EasyChair Preprint

№ 1440

Tamil talk: What you speak is what you get!

Naomi Weston, David Mann, Raj Ramachandran and Kim Frederic

September 1, 2019

Tamil Talk: What you speak is what you get!

Abstract- Tamil is one of the longest surviving classical languages in the world. Speech to text in Tamil would provide huge benefit to a lot of native Tamil speakers throughout the world. There are many speech recognition and speech to text systems available for a wide variety of languages but many minority languages, such as Tamil are overlooked. In this paper, we propose to develop a system for Tamil speech to text that will be consitent with the pronunciation of the user and conforms wih the syntax of the language.

Index Terms— Speech to text, application, Agile, Tamil Language, Tamil Orthography, Speech Recognition, ASR

1 Introduction

Speech is the ability to express thoughts and feelings by articulating sounds. It is a key component of communication for humans. Each language uses phonetic compostitions of a limited set of vowels and consonant sounds that form words. Speech recognition is the interdisciplinary subfield of computational linguistics that develops methodoligies and technologies that enables the recognition and translation of spoken language into text (Rabiner et al, 2004).

Tamil is a syllabic language with almost one to one correspondence to the sound and orthography (Keane,2004). The work os Shulman (2016), point to unique syllables of Tamil often mispronounced especially the syllable 'zha' by the native speakers for various reasons as also identified and raised by Ramachandran (2018) in the context of not just predicting the user acceptance of speech to text but also to be able to use speech to text in Tamil due to the nature of the language. There are very few studies on recognition of 'zha' primarily by Srinivasan et.al (2009, 2010, 2013). It Is argued that David (2010) views on language variation do not apply in this context due to the syllabic nature of the language. Some of the participant comments from Ramachandran (2018) informed the design and development of Tamil talk:

"Language and spelling cannot be changed"
"You have to pronounce the word correctly"
"People must change their pronunciation"

The first successful speech recognition machine was released by three Bell Labs researchers in 1952 and it was called Audrey. This machine was able to recognize spoken digits with 90% accuracy; however, it only recognized the inventor's voice (Boyd, 2018). The technology then developed into understanding English words about 10 years later. Raj Redy was the first person to research on continuous speech recognition as a graduate student in the late 1960s, as systems prior to this, required users to pause after ever word (Carnegie Mellon, 2010).

Speech recognition technology has grown in sophistication and accessibility leading to the ability to convert speech into text for hundreds of languages and dialects. This includes languages like Tamil which uses a script that is different to Roman letters and uses completely different structure. Nowadays, almost everyone has a smartphone which is more than capable of recognizing spoken language using the most basic of hardware. The research conducted in this paper points to different design possibilities and details of developing a speech to text application. An application was built to convert spoken Tamil to text and display what has been spoken onto the screen of a given device. Example, a smartphone or a laptop.

1.1 Overview

The structure of the paper is as follows:

- The first section deals with the research carried out into the Tamil language and investigates any speech to text applications that are currently available.
- The second section describes the method used to design, develop , implement and test the application developed.
- The third section documents the testing carried out on the developed application.
- The final section provides the discussion and conclusion for the project and suggested work that could be conducted in the future.

1.2 Aims

The aim of this research is to implement a speech to text application in Tamil based on the conceptual framework of Ramachandran (2018) 'what you speak is what you get'.

The aim of application itself is designed to work as a teaching tool for native and non-native Tamil speakers to confirm correct pronunciation of syllbles and words.

2 Literature Review

Speech to text is the conversion of spoken words into text. ASR applications are very widely available and help assist millions of people every day, however, many of these applications have language-based limitations by only supporting a handful of languages. Developing an application that supports "minority languages" would provide great technological benefit and help develop more inclusive and accessible technology (Fu et al, 2018). Pandharipande

(2002) defines minority langaues as a language spoken by less than 50% of the given region, state, or community.This research views the definition of minority language as defined by Pandharipane (2002) outside of the context and region where Tamils are in majority. For example in the Indian state of Tamil Nadu and Puducherry.

2.1 Tamil

Tamil is a Dravidian language spoken by eighty million people in South Asia and across fifty-five diaspora countries but predominantly by the Tamil people of Tamilnadu (Steever, 1998). It is one of the longest surviving classical languages in the world and recorded Tamil literature has been documented for over 2000 years (Zvelebill, 1992).
The Tamil script consists of 12 vowels (Uyir Ezhuthukkal),

Fig 1 Basic Tamil script (Lo, 2012)

18 consonants (Mei Ezhuthukkal) and one special character, the Aytha Ezhuthu (see fig.1). The vowels and consonants combine to form 216 compound characters (Uyir Mei Ezhuthukkal), giving a total of 247 haracters (Lo, 2012). Unlike other South Asian scripts, 3.1such as Gujarati, Tamil does not have script to represent voiceless aspirated (such as “kh”), voiced (“g”) and voiced aspirated stops (“gh”).

2.2 Speech to Text in Tamil

There are many existing speech-to-text software and applications that support the Tamil language. The most commonly used system is Google Translate (McGuire, 2018); which could potentially be used to translate from one language to another, such as French to English, but could also be used for speech to text. Unfortunately, there are shortcomings with the existing application. The existing system does not incorporate the concept of ‘what you speak is what you get’. The concept as seen in Ramachandran (2018) is based on the structure of the language as opposed to an application that accommodates language and pronunciation variation of the users.

