes, sentences) [9]. Hidden Markov Models work by augmenting Markov Chains. A Markov Chain is a model that tells us something about the probabilities of sequences of random variables, *states*, each of which can take on values from some set [9].

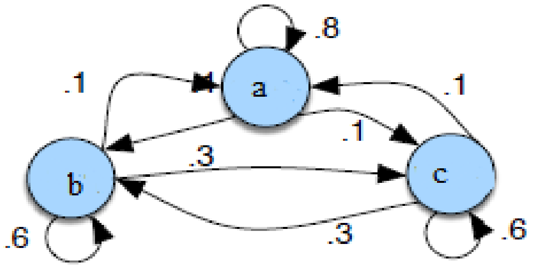
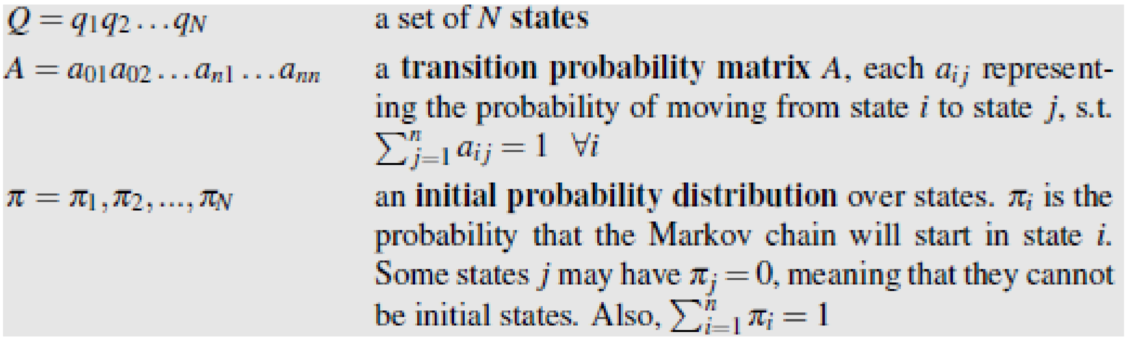


Figure 3.2: A Markov Chain showing states and transitions [9]

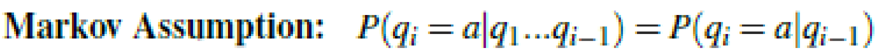
The states in a Markov Chain are represented as nodes in a graph and the transitions, with their probabilities, as edges, as showcased in Fig.2 . Since the transitions are probabilities, the values of arcs leaving a given state must sum to 1 [9]. Formally, a Markov Chain is specified by the following components:



$Q = q_1 q_2 \ldots q_N$	a set of N states
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Figure 3.3: Components of a Markov Chain [9]

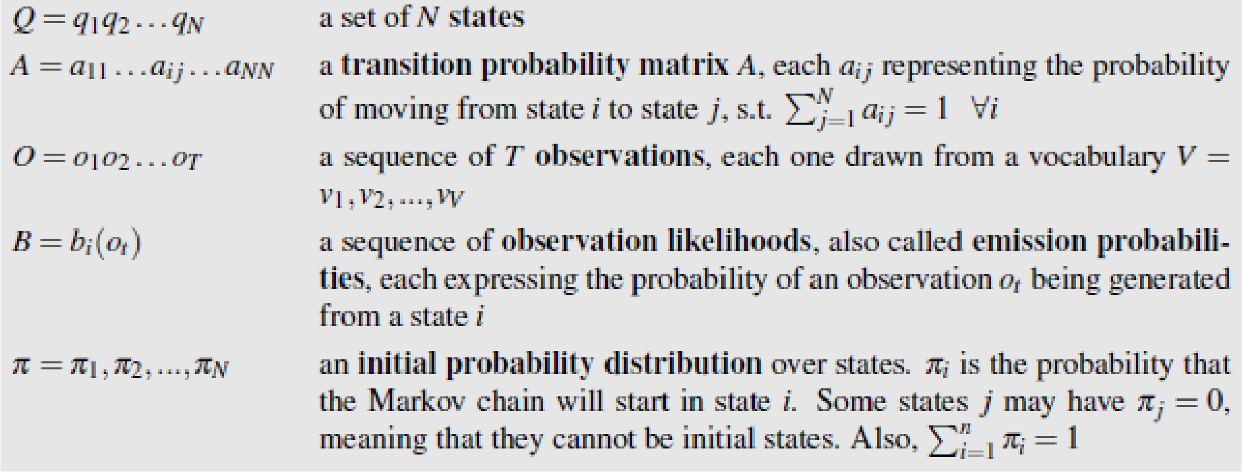
Markov Chains operate under the Markov Assumption, which states that in dealing with the probabilities of a given sequence, the probability of the future does not depend on the past but only on the present.



