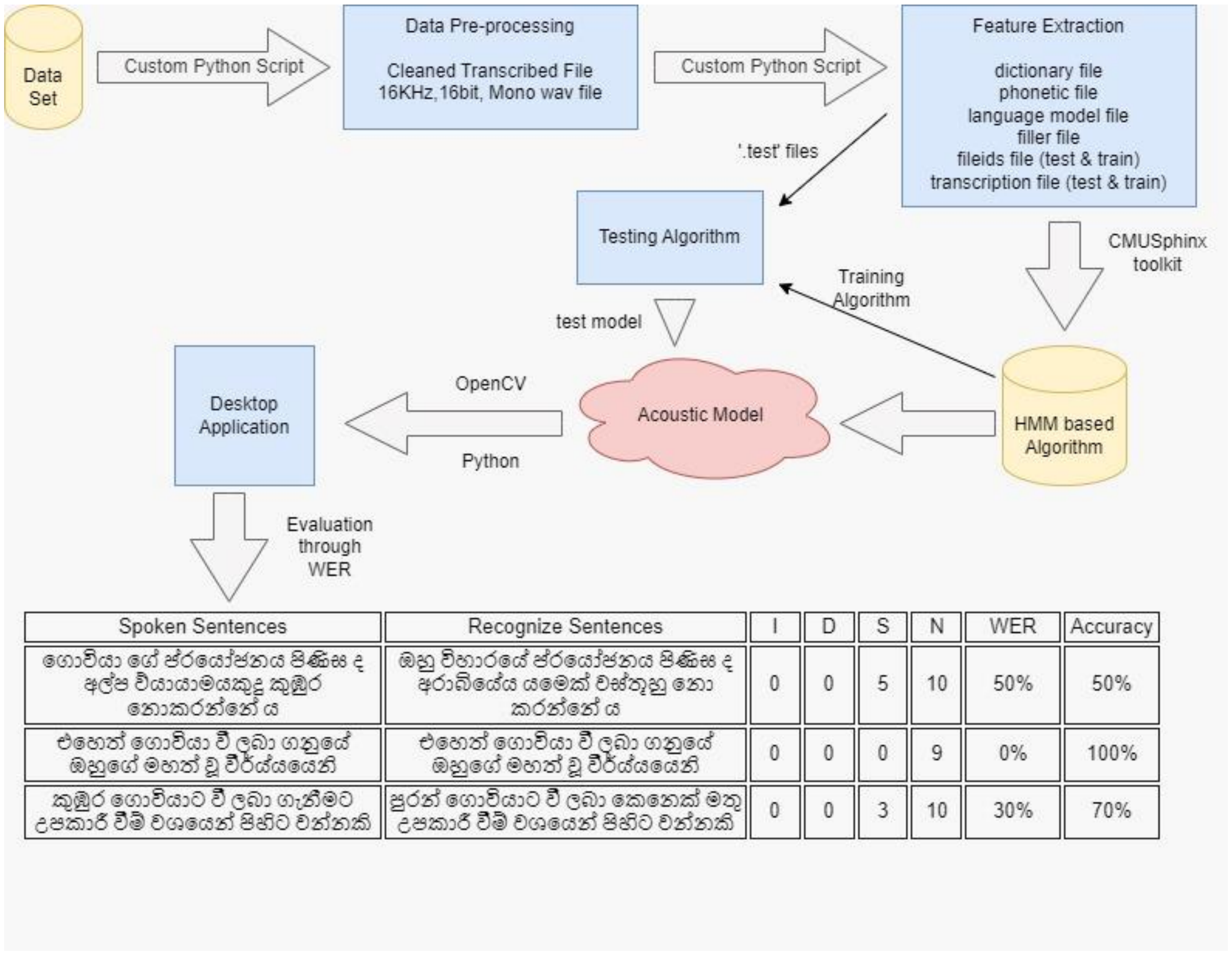
Real-Time Subtitle Generator for Sinhala Speech

R.V.P.S. Akesh and R.G.N. Meegama

Department of Computer Science, University of Sri Jayewardenepura
Gangodawila, Nugegoda, Sri Lanka

Date Received: 27-08-2023 Date Accepted: 27-12-2023



Spoken Sentences	Recognize Sentences	I	D	S	N	WER	Accuracy
ගොවියා ගේ ප්‍රයෝජනය පිණිස ද අල්ප ව්‍යායාමයකුදු කුඹුර නොකරන්නේ ය	ඔහු විහාරයේ ප්‍රයෝජනය පිණිස ද අරාබියේය යමෙක් වස්තුහු නො කරන්නේ ය	0	0	5	10	50%	50%
එහෙත් ගොවියා වී ලබා ගනුයේ ඔහුගේ මහත් වූ වීර්ය්‍යයෙනි	එහෙත් ගොවියා වී ලබා ගනුයේ ඔහුගේ මහත් වූ වීර්ය්‍යයෙනි	0	0	0	9	0%	100%
කුඹුර ගොවියාට වී ලබා ගැනීමට උපකාරී වීම් වශයෙන් පිහිට වන්නකි	පුරන් ගොවියාට වී ලබා කෙනෙක් මතු උපකාරී වීම් වශයෙන් පිහිට වන්නකි	0	0	3	10	30%	70%