This kind of predictive speech-to-text will give the users a sense of pronouncing the words correctly, where in reality the words are pronounced differently. This makes users believe they speak correctly and will teach them the language incorrectly which is a significant issue (Devarajan, 2009).

2.3 Design considerations for Tamil talk

In a standard speech recognition system (fig.2) the raw speech is typically sampled at a high frequency (16KHz – 8KHz) this produces a sequence of amplitude values over time. This raw speech data is then transformed and compressed to enable simplified processing (Deng et al, 2007). Many signal analysis techniques are available which can extract useful features and compress the data by a factor of ten, such as Fourier analysis, Perceptual Linear Prediction, and Lineal Predictive Coding.

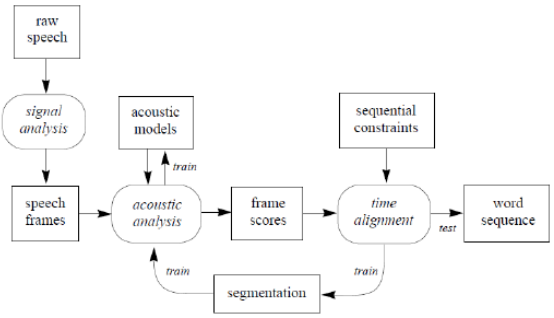

Fig 2. Trivedi (2014)

The speech-to-text transformation is one of the most difficult tasks in computer science because it consists of many difficult problems (Pornpanomchai et al, 2012) but it is an important technology to develop and to develop well. Speech recognition has many potential applications including command and control, dictation, transcription of recorded speech, searching audio documents, and interactive spoken dialogues (Gale and Young, 2007). At the core of all speech recognition systems consists of a set of statistical models representing the various sounds of the language to be recognised (Baum and Eagon, 1967).
The task of speech recognition is to find the best matching word-sequence ($\hat{\mathbf{W}}$) given the data of an utterance ($\boldsymbol{O}$). $\boldsymbol{O}$ is a sequence of input vectors generated from the raw speech data. According to Bayes’ Theorem the task can be formulated as seen below (fig.3).

$$\hat{W} = \arg\max_{W} P(W|O) = \arg\max_{W} \overbrace{P(O|W)}^{AM} \cdot \overbrace{P(W)}^{LM}.$$

Fig 3. Blanken, de Vires, Blok and Feng, 2007

To decode the sequence of words spoken you have calculate the next probability (fig.4) where $\boldsymbol{W}$ is the decoded sequence and $\boldsymbol{O}$ is the observed sequence or incoming feature vector (Blanken et al, 2007).

$$\hat{W} = \arg\max_{W} P(W|O).$$

Fig 4. Blanken, de Vires, Blok and Feng, 2007

This splits the task into two components ***P(O | W)***, also known as the acoustic model and ***P(W)***,which is also known as the language model. In most speech recognition systems the acoustic model is represented by the HMM or the Hidden Markov Model (Kreyssig, 2018). Each HMM is a finite state machine with ***n*** states whereby each state, besides the first and last, has specific output probabilities and each arc between states is associated with a transition probability (Gales and Young, 2007). Previously the output probabilities were modelled by multivariate Gaussian Mixture Model or GMM (Stuttle, 2003). Given restraints in computational power the GMM is restricted to have a diagonal co-variance matrix and hence require independence between the input dimensions.

Artificial neural networks (ANNs) on the other hand, do not need this independence requirement, which is why they are more currently used for acoustic modelling and give increased performance to speech recognition, particularly Convolutional Neural Networks and Recurrent Neural Networks (Duran and Battle, 2018). There are several possible ways to exploit ANNs in automatic speech recognition systems, such as the Hidden Markov Model-Artificial Neural Network hybrid system which takes the advantage of ANN's strong representation learning power and HMM's sequential modelling ability.

There is also research into using what is called Connectionist Temporal Classification (fig.5) which is very similar to the standard HMM-ANN approach however a single-state HMM is used, usually with three times lower output frame-rate and after each single-state HMM a "blank" state is allowed (Graves et al, 2006). Further, this model does not need alignment of the training data, which is an advantage, but only starts to show benefits starting with very large quantities of data (Yu and Deng, 2014).

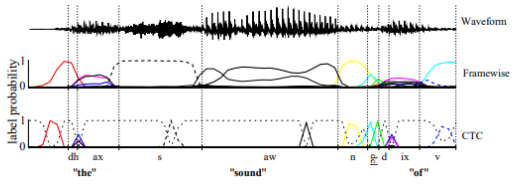

Fig 5. Framewise and CTC networks classifying a speech signal (Graves et al, 2006)

All these options were considered at the beginning but we eventually worked on two options that was thought would provide a complete solution of creating a Speech-to-Text application in Tamil that uses the conceptual framework of "what you speak is what you get". While it was deemed the option to develop an application using the Dictionary Model would be the best fit for this project, both options have been described in sections 2.5 and 2.6, allowing for the information to be used for any enhancement to the solution at later date.

2.4 Application Programming Interfaces

An Application Programming Interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. Many API's are built with the intention to allow 3rd party developers to build interesting applications and designed to expand the reach of an organisation (Clarke, 2004).