Markov Assumption: $P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})$

Figure 3.4: An expression for the Markov Assumption [9]

However, unlike Markov Chains, the Hidden Markov Model allows us to talk about both *observed* events (like words that we see in the input) and *hidden* events (like part-of-speech tags) that we think of as causal factors in our probabilistic model [9]. Hidden Markov Models are specified by the following components:



$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Figure 3.5: Components of a Hidden Markov Model [9]

Hidden Markov Models operate under two simplifying assumptions. First, Hidden Markov Models operate under the assumption that the probability of a particular state depends only on the previous state. This is shown in Figure 3.6 below:

$$P(q_i = a | q_1 \ldots q_{i-1}) = P(q_i = a | q_{i-1})$$

Figure 3.6: The first assumption of a Hidden Markov Model [9]

Second, the probability of an output observation o_i depends only on the state that produced the observation q_i and not on any other states or any other observations. This is shown in Figure 3.7 below:

$$P(o_i | q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i | q_i)$$

Figure 3.7: The second assumption of a Hidden Markov Model [9]

For the Hidden Markov Model, the process of determining the hidden variables corresponding to a sequence of input observations in know as decoding. This process of decoding can be summarized into the expression shown in Figure 3.8 below:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

Figure 3.8: The decoding expression of a Hidden Markov Model [9]

where

$\hat{t}_1^n$ = the most probable tag sequence given an input sequence of n words

t_1^n = a tag sequence consisting of 1-to-n elements

w_1^n = a word sequence consisting of 1-to-n elements

The decoding algorithm for the Hidden Markov Model is known as the Viterbi algorithm. The Viterbi algorithm decodes the given input sequence in a Hidden Markov Model by first setting up a probability matrix, with one column for each observation O_t and one row for each state q_t. Given that we had already computed the probability of being in every state at time *t*-1, we compute the Viterbi probability of the current cell, v_t (j) , by taking the most probable extensions of the paths that lead up to the cell. The Viterbi algorithm fills each cell recursively and the Viterbi probability of a given cell can be computed with the following expression:

$$v_t(j) \;=\; \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

Figure 3.9: An expression for the Viterbi algorithm [9]

where

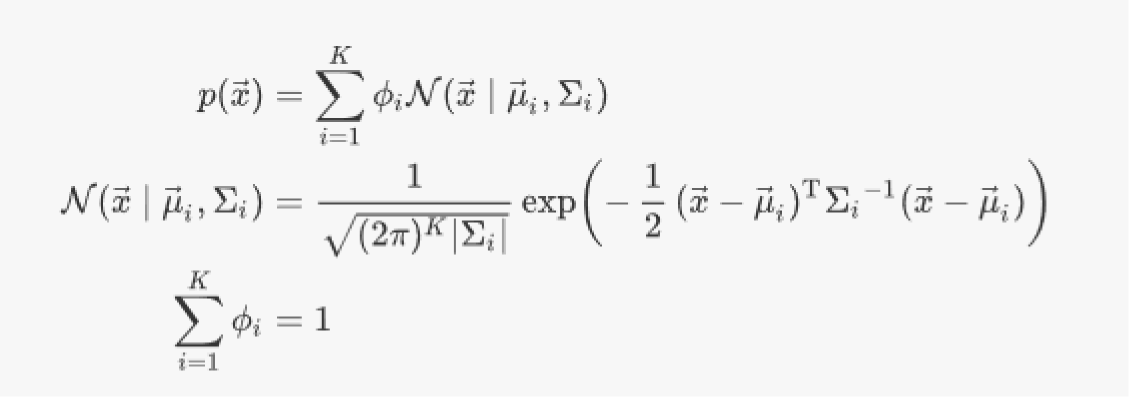
v_t (j) = the previous Viterbi path probability from the previous time step

a_{ij} = the transition probability from the previous state *qi* to the current state *qj*

$b_j(o_t)$ = the state observation likelihood of the observation symbol o_t given the current state *j*

3.2.3 Gaussian Mixture Models

A Gaussian Mixture Model is a probability distribution that consists of multiple probability distributions [20]. It is often used in ASR systems to assign a likelihood score to acoustic feature observations. That is, it helps to determine how likely a particular HMM state *q* (words, letters, morphemes, sentences) generated a given acoustic feature vector. It is parameterized by values representing a mixture of component weights, component means and covariances [20]. Gaussian Mixture Models can be represented as:



$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$$

$$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2} (\vec{x} - \vec{\mu}_i)^{\mathrm{T}} \Sigma_i{}^{-1} (\vec{x} - \vec{\mu}_i) \right)$$

$$\sum_{i=1}^{K} \phi_i = 1$$

Figure 3.10: Expressions for Gaussian Mixture Models [20]

where

$\vec{x}$ = a vector consisting of the input values to be modelled

ϕ_i = the weight of the ith Gaussian

$\vec{\mu}_i$ = the mean of the ith Gaussian

Σ_i = the covariance of the ith Gaussian

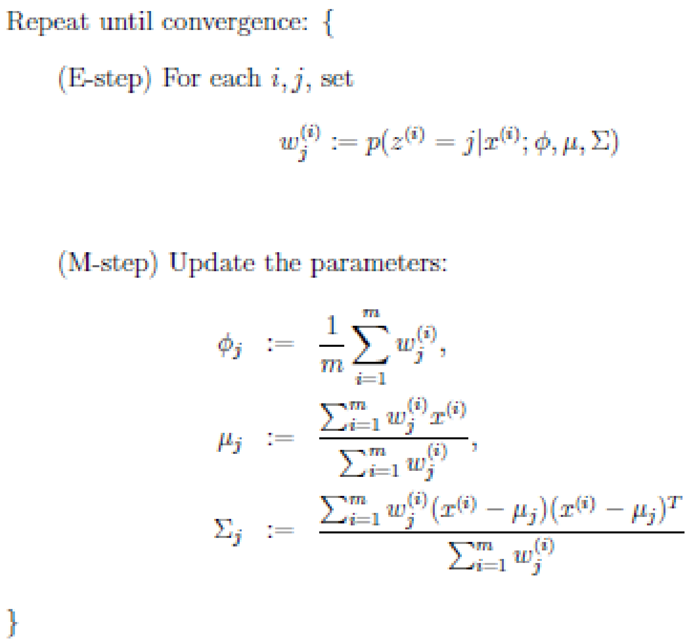
K = the number of Gaussians

$p(\vec{x})$ = the probability that a vector of input values *x* was drawn from the mixture of Gaussian models

$\mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i)$ = the Gaussian distribution of a vector of input values *x*, given the mean and covariance of the *ith* Gaussian

In building ASR systems, a Gaussian Mixture Model instead of a regular Multivariate Gaussian is often used because a particular cepstral feature within a given input feature vector may have a very non-normal distribution and the assumption of a normal distribution may be too strong an assumption [9].

To estimate the parameters of a Gaussian Mixture Model (ϕ_i, $\vec{\mu}_i$, Σ_i) when given a vector of input values, the Expectation Maximization (EM) algorithm is used. The EM algorithm is an iterative algorithm with two steps: an expectation or E-step and a maximization or M-step [23]. In the E-step, the algorithm tries to “guess” the probability of each gaussian (the weight of each gaussian) given a vector of input values and in the M-step, the algorithm updates the parameters of the model based on the guesses made. The Expectation Maximization algorithm is showcased in Figure 3.11 below:



```
Repeat until convergence: {
    (E-step) For each $i, j$, set
        $w_j^{(i)} := p(z^{(i)} = j | x^{(i)}; \phi, \mu, \Sigma)$

    (M-step) Update the parameters:
        $\phi_j := \frac{1}{m} \sum_{i=1}^{m} w_j^{(i)},$
        $\mu_j := \frac{\sum_{i=1}^{m} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{m} w_j^{(i)}},$
        $\Sigma_j := \frac{\sum_{i=1}^{m} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{m} w_j^{(i)}}$
}
```

Figure 3.11: The Expectation Maximization Algorithm [23]

In the algorithm in Figure 3.11 above:

$w_j^{(i)}$ = the weight of the *jth* gaussian given the *ith* input vector example

ϕ_j = the probability of the *jth* gaussian

μ_j = the mean of the *jth* gaussian

Σ_j = the covariance of the *jth* gaussian

3.2.4 Hidden Markov Models with Gaussian Mixture Models

In this machine learning approach, Hidden Markov Models assume a Gaussian Mixture Model (with a variable number of clusters) in each of its states and the usage of these states across time is governed by a transition matrix [24]. This transition matrix is learned from training data and it defines the probabilities of moving from one state to another, ensuring that the data are optimally explained [24]. The structure for Hidden Markov Models with Gaussian Mixture Models can be represented in Figure 3.12 as:

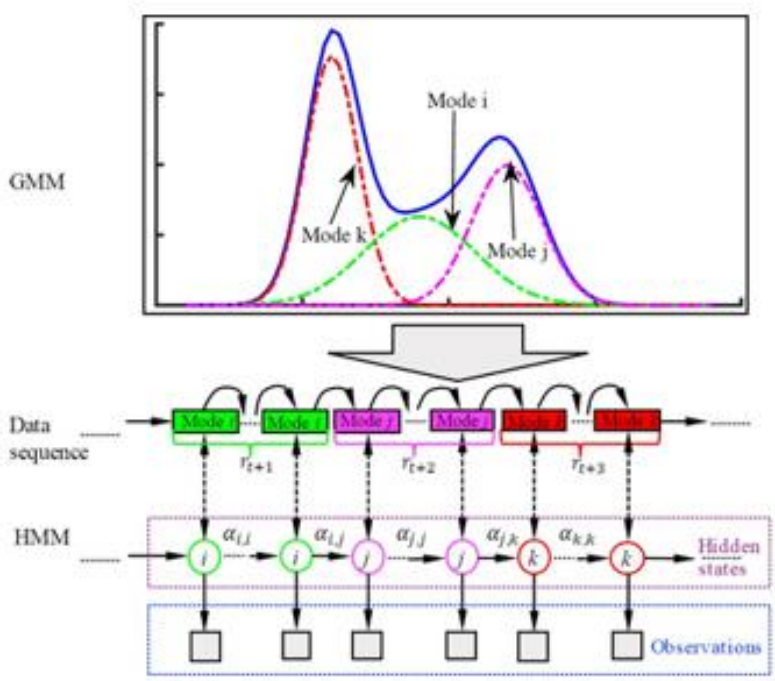


Figure 3.12: The structure of the Hidden Markov Model with Gaussian Mixture Model machine learning approach [24]

3.3 Building the Search System

The main objective of the Search System was to take in transcribed speech data, search for a Twi song title which was most similar to it and play the YouTube video of that song. To enable the execution of the Search System, a repository of Twi song titles of popular Twi songs was created. This repository did not only contain Twi song titles, but also contained the hyperlinks to the YouTube videos of these Twi songs. After the repository was formed, the Search System was built based on the Ratcliff-Obershelp text similarity algorithm. This algorithm was used to determine, for each transcribed speech data provided, the right song video to play on YouTube.

Chapter 4: Methodology 2 – Implementation

4.1 Implementation of Speech Data collection

The Twi sentences for which speech data was collected were chosen because of the following reasons: (1) they are the titles of very popular Ghanaian Twi songs and (2) the music videos of these songs can easily be found on YouTube. The research participants who provided the speech data for this research were obtained by sending out an e-mail, which asked for Twi-speaking volunteers to participate in this research. The speech data which was collected and used in this research consists of both offline audio recordings and live audio recordings. The offline audio recordings consist of voice recordings which were collected, stored and subsequently used in the development and testing of the system. The live audio recordings consist of voice recordings which were collected and immediately used as the system was being tested. After the process of data collection, the recorded audios were split into two distinct datasets by first selecting all of the recorded audios which were provided by one speaker to constitute the first dataset and then selecting the remaining recorded audios to constitute the second dataset.

4.1.2 Implementation of the voice recorder program used to collect speech data

In the implementation of the voice recorder program, the PyAudio and Wave python packages were first imported in the python script of the program. After this first step, the variables in the computer program that represented: (1) the chunk of speech data to be recorded per second (CHUNK), (2) the format in which the speech data was to be recorded (FORMAT), (3) the number of channels in which the speech data was to be recorded (CHANNELS), (4) the sample rate at which the speech data was to be recorded (RATE), (4) the number of seconds for which the speech data was to be recorded (RECORD_SECONDS) and (5) the name of the output file where the recorded speech data was going to be stored (WAVE_OUTPUT_FILENAME), were all initialized. CHUNK was initialized to 1024 bits, FORMAT was initialized to pyaudio.paInt16, CHANNELS was initialized to 1, RATE was initialized to 48000 Hz, RECORD_SECONDS was initialized to 5 seconds and WAVE_OUTPUT_FILENAME was initialized to recorded_speech.wav.

During the process in which the computer program is recording the speech data, an input stream is opened with PyAudio and data frames are drawn from this input stream and stored in a list data structure. After the seconds allotted for the recording of the speech data has elapsed, the Wave python package is used to store the data frames obtained from the speech input data in a .wav file, called “recorded_speech.wav”.