**Correspondence: rgn@sjp.ac.lk*
© *University of Sri Jayewardenepura*

Abstract

In today’s digital era, the significance of speech recognition technology cannot be overstated as it plays a pivotal role in enabling human-computer interaction and supporting various applications. This paper focuses on the development of a real-time subtitle generator for Sinhala speech using speech recognition techniques. The CMUSphinx toolkit, an open-source toolkit based on the Hidden Markov Model (HMM), is employed for the implementation of the application. Mel-frequency cepstral coefficients (MFCC) are utilized for feature extraction from the given ’wav’ format recordings. The paper places significant emphasis on the importance of a real-time subtitle generator for Sinhala speech and explores the existing literature in the field. It outlines the objectives of the research and discusses the achieved outcomes. By fine-tuning hyperparameters to enhance the recognition accuracy of the system, impressive results of 88.28% training accuracy and 11.72% Word Error Rate (WER) are attained. The significance of this research is underscored by its methodological advancements, robust performance metrics, and the potential impact on facilitating seamless interactions and applications in the Sinhala speech domain.

Keywords: Speech recognition, Real-time, Subtitle, CMUSphinx, Open source, Hidden Markov Model, Mel-frequency cepstral coefficients, ’wav’, Accuracy, Word Error Rate

1. Introduction

In the contemporary era, the internet and television have evolved into indispensable tools for knowledge acquisition, relying extensively on video content often complemented by subtitles. Subtitles, essential for individuals such as the hearing impaired, non-native speakers, and subtitle enthusiasts, come in two forms: real-time and non-real-time. While non-real-time subtitles are commonly used for pre-recorded content, they prove inefficient for live broadcasts or online meetings. Real-time subtitles, generated instantly through computer algorithms, offer a solution to this limitation, enhancing efficiency for both live and recorded content.

Despite real-time subtitle applications being available for certain languages, Sinhala, a prominent language, lacks such a tool. This absence underscores the need for a real-time Sinhala subtitle generator, particularly beneficial for individuals facing hearing impairments or seeking an immediate and seamless viewing experience. As the aging population often contends with sensory challenges, a real-time Sinhala subtitle generator could prove invaluable, facilitating effortless participation in online meetings and television viewing.

This study addresses the absence of a real-time Sinhala subtitle generator and aims to construct an efficient tool specifically designed for Sinhala speech. The primary objectives are to design and develop a software module for accurate Sinhala voice recognition, to convert recognized Sinhala voice into Sinhala Unicode text script, employing a dedicated database for mapping, to develop a real-time file generation system to save Sinhala text recognized from voice and to create a computer application for seamless integration of Sinhala text with video in real-time.

The application's success hinges on accurately identifying Sinhala alphabet letters and understanding the language's unique structural modifications. Unlike English, Sinhala requires specific modifications for each letter, posing challenges for accurate recognition. Additionally, variations in speakers' voices, influenced by mood and environment, contribute to the complexity of speech recognition, adding to the difficulty of accurate word recognition.

**Correspondence: rgn@sjp.ac.lk*
© University of Sri Jayewardenepura

The literature emphasizes the significance of large vocabulary speech recognition systems, with open-source options like the Sphinx toolkit standing out (Gaida, et al., 2014). Comparative studies between speech recognition engines, such as CMUSphinx and Mozilla DeepSpeech, highlight the importance of performance and resource efficiency (Ramani, Rao, Vidya, & Prasad, 2020). Research also explores advancements in decoder algorithms, including recurrent neural network (RNN) encoder-decoder approaches for continuous large vocabulary speech recognition (Ramani, Rao, Vidya, & Prasad, 2020) (Toshniwal, et al., 2018).

Several projects have focused on Sinhala speech recognition, employing tools like the Sphinx toolkit (Gunasekara & Meegama, 2015). These initiatives range from real-time translation to bidirectional conversion algorithms and web-based speech-to-text translators (Punchimudiyanse & Meegama, Web based automated speechto-text translator for the sinhala language, 2016). Notable efforts include the development of a Sinhala Interactive Voice Response (IVR) system and the exploration of continuous speech recognition using artificial neural networks (Nallathamby, Kariyawasam, Pullaperuma, Vithana, & Jayasena, 2011).

This research aims to contribute to the growing field of speech technologies (Sandasarani, 2015) by developing a real-time Sinhala subtitle generator. The outlined objectives and comprehensive review of related work lay the foundation for advancing accessibility and user experience in Sinhala video content consumption.