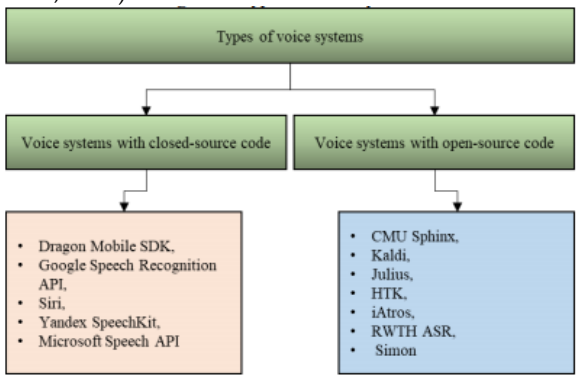

Fig 6. Types of voice systems (Matarneh et al, 2017)

All speech recognition engines/ API's initially work in the same way as the user's voice is passed through the microphone input to reach the recognition system. There are a number of options for Automatic Speech Recognition API's available that can be divided into closed source code and open source code (fig.6).

Closed source code means there is no physical access to the code, a user will be unable to manipulate and modify it. Two of the biggest companies building voice-powered applications are Google and Microsoft; however these API's code is inaccessible (Samudravijaya and Barol, 2013).

CMU Sphinx is one of the most famous systems and is completely open source. Sphinx enables developers to download and use their code freely (Lange and Suendermann, 2014) it includes speech recognisers and acoustic model trainers (Matarneh et al, 2017).

There are many benefits of using API's when building applications, especially if the API meets the specifc needs of an application, then avoiding the reinvention of the proverbial wheel is a standard piece of wisdom in software development circles (Atwood, 2008). If it is not a good fit for a proposed application but the API is open source and

open code then users will be able to access and manipulate the code to fit with an application. Additionally, there are also disadvantages to API's that need to be considered. Anyone could have written the code and if it is closed source you will have no way of knowing what it actually does and you will not have the option or the ability to improve on it.

2.4.1 Google Speech API

Google Speech Recognition API is a technology used widely in different research fields for different language. It allows the user to voice search and its technology is integrated into many smartphones and computers. Originally it only supported a short request of a maximum 40 words and has only recently improved its speech recognition by using a new technology, deep learning neural networks (Kepuska, 2017).

Part of Googles improvement is a major new feature in the Speech-to-text API that now allows developers to select between different machine learning models (Lardinois, 2018). The speech API now supports over 100 languages, including ancient languages such as Georgian (first spoken in 430AD), as well as Swahili, Gujarati and Tamil in a bid to make the internet more inclusive (Techseen Bureau, 2017).

Google acquired several deep learning companies over the years such as, DeepMind, DNNresearch and JetPac and using these deep learning neural networks Google achieve an 8 percent error rate in 2015, a reduction of more than 23 percent from 2013 (Beat and Novet, 2016).

2.4.2 CMU Sphinx

In 1986 Sphinx was launched and became one of the most successful open source systems developed for research purposes using HMM (Juang & Rabiner, 2005). Developed at Carnegie Mellon University (CMU) it currently has one of the largest vocabularies and speaker independent recognition codebase. This speaker independent speech recogniser and uses HMM and n-gram statistical language model, which is able to recognise continuous speech with a big vocabulary (Matarneh et al, 2017).

Since its initial release, Sphinx has gone through a number of different modifications. In the past, the decoding strategy of the Sphinx systems tended to be deeply entangled with the rest of the system. As a result of these constraints, the systems were difficult to modify for experiments in other areas (Walker et al, 2004).

However, the newest version, Sphinx-4, released in 2010, works with various kinds of language specifications such as grammars, statistical language models or blends of both (Twiefel, Baumann, Heinrich & Wermter, 2014). This means a major benefit of the Sphinx system is that researchers are given more flexibility in the way they incorporate acoustic models allowing constraints to be imposed on the input from the user (Ashwell & Elam, 2017), making it a great way for identifying particular phonetic sounds for set phrases.

Sphinx has now developed into a toolkit that can be used to develop powerful speech recognition applications and includes several parts (Belenko & Balakshin, 2017):

1. PocketSphinx is small fast program, processing sound, acoustic models, grammars and dictionaries.
2. The library Sphinxbase is necessary for PocketSphinx work.
3. Sphinx4 is the recognition library.
4. Sphinxtrain which is for acoustic models training.

2.4.3 Microsoft Speech API

Microsoft's Speech API has been around since 1993, developed by three of the four people responsible for the CMU Sphinx-II speech recognition system (Kepuska & Bohouta, 2017). Its system is very similar to Google Speech API but uses a server application programming interface (SAPI) for its data. It includes a set of effective methods and its data is well integrated into the .NET framework (Kamarudin et al, 2013).

Microsoft has continued to develop the powerful speech API and has released a series of increasingly powerful speech platforms (Lacoma, 2019), focussing on increasing the emphasis on speech recognition systems and improved Speech API by using a context dependent deep neural network hidden Markov model (Manaswi, 2018).

2.5 The Hidden Markov Model

The first method considered was the Hidden Markov Model. The basic need was identifying if it would meet the system requirements in terms of providing a solution that not only points to an indigenous method to develop a software application but also, the issue of language maintenance, code switching and code mixing (Shulman 2016, Schiffman 2002) by the native Tamil speakers both in the native region and in the diaspora. Ramachandran (2018) recommends an indigenous approach to design, develop and evaluate the user acceptance of speech to text in language based technology such as speech to text.

A hidden Markov model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with hidden states (Trivedi, 2014). A HMM can be presented as the simplest dynamic Bayesian

network (Wantanabe, 2011). HMM can be seen as a black box, where the sequence of output symbols generated over time is observable, but the sequence of states visited is hidden from view (Juang & Rabiner, 2005).