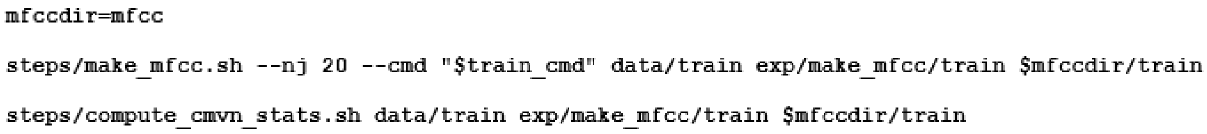
4.2 Implementing the Automatic Speech Recognition System

The Automatic Speech Recognition System that was used in this research was built using an open-source library which is called ‘Kaldi’. Kaldi is a toolkit for speech recognition. It was written in C++ and licensed under the Apache License v2.0. In this research, Kaldi was used to implement the processes of feature extraction, acoustic modelling and decoding.

However, even before the feature extraction, acoustic modelling and decoding processes were executed, certain data preparation tasks were performed. In this data preparation stage, all the amassed speech data were categorized under either training data or test data and the following files were created: text, spk2gender, spk2utt, utt2spk and wav.scp. The text file was created to contain the mappings between utterances and utterance ids. The spk2gender file was created to contain the mappings between speakers and their gender. The spk2utt file was created to contain a mapping between each speaker identifier and all the utterance identifiers associated with the speaker. The wav.scp file was created to contain a mapping between the id of each distinct recorded audio and its corresponding storage location within a computer system.

4.2.1 Implementing Feature Extraction in the Automatic Speech Recognition System

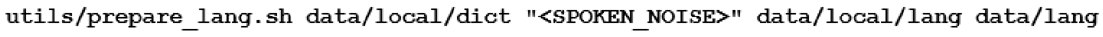
The process of feature extraction in the Automatic Speech Recognition System was executed by running the following 3 lines of script found in the run.sh file, which can be found in the Kaldi program folder:



```
mfccdir=mfcc
steps/make_mfcc.sh --nj 20 --cmd "$train_cmd" data/train exp/make_mfcc/train $mfccdir/train
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir/train
```

4.2.2 Implementing Acoustic Modelling in the Automatic Speech Recognition System

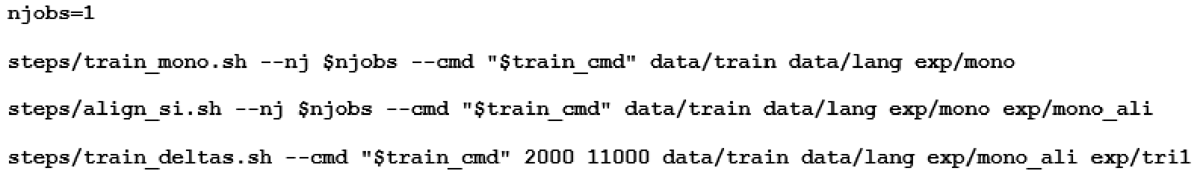
Before the acoustic model was developed, a language model was first constructed. To implement the language model in Kaldi, the following files were initially created: lexicon.txt, nonsilence_phones.txt, silence_phones.txt and optional_silence.txt. The lexicon.txt file was created to contain every word from the language dictionary with its corresponding phone transcriptions. The nonsilence_phones.txt file was created to list the non-silence phones that were present in the collected speech data. The silence_phones.txt file was created to list the silence phones that were present in the collected speech data. The optional_silence.txt file was created to list the optional silence phones that were present in the collected speech data. After these files were created, the next step was to generate raw language files from them. To do this the following line of script found in the run.sh file was executed:



```
utils/prepare_lang.sh data/local/dict "<SPOKEN_NOISE>" data/local/lang data/lang
```

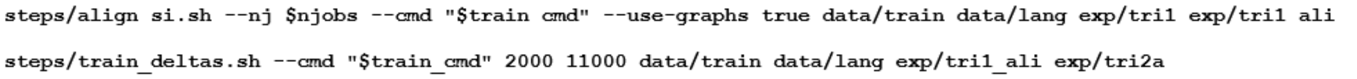
The execution of the above command created a new folder called ‘lang’ which contained a Finite State Transform describing the Twi language.

After the language model was constructed, the acoustic model was developed by first training the Automatic Speech Recognition System on the monophones of the of the speech data collected. This was done by executing the following lines of script found in the run.sh file:



```
njobs=1
steps/train_mono.sh --nj $njobs --cmd "$train_cmd" data/train data/lang exp/mono
steps/align_si.sh --nj $njobs --cmd "$train_cmd" data/train data/lang exp/mono exp/mono_ali
steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train data/lang exp/mono_ali exp/tri1
```

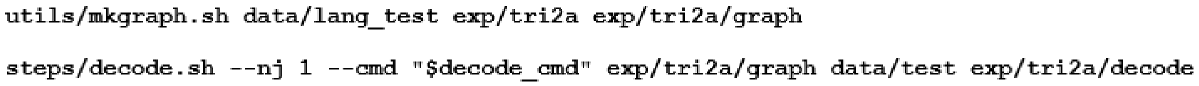
After the process of monophone training, the acoustic model was further developed by training the Automatic Speech Recognition System on the triphones of the speech data collected. This was done by executing the following lines of script found in the run.sh file:



```
steps/align si.sh --nj $njobs --cmd "$train cmd" --use-graphs true data/train data/lang exp/tri1 exp/tri1 ali
steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train data/lang exp/tri1_ali exp/tri2a
```

4.2.3 Implementing Decoding in the Automatic Speech Recognition System

The process of decoding in the Automatic Speech Recognition System was executed by running the following 2 lines of script found in the run.sh file, which can be found in the Kaldi program folder:



```
utils/mkgraph.sh data/lang_test exp/tri2a exp/tri2a/graph
steps/decode.sh --nj 1 --cmd "$decode_cmd" exp/tri2a/graph data/test exp/tri2a/decode
```

4.3 Implementation of the Search System

For the implementation of the Search System, the transcription of the audio file obtained from the Automatic Speech Recognition system was compared to all of the Twi song titles in a Twi song repository using the Ratcliff-Obershelp algorithm. To execute this algorithm in the Search System, the SequenceMatcher function was first imported from the Python language difflib library and used to calculate the level of similarity between the provided transcription and each Twi song title. After this, the Twi song with the highest level of similarity is chosen and its corresponding YouTube link is opened to play the video of the song.

Chapter 5: Experiments and Results

This Chapter discusses the experiments which were conducted to assess the intrinsic and extrinsic performance of the HMM – GMM machine learning model that was developed in this research with limited data. This Chapter also reveals and discusses the results that were obtained from the experiments that were performed.

5.1 Experiments

In this research, 4 different experiments were performed to measure the intrinsic quality of the HMM – GMM machine learning model. In the first experiment, the recorded audio files that were used for training the HMM – GMM model were all recorded from the same speaker and the recorded audio files that were used for testing the HMM – GMM model were not only from the same speaker who also recorded the training audios, but these audios also contained the same Twi words that were used to train the HMM – GMM model.

In the second experiment, the recorded audio files that were used for training the HMM – GMM model were all recorded from the same speaker and the recorded audio files that were used for testing the HMM – GMM model were also recorded from that same speaker. However, the recorded audio files that were used to test the HMM – GMM model contained Twi words that were different from the Twi words that were used to train the HMM – GMM model.

In the third experiment, the recorded audio files that were used for training the HMM – GMM model were all recorded from different speakers and the recorded audio files that were used for testing the HMM – GMM model were also recorded from different speakers. However, the recorded audio files that were used to test the HMM – GMM model contained the same Twi words that were used to train the HMM – GMM model.

In the fourth experiment, the recorded audio files that were used for training the HMM – GMM model were all recorded from different speakers and the recorded audio files that were used for testing the HMM – GMM model were also recorded from different speakers. However, the recorded audio files that were used to test the HMM – GMM model contained Twi words that were different from the Twi words that were used to train the HMM – GMM model.

Even though all these four experiments were performed to obtain answers to the first research question, the first and second experiments were specifically performed to obtain answers for the second research question and the third and fourth experiments were specifically performed to obtain answers for the third research question.

For the extrinsic evaluation of the HMM – GMM machine learning model, a "Search Twi Song" system was developed based on the model. The song system takes in a spoken Twi song title, decodes the speech provided and displays the YouTube video of the Twi song mentioned. In experimenting with this system, the objective was to find out how well the system would perform when: (1) a speaker whose voice was used to train the system uses the system, and (2) a speaker whose voice was not used to train the system uses the system.

5.2 Results from the experiments

The results of the first four experiments were obtained on the basis of calculating the Word Error Rate of the HMM-GMM model in each of the experiments. The Word Error Rate (WER) is a measure of the performance of an Automatic Speech Recognition System [25]. The WER for an ASR system can be calculated by using the following formula:

$$\text{WER} = \frac{S+D+I}{N}$$

where

WER = Word Error Rate

S = The number of substitutions made

D = The number of deletions made

I = The number of insertions made

N = The number of words that were actually said

In the first experiment, the WER that was obtained after testing was done was 0.

In the second experiment, the WER that was obtained after testing was 0.7419.

In the third experiment, the WER that was obtained after testing was 0.216.

In the fourth experiment, the WER that was obtained after testing was 0.7517.

For the extrinsic evaluation of the “Search Twi Song” system created: (1) The system worked 95.4% of the time, out of 10 tries, when a speaker whose voice was used to train the system uses the system. (2) The system worked 94.3%, out of 10 tries, of the time when a speaker whose voice was not used to train the system uses the system.

5.3 Discussions of the results from the experiments

For the results from the experiments regarding the intrinsic evaluation of the developed HMM – GMM machine learning model:

The WER of 0 obtained from the first experiment could be attributed to the fact that: (1) the Twi words that was used for training were the same Twi words that were used for testing and (2) the speech data for both training and testing datasets were provided by the same speaker.

The high WER of 0.7419 obtained from the second experiment could be attributed to the fact that the model was being tested on Twi words that it did not encounter during training.

The low WER of 0.216 obtained from the third experiment could be attributed to the fact that the model had already encountered the Twi words that were used for testing during training. However, the errors made during testing could be attributed to the fact that speech data from different speakers were used for both training and testing.

The high WER of 0.7517 obtained from the fourth experiment could be attributed to the fact that: (1) the model was tested on Twi words that it did not encounter during training and (2) speech data from multiple speakers were used for both training and testing.

For the results from the experiments regarding the extrinsic evaluation of the developed HMM – GMM machine learning model:

The high performance of 95.4% obtained, when a single speaker's voice is used to both train and test the "Search Twi Song" system, can be attributed to the fact that the features of the speech data used in training the system are the same as those that are used in testing the system.

Also, the high performance of 94.3% obtained, when the voices of different speakers are used to train and test the "Search Twi Song" system, can be attributed to the Search System component of the "Search Twi Song" system. This is because, even when the Automatic Speech Recognition System, within the "Search Twi Song" system , provides a transcription which is not entirely accurate, the text similarity algorithm in the Search System would just determine whether the transcription provided is similar in any way to the Twi song titles within the system. If the transcription happens to be similar to any of the Twi song titles, that song title's YouTube video would be played.

Chapter 6: Conclusion and Future Work

6.1 Summary

This research explored the question of whether a functional Natural Language Processing system could be built for the Twi Ghanaian native language with limited language data. The functional Natural Language Processing system that was built in this research was a "Search Twi Song" system. This system was made up of three components, which are: (1) the voice recording component, (2) the Automatic Speech Recognition system component and (3) the Search System component. The voice recording component of the system was built using the PyAudio Python library. The Automatic Speech Recognition component of the system was built based on the Hidden Markov Model with Gaussian Mixture Models (HMM – GMM) machine learning model. The Search System component of the system was built based on the Ratcliff-Obershelp algorithm and implemented with the SequenceMatcher function from the difflib Python library. The limited speech data that was used to train the HMM-GMM model, which was used in the "Search Twi Song" system, was made up of 279 Twi language voice recordings, with a total duration of just 4.65 minutes.

The empirical results from the intrinsic and extrinsic experiments performed on both the developed HMM – GMM machine learning model and the "Search Twi Song" system demonstrate general positive outcomes. Since the "Search Twi Song" system performed relatively well given the small amount of training data available, we can conclude from this research that functional Natural Language Processing systems can be developed for the Twi Ghanaian native language with limited data.

6.2 Limitations

This section discusses the constraints that negatively affected the performance of the machine learning model that was used in this research. The main limitation that was faced in this research was the lack of easily accessible Twi Speech data that could be used to train the machine learning model.

6.3 Suggestion for Future Work

This section presents a suggested extension to this research. By employing the suggestion presented in this section, this research study can be improved upon and applied in a wide variety of ways. The main suggestion offered is that, more Twi speech data can be mined from unconventional sources such as Bilingual Turn Preaching or Twi movies and used in the training of the machine learning model used in this research study.

References

[1] English Oxford Living Dictionary. Speech. Retrieved from: https://en.oxforddictionaries.com/definition/speech

[2] Ethnologue. Summary by language family. Retrieved from: https://www.ethnologue.com/statistics/family

[3] Karen S. Jones. 2001. Natural Language Processing: a historical review. Retrieved October 8, 2018 from https://www.cl.cam.ac.uk/archive/ksj21/histdw4.pdf

[4] Benjamin P. King. 2015. *Practical Natural Language Processing for Low-Resource Languages*. Ph.D. Dissertation. University of Michigan, Ann Arbor, MI.

[5] Steven Bird and Gary Simons. 2003. Seven Dimensions of Portability for Language Documentation and Description. Retrieved March 19,2019 from https://www.researchgate.net/publication/220486407_Seven_Dimensions_of_Portability_for_Language_Documentation_and_Description

[6] F. Lüpke. "Orthography development," in *Handbook of endangered languages* (P. Austin and J. Sallabank, eds.). Cambridge University Press, 2011.

[7] Kevin P. Scannell, "The Crúbadán project: Corpus building for under-resourced languages," in Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus *Workshop*, vol. 4, (Louvain-la-Neuve, Belgium), pp. 5–15, 2007

[8] K. Samudravijaya. nd. Automatic Speech Recognition. Retrieved March 19,2019 from http://www.iitg.ac.in/samudravijaya/tutorials/asrTutorial.pdf

[9] Daniel Jurafsky and James H. Martin. 2014. *Speech and Language Processing*. Uttar Pradesh, India.

[10] Docsoft Inc. 2009. What is Automatic Speech Recognition? Retrieved March 19,2019 from http://www.docsoft.com/resources/Studies/Whitepapers/whitepaper-ASR.pdf

[11] Preeti Saini and Parneet Kaur. 2013. Automatic Speech Recognition: A Review. Retrieved March 19,2019 from http://www.ijettjournal.org/volume-4/issue-2/IJETT-V4I2P210.pdf

[12] Ira Badyal and Divya Gupta. 2015. The State of the Art of Automatic Speech Recognition: An Overview. Retrieved March 19,2019 from https://ijcsmc.com/docs/papers/February2015/V4I2201573.pdf

[13] Long Duong. 2017. Natural Language Processing for Resource-Poor Languages. Ph.D. Dissertation. University of Melbourne, Melbourne.

[14] Andreas Kathol, Kristin Precado, Dimitra Vegyri, Wen Wang and Susanne Riehemann. n.d. Speech Translation for Low Resource Languages: The Case Of Pashto. Retrieved October 8, 2018 from https://pdfs.semanticscholar.org/c365/d34875fdbb62d9e810c42445cf9b49832236.pdf

[15] Luis Rygaard. 2015. Using Synthesized Speech to Improve Speech Recognition for Low-Resource Languages. Retrieved October 8, 2018 from https://parasol.tamu.edu/dreu2015/Rygaard/report.pdf

[16] Chunxi Liu, Preethi Jyothi, Hao Tangz, Vimal Manohar, Mark Hasegawa-Johnson and Sanjeev Khudanpur. 2016. Adapting ASR for Under-Resourced Languages Using Mismatched Transcriptions. Retrieved October 8, 2018 from https://ieeexplore.ieee.org/document/7472797/

[17] Ankur Gandhe. nd . Neural Network Language Models for Low-Resource Languages. Retrieved October 8, 2018 from https://www.cs.cmu.edu/~fmetze/interACT//Publications_files/publications/IS14full_NNLM_lowResource_CameraReady.pdf

[18] Chorowski, Jan, Dzimitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent NN: first results. Retrieved October 8, 2018 from arXive-prints abs/1412.1602.

[19] Maas Andrew, Ziang Xie, Dan Jurafsky, and Andrew Ng. 2015. Lexicon-free conversational speech recognition with neural networks. In Proceeding of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies., 345– 354.

[20] Graves Alex, Santiago Fernandez, and Faustino Gomez. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceeding of International Conference in Machine Learning (ICML), 369–376.

[19] Kevin Gurney. 1997. *An introduction to neural networks.* UCL Press, London.

Neural Network Picture: https://chatbotslife.com/how-neural-networks-work-ff4c7ad371f7

[20] John McGonagle, Geoff Pilling and Vincent Tembo. 2019. Gaussian Mixture Model. Retrieved March 28, 2019 from https://brilliant.org/wiki/gaussian-mixture-model/

[21] Wael H. Gomaa. 2013. A Survey of Text Similarity Approaches. Retrieved March 28, 2019 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.5446&rep=rep1&type=pdf

[22] Mohit Mayank. 2019. String Similarity – the basic know your algorithms guide. Retrieved March 28, 2019 from https://itnext.io/string-similarity-the-basic-know-your-algorithms-guide-3de3d7346227

[23] Dong Yu and Li Deng. 2015. *Automatic Speech Recognition: A Deep Learning Approach*. Springer-Verlag, London.

[24] Farheen Fauziya. 2014. A Comparative Study of Phoneme Recognition using GMM-HMM and ANN based Acoustic Modelling. Retrieved March 28, 2019 from https://pdfs.semanticscholar.org/33e2/e2a620b02251faec6a8f55eb0164f9ba1db9.pdf

[25] Eduard Polityko. 2019. Word Error Rate. Retrieved March 28, 2019 from https://www.mathworks.com/examples/matlab/community/35304-word-error-rate

Appendix

The 9 distinct Twi sentences for which speech data was collected

1) Obia wo ne master
2) Asem beba debi
3) Bue kwan
4) Obi nyani me
5) Aha ye de
6) Yesu wo so
7) Awuradi wo ne m’ade nyinaa
8) Awurade ne me hann
9) Abode nyinaa beyi w’aye