2. Methodology

2.1. Introduction to Hidden Markov Model

HMM is a probabilistic graphical model that allows the computation of the joint probability of a set of unknown variables (hidden states) given a set of observed states. The hidden states, also known as latent states, are selected based on the highest probability of the joint sequence. This model relies on the inference based on the Markov process assumption; hence it is referred to as the ‘Hidden Markov Model’. The advantage of this model is that it only requires the present state to predict the future state, without relying on past information.

Typically, ‘states’ refer to the hidden states, while ‘observations’ refer to the observed states. As mentioned earlier, to calculate the joint probability of the sequence of hidden states, three types of information are needed: transition probability (the probability of transitioning to a new state conditioned on a present state), emission probability (the probability of transitioning to an observed state conditioned on a hidden state) and initial state probability (the initial probability of transitioning to a hidden state which can also be looked at as the prior probability).

2.2. Main Theory of the Hidden Markov Model in Speech Recognition Sector

The main purpose of a speech recognition application is to create a text sequence based on input voice. A statistical model like Hidden Markov Model (HMM) can be used for this process. By using HMM, the application can identify hidden states and observable states.

To generate observable states, a voice clip is used, and the features of the voice are extracted using methods like MFCC (Mel Frequency Cepstral Coefficients). These extracted features serve as the observable states of the HMM.

The hidden states in speech recognition are phonemes, which are the basic units of sound in language. HMM transitions between hidden states to find the best phoneme sequence for the extracted features. From this phoneme sequence, relevant word sequences are formed.

Several important models aid in finding the best phoneme sequence. The language model handles word sequences' likelihood, considering factors such as "I write a thesis" being more likely than "Me write a thesis." The pronunciation model converts words to phonemes, while the acoustic model models a sequence of feature vectors based on a sequence of phonemes. These models, together with HMM, contribute to the speech recognition process. Figure 1 shows High-Level architecture for speech recognition that links HMM.

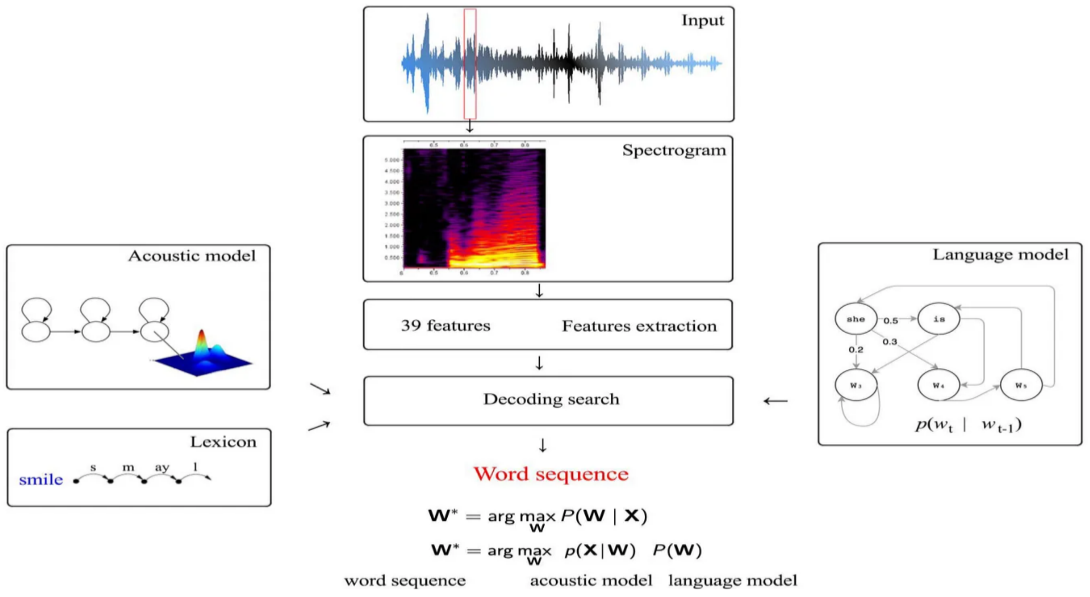


Figure 1: High-Level architecture for speech recognition that links HMM

Using the language model and the acoustic model, the maximum likelihood text sequence can be found:

$$W^* = arg_w max P(X|W)P(W) \tag{1}$$

where P(X|W) is the acoustic model and P(W) is the language model.

The hidden Markov model consists of hidden states and observable states. In Figure 2, the bottom nodes represent the extracted features from the audio clip known as observable states and the top nodes represent hidden states (phones) of HMM. Horizontal arrows show the transition of the phone sequence.

**Correspondence: rgn@sjp.ac.lk*

© University of Sri Jayewardenepura