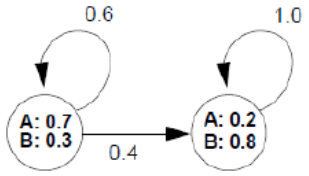

Fig 7. Trivedi, 2014

Figure 7 shows a simple HMM with two states and two output symbols, A and B. When applied to speech recognition, the states are interpreted as acoustic models, indicating what sounds are likely to be heard during their corresponding segments of speech; while the transitions provide temporal constraints, indicating how the states may follow each other in sequence (Trebelskis, 1995).

There are three basic algorithms associated with Hidden Markov Models; firstly the Forward algorithm which is useful for isolated word recognition. The number of possible paths increases with the length if a sequence so the goal of the forward algorithm is to compute the joint probability, taking advantage of the conditional independence rules of the HMM to perform calculations recursively and avoid incurring exponential computation time (Rabiner, 1989).

The second algorithm is the Viterbi algorithm which is a dynamic programming algorithm useful for isolated word recognition. In speech-to-text, the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the hidden cause of the acoustic signal (Anguera et al, 2010). The Viterbi algorithm finds the most likely string of text given the acoustic signal.
The final algorithm is the Forward-Backward algorithm which is an interference algorithm that is useful for training a HMM (Rabiner, 1989). This algorithm involves three steps:

1. Computing forward probabilities
2. Computing backward probabilities
3. Computing smoothed values.

It computes the posterior marginal of all hidden state variables and makes use of the principle of dynamic programming to efficiently compute the values that are requires to obtain the distributions in two passes. The first pass goes forward in time while the second goes backwards also known as the forward message pass and the backward message pass (Juang & Rabiner, 2005).

2.5.1 Feasibility of HMM for Tamil talk

The Hidden Markov Model provides a simple and effective framework for modelling time-varying spectral vector sequences. As a result, almost all large vocabulary continuous speech recognition systems are based on HMMs (Gale & Young, 2007). Since speech has a temporal structure and can be encoded as a sequence of spectral vectors spanning the audio frequency range, the Hidden Markov Model provides a natural framework for speech recognition.

According to Trivedi (2014), there are many limitations of the Hidden Markov Model which include:

- Constant observation of frames
- The Markov assumption
- Lack of formal methods for choosing a model topology
- Large amounts of training data required
- Weak duration modelling
- Restricted output PDFs
- The assumption of conditional independence

However, the main concern when implementing the HMM technique with a speech-to-text application for the Tamil language, is computational efficiency (Srinivasen et al, 2009). Although it is possible to alleviate this problem using the forward-backward procedure this would not suit the main system requirement as it works on estimating and predicting the next sequence rather than using the "what you speak is what you get" framework.

2.6 The Dictionary-Based Approach

The second available option was the Dictionary-based approach. This section investigates and discusses it based on the project.

2.6.1 Description

Human day-to-day speech is referred to as spontaneous speech; it is not scripted and therefore adds a variety of phenomena to a speech recognition task with false starts, human and non-human noises, new words and alternative pronunciations. All of this has to be tackled when creating a system for spontaneous speech recognition (Sloboda and Waibel, 1996). Rather than looking for the "correct" pronunciation of a word the system should automatically expand and adapt the phonetic dictionary and choose a word according to its frequency.

A dictionary-based approach is a basic approach for a speech-to-text system (Pornpanomchai et al, 2012). The pronunciation and sounds are looked up in a sound wave dictionary, these dictionaries are usual modified by hand or by applying phonological rules to a given dictionary

(Sloboda and Waibel, 1996).

Sloboda and Waibel (1996) proposed this Dictionary Learning Algorithm for using both a speech and phoneme recogniser:

1. Collect all occurrences of each word/tuple in the database and run the phoneme recognizer on them using the smoothed phoneme LM
2. Compute statistics of the resulting phonetic transcriptions of all words/tuples
3. Sort the resulting pronunciation candidates using a confidence measure and define a threshold for rejecting statistically irrelevant variants
4. Reject variants that are homophones to already existing dictionary entries
5. Reject variants which only differ in confusable phonemes
6. Add the resulting variants to the dictionary
7. Test with the modified dictionary on the cross validation set (optional)
8. Retrain the speech recogniser, allowing the use of multiple pronunciations during training.
9. As an optional step corrective phoneme training can be performed
10. Test with the resulting recognizer and the modified dictionary on the cross validation set
11. Create a new smoothed language model for the phoneme recogniser, incorporating all new variants.
12. Optional second pass

There are disadvantages to the Dictionary Approach especially when entries are added by hand as this usually focuses on single occurrences of a word and it can introduce a number of errors into the system (Pornpanomchai et al, 2012). When modifying words in a dictionary by hand it is important to be aware that experts tend to use the “correct" phonetic transcription of a word which is not necessarily the most frequent or even the most likely transcription for a given task. The actual pronunciations may also be very different for the pronunciation deemed as “correct”, and in spontaneous speech there are a lot of alternative pronunciations and they are not easy to predict (Diehl et al, 2014).

2.6.2 Feasibility

The phonetic dictionary is one of the main knowledge-sources for speech recognition but it is still regarded as being less important at acoustic or language modelling. As we have seen in speech recognition research the emphasis is often on the correct pronunciation of a word as it can be found in a lexicon, but this correct pronunciation does not have to be the main feature and does not provide the best recognition accuracy.

The Dictionary-Based method proposes a solution to reduce the system complexity, but this method will fail if a word looked up cannot be found in the systems dictionary. However for this application this method should provide the user with the accuracy of their speech and not the “correct” spelling of the word as specified in the requirements.

3 METHOD

Selecting the most appropriate development methodology is not easy. For this project there were two considerations that were priorty, the type and complexity of application that was being developed, stakeholders, timescale and the experience of the development team.

3.1 Requirements and design

This section identifies the requirements expected to be met to produce a successful product for the end client. The requirements must be clear, complete and understandable to the stakeholders so that all parties are aware of what the system should be doing and any constraints to the software development process (Elgabry, 2016).

The functional requirements for the application were captured during a meeting with the sponsor (Raj Ramachandran). The method of project management chosen by the team was agile, so the approach to the development was iterative. The application itself was quite simple in design, meaning there are very functions.

- It should be a web application that is able to run on a PC, smartphone or a tablet.
- The application must be able to print Tamil orthography to represent the spoken Tamil word.
- The application must be able to interpret and understand sound waves produced by the speaker in order to pick up the original word or sound.
- The application must be able to interpret and understand real words and ignore words with errors or mispronunciation.
- As the application uses a microphone, it must understand what to ignore and what to accept, meaning if there is 2+ people speaking or there is background noise, it must focus on just the user.

The above are the functional requirements that entail the capabilities, appearance and interactions this system has with the users also known as the target audience for this project (Rumbaugh, 1999). They were measured during the testing and verification stage which is detailed further on in the paper.

The following are the non-functional requirements for the application:

- The application must be able to recognize the spoken Tamil word to the written Tamil word in real time
- The application must to be available for use by several users simultaneously
- As the application is web based, users should be able to access the application on any internet browser on any device, whether it be Linux, Windows, iOS or Android.
- The application should achieve a high accuracy rate for translation from speech to text
- The application will provide the user with clear options of how to use the features
- The application shall have a simple design that only displays the necessary features
- The system must conform to IEEE standards

The design process started with each member researching applications already available. This research was discussed during initial meetings with the full group and the sponser. A high-level idea of the layout was put together. From this a selection of mockups were created using photoshop, and other design software, and an initial back-end prototype was developed calling on Google's speech recognition software.

Further web design mockups were created and then compiled to create a high-fidelity prototype which was demonstrated to the sponser where feedback was obtained and designs further modified.

This helped us steer in the right direction as to how visually the application should look and this was consistently implemented throughout the design and development stages. This technique proved to be the beneficial as it relied on the perspective of many people.

To visualise the application a series of use cases/user stories were created, to map out the user journey and information flow through the application.

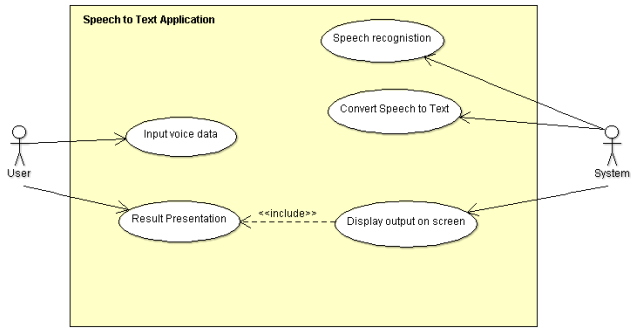

Figure 2 Use Case for the Application

3.2 Implementation

The tools used for initial implementation on this application were CMU Sphinx Python-based libraries, SQL Database and Javascript for the front end. These tools were chosen be the structure as they are able to interlink with one another to create a functioning application.

CMU Sphinx and Pocket Sphinx are the best source for this application as it was designed to support such a project.

4 Architecture of Tamil Talk

The following section dels with the the application from both the back end and client front end.

4.1 System Architecture

Originally the system needed to be in a format that was shareable with others and to provide this it was decided the software will be created in a mobile application format sharable on the Android Play Store or iOS app store. Later on it was requested the application be web-based so to keep in line with these requests, the following technologies were decided to be used:

- Python
- Ionic Hybrid web application framework
- CMU Sphinx PocketSphinx API
- Apache Server
- MySQL
- MVC Design Pattern

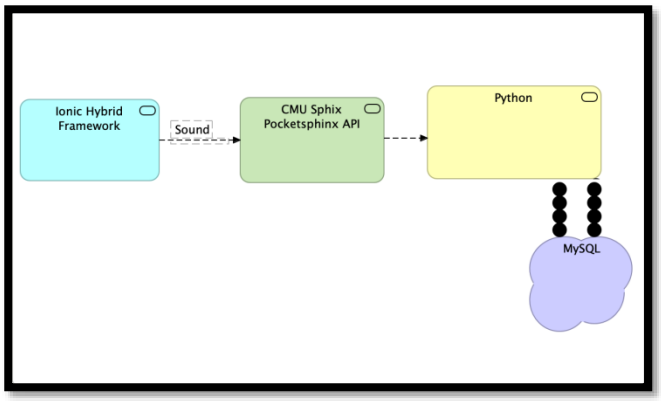

Fig 8. TamilTalk Architecture

To summarize figure 8, the initial design of the web-based application had a MVC design with an Ionic based front end that communicates with a back-end written in Python, while using the PocketSphinx API to convert the sound into text.

4.2 HTML, CSS & Javascript

Due to unforeseen circumstances the original design model was not able to be fully developed. To ensure that a working software was developed for testing another approach had to be adapted. Time constraints placed an urgency on the development process, with this in mind a simple API implementation was used for the application. The Google Web Speech API was the choice of developers

as it allowed for an effienct build of a working web application. This API provides a prebuilt dictionary of Tamil syllables, words, and orthography unique to the four major regions where the language is spoken (India, Sri Lanka, Mayalsia, Singapore).

The main functionality was written in JavaScript, so it allowed easy implantation into a web browser. The graphical user interface was transferred over from the orginal design as it was built using simple HTML and CSS. Several constraints would arise while using the Web Speech API to provide the back-end functionality. The Web Speech API's sever side language is not open source, so ensuring the correct process of conversion from speech to text became a difficult task. Additionally, this API would require all potential users to have an internet connection as well as use the latest version of Google Chrome, as it is the only web browser that works with this API.

5 Testing & Verification

Testing was carried out to help define whether the requirements were achieved and this was implemented in the final stages of the development iteration.

Design protocol testing is imperative in software development as it will help entail how achievable the requirements are. The non-functional requirements were required to be successful and to demonstrate the quality level of the project.

This verification method was conducted by individuals of the team that were not associated with the development of the 'coding aspect' of the software to achieve an unbiased result. Other forms of verification were to monitor the risks associated with the development of the software on a continual basis to measure the success of the application; more risks entailed a higher failure rate.

An important verification method already undertaken was from our sponser, whom of which provided feedback on initial design storyboards and saw the benefits of any initial ideas presented. The feedback was added into the actual design of the application therefor showing the potential that it has and how it can be taken much further in the future.

5.1 System Testing

In the testing phase the application was subjected to a single round of black-box testing. In other words, the user testing the application had no prior knowledge or explanation of the application, only a user manual to aid in navigation.

The testing itself was performed on a MacBook Pro in a classroom at the University of the West of England. The user was given a set of instructions for tasks to be accomplished while researchers observed and recorded the outcomes.

The outcomes of these tasks were measured on a success/failure scale. Four separate test cases were completed during the testing phase with varying tasks for both the application and the user to complete successfully which allowed for testing of the interface design usability and the application functionality at the same time.

5.2 Results

The user was able to successfully navigate the application to accomplish the primary task of speaking into the microphone and the application converting that to Tamil orthography. Usability was high throughout the majority of the application with a few features, such as the copy and paste button, providing the user with problems.

Specific to application performance the test cases provided mixed results. In Test case one, voice-input data, the application was successful in 100% (4 of 4) of tasks tested and 66.67% (4 of 6) of total tasks. Two tasks in test case one were unable to be completed due to testing environment. Test case two showed the application successfully completing 66.67% (2 of 3) of the tasks tested and 50% (2 of 4) of the total tasks. One task of test case two was unable to be tested due to lack of access to API server-side functionality. In test case three the application successfully completed 100% (2 of 2) of the tasks tested and 50% (2 of 4) of the total tasks. Again, the two untested tasked were due to lack of access to server-side API access. Finally, in test case four the application successfully completed 33% (1 of 3) of tasks tested and 25% (1 of 4) of the total tasks. One task was not able to be tested due to testing environment. In total, the application successfully completed 75% (9 of 12) of tasks tested and 47.9% (9 of 19) of the total tasks in the test cases.

6 Discussion

This section looks at the the implications of the results and further development of the application. The results of the testing of this application provide mixed feedback. The main functionality of this application was developed successfully using available industry standard tools. The user was able to speak into the application and receive accurate feedback in Tamil orthography. One of the main requirements of this application was not met, though. When the user would pronounce words or syllables incorrectly, the application would not provide the intended feedback. This application, like many other applications, would predict what it anticipated the user meant to say instead of provided feedback of exactly what they said, even if pronounced incorrectly. Several reasons led to these results such as using a closed source API, development abilities, and development time due to circumstances that arose during the process.

7 CONCLUSION

This paper has attempted to implement speech to text as conceptualized by Ramachandran (2018) and deal with a complex contemporary probem in the field of speech to text and its user acceptance in the Tamil context. In order to provide the best outcomes in future development of Tamil speech to text applicatons we believe that a phonetic dictionary would provide the best results for Tamil speakers. Developing a unique back-end, ideally with the use of open-source API or no API, would ideally allow for the application to achieve the concept of "what you say is what you get". Time constraints, along with development knowledge kept this research from meeting all requirements, but it was shown that provding an application for "minority languages" like Tamil is more readily available than ever.

This project was challenging, a better understanding of the language would be advised in order to fully create a phonetic dictionary for the outcome to be achieved. Users would discover realise errors in their pronunciation organically with the use of such an application as words would not be "corrected" for incorrect pronunciation.

Further work

Future research on the production of this type of application may consider implementing a Dynamic Time Warping (DTW) Algorithm to aid in accurate syllable matching with the database. The DTW algorithm measures the similarities in two sequences, which could vary in speed or distance, and is based on dynamic programming (Yadav et al. 2018). DTW would allow for developers to resolve issues with different rate of speed and different accents in a user's speech. By matching the input speech against the time axis of other syllables in the database, the DTW algorithm could provide the accuracy of actual syllables spoken, intentionally or unintentionally, that this software requires.

REFERENCES

Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., & Vinyals, O. (2010). "Speaker Diarization: A Review of Recent Research". IEEE TASLP. (Available at: http://www1.icsi.berkeley.edu/~vinyals/Files/taslp2011a.pdf)

Ashwell, T. & Elam, J. (2017). "How accurately can Google Web Speech API recognize and transcribe Japanese L2 English learners' oral production?". The JALT CALL Journal, Vol. 13, No.1 pp 59 – 76

Atwood, J. (2009). "Don't Reinvent the Wheel, Unless you Plan on Learning more about Wheels". Coding Horror. (Available at: https://blog.codinghorror.com/dont-reinvent-the-wheel-unless-you-plan-on-learning-more-about-wheels/)

Beat, V. & Novet, J. (2016). "Google says its speech recognition technology now has only an 8% word error rate". Venture beat. (Available at: http://venturebeat.com/2015/05/28/)

Belenko, M.V. & Balakshin, P.V. (2017). "Comparative Analysis of Speech Recognition Systems with Open Code". Mezhdunarodnyiy Nauchno-Issledovatelskiy Zhurnal,)pp 13-18.

Blanken, H., de Vries, A., Blok, H. & Feng, L. (2007). "Multimedia Retrieval (Data-Centric Systems and Applications)". Springer Science & Business Media pp203 – 205

Boyd, C. (2018, January 10). *The Past, Present, and Future of Speech Recognition Technology*. Retrieved from The Startup: https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf

Clarke, S. (2004). "Measuring API Usability". Dr. Dobb's: The World of Software Development. (Available at: http://www.drdobbs.com/windows/measuring-api-usability/184405654)

"CMU-Software Engineering-Faculty-Raj Reddy". Carnegie Mellon. Retrieved 18 August 2011.

CMUSphinx - https://cmusphinx.github.io/wiki/about/

Deng Y. , Li X. , Kwan C.,Raj B.,Stern R. (2007). "Continuous Feature Adaptation for Non-Native Speech Recognition", International Journal of Computer, Information Science and Engineering ,Vol:1 No:6

Devarajan, S. (2009, February 21). Relationship between Japanese and Dravidian (Tamil). Retrieved from http://japanese-dravidian.blogspot.com/2009/01/relationship-between-japanese-and.html

Duran, N. & Battle, S. (2018). "Probabilistic Word Association for Dialogue Act Classification with Recurrent Neural Networks". Boracchi, G., Iliadis, L., Jayne, C. and Likas, A., eds. (2018) Engineering Applications of Neural Networks. Springer, pp. 229-239. ISBN 9783319651729

Fu T., Gao S., Wu X. (2018) Improving Minority Language Speech Recognition Based on Distinctive Features. In: Peng Y., Yu K., Lu J., Jiang X. (eds) Intelligence Science and Big Data Engineering. IScIDE 2018. Lecture Notes in Computer Science, vol 11266. Springer, Cham

Gales, M., Young, S. (2007). "The Application of Hidden Markov Models in Speech Recognition". Foundation and Trends in Signal Processing. P195-304.

Graves, A., Fernandez, S., Gomez. & Schmidhuber, J. (2006). "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland 2Technische Universit¨at M¨unchen (TUM), Boltzmannstr. 3, 85748 Garching, Munich, Germany

Husin, M., Stewart, D., Ming, J. & Smith, G. (2011). "Creating a Spontaneous Conversational Speech Corpus". Advance Publication,

Crystal, D. (2010). *A little book of language*. UNSW Press.

Juang, B.H., & Rabiner, L.R. (2005). "Automatic speech recognition – A brief history of the technology development". Elsevier Encyclopaedia of Language and Linguistics (2nd ed., pp. 806–819).

Kamarudin, M. R., Yusof, M. A. F. M., & Jaya, H. T. (2013). "Low cost smart home automation via Microsoft speech recognition". International Journal of Engineering & Computer Science, pp 6-11.

Keane, E. (2004). Tamil. *Journal of the International Phonetic Association*, *34*(1), 111-116.

Kepuska, V. & Bohouta, G. (2017). "Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx)". Int. Journal of Engineering Research and Application ISSN : 2248-9622, Vol. 7, Issue 3, (Part -2) pp.20-24

Kikel, C. (2018). "What Does Google Speech Recognition API Mean for the Industry". Total Voice Tech (Available at: https://www.totalvoicetech.com/what-does-google-speech-recognition-api-mean-for-the-industry/)

Kreyssig, F. (2018). "Deep Learning for User Simulation in a Dialogue System". University of Cambridge. (Available at: http://mi.eng.cam.ac.uk/~flk24/doc/Thesis-MEng.pdf)

Lacoma, T. (2019). "How to set up speech-to-text in Windows 10". Digital Trends (Available at: https://www.digitaltrends.com/computing/how-to-set-up-speech-to-text-in-windows-10/)

Lange, P. & Suendermann, D. (2014). "Tuning Sphinx to Outperform Google's Speech Recognition API". The Baden-Wuerttemberg Ministry of Science and Arts as part of the research project. (Available at: http://suendermann.com/su/pdf/essv2014.pdf)

Lardinois, F.(2018). "Google 1launches an improved speech-to-text service for developers". Tech Crunch (Available at: https://techcrunch.com/2018/04/09/google-launches-an-improved-speech-to-text-service-for-developers/)

Lo, L. (2012). "Tamil". Ancient Scripts (Available at: http://www.ancientscripts.com/tamil.html)

Manaswi,. N. (2018). "Deep Learning with Applications Using Python". Springer Science + Business Medi, New York, Apres Media pp 135 – 144

Matarneh, R., Maksymova, S., Lyashenko, S., & Belova, N. (2017) "Speech Recognition Systems: A Comparative Review". IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 5, Ver. IV pp71-79

McGuire, N. (2018). "How Accurate is Google Translate?". Argo Translation (Available at: https://www.argotrans.com/blog/accurate-google-translate-2018/)

Pandharipande, R. (2002) "Minority Matters: Issues in Minority Languages in India". International Journal on Mulitcutltural Socieitis (IJMS). Vol 4, No. 2, 2002., p 213-234.

Patel, C. & Kopparapu, S. (2015). "A Multi-criteria Text Selection Approach for Building a Speech Corpus". International Conference on Text, Speech, and Dialogue TSD 2015: Text, Speech, and Dialogue pp 15-22

Pornpanomchai, C., Ngamwongsakollert, P., Tangpitaksame, P. & Wonvattanakij, C., (2012). "Thai-Speech-to-Text Transformation Using Dictionary-Based Technique". Singapore, IACSIT Press, pp. 65-69.

Proffitt, B. (2013). "What APIs are and Why They're Important". Readwrite (Available at: https://readwrite.com/2013/09/19/api-defined)

Rabiner, L. (1989) "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition". Proceedings of the IEEE, 77 (2), pp 257–286

Ramachandran, R., 2018. *Predicting user acceptance of Tamil speech to text by native Tamil Brahmans* (Doctoral dissertation, Sheffield Hallam University).

Samudravijaya, K. & Barol, M. (2003). "Comparison of Public Domain Software Tools for Speech Recognition." ISCA Archive. (Available at: https://www.isca-speech.org/archive_open/wslp_03/wslp_125.pdf)

Sanjay, G. V. et al., (2018). "Dictionay Application With Speech Recognition and Speech Synthesis". International Journal of Advances Research in Computer Science, 9(1), pp. 27-29.

Shulman, D. (2016). *Tamil*. Harvard University Press.

Sloboda, T. & Waibel, A. (1996). "Dictionary Learning For Spontaneous Speech Recognition". International Conference on Spoken Language Processing.

Speech Recognition (Available at: https://developer.mozilla.org/en-US/docs/Web/API/SpeechRecognition)

Srinivasan, A., Srinivasa, K., Kannan, K., & Narashimhan, D. (2009). "Speech Recognition of the letter "zha" in Tamil Language using HMM". International Journal of Engineering Science and Technology Vol.1(2), pp 67-72

Srinivasan, A., Rao, K.S., Kannan, K. & Narasimhan, D. 2010, "Speech Recognition of the letter 'zha' in Tamil Language using HMM",

Srinivasan, A. 2013, "Real time speaker recognition of letter 'zha' in Tamil language", IEEE, , pp. 1..

Steever, S. (1998), "The Dravidian Languages", London: Routledge, pp. 1–39,

Stuttle, M. (2003). "A Gaussian Mixture Model Spectral Representation for Speech Recognition." Cambridge University Engineering

Department. (Available at: http://mi.eng.cam.ac.uk/~mjfg/thesis_mns25.pdf)

Sultana, S., Akhand, M. & Rahman, M. (2012). "Bangla Speech-to-Text Conversion using SAPI". International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia pp 385 – 390

Techseen (2017). "Google Cloud Speech API now supports 30 more languages". (Available at: https://techseen.com/2017/08/14/google-cloud-speech-api-update/)

Thangarajan, R., Natarajan, A.M. & Selvam, M. (2009) "Syllable modeling in continuous speech recognition for Tamil language". International Journal of Speech Technology. (Available at: https://doi.org/10.1007/s10772-009-9058-0)

Thompson, I. (2015, September 29). Tamil. Retrieved from aboutworldlanguages.com: http://aboutworldlanguages.com/tamil

Trebelskis, J. (1995). "Speech Recognition using Neural Networks". CMU-CS-95-142, School of Computer Science, Carnegie Mellon University Pittsburgh, Pennsylvania 15213-3890.

Trivedi, P. (2014). "Introduction to Various Algorithms of Speech Recognition: Hidden Markov Model, Dynamic Time Warping and Artificial Neural Networks". International Journal of Engineering Development and Research. Volume 2, Issue 4.

Twiefel, J., Baumann, T., Heinrich, S., & Wermter, S. (2014). "Improving domain independent cloud-based speech recognition with domain-dependent phonetic post-processing". In Proceedings of the 28th aaai Conference on Artificial Intelligence (aaai-14) pp. 1–7

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P, & Woelfel, J. (2004). "Sphinx-4: A flexible open source framework for speech recognition". Smli, (tr-2004- 139), 1–9. (Available at: http://egouvea.users.sourceforge.net/paper/smli_tr-2004- 139.pdf)

Watanabe, S. (2011). "Bayesian Approaches in Speech Recognition". APSIPA NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan

Web Speech API (Available at: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API)

Writing your own Voice Recognition (Available at: https://www.codeproject.com/Articles/1184834/Writing-Your-Own-Voice-Recognition-Application)

Yu D., Deng L. (2015) "Deep Neural Network-Hidden Markov Model Hybrid Systems". Automatic Speech Recognition. Signals and Communication Technology. Springer, London pp. 99-116

Zvelebil, K. (1992), "Companion studies to the history of Tamil literature". Leiden: Brill, ISBN 978-90-04-09365-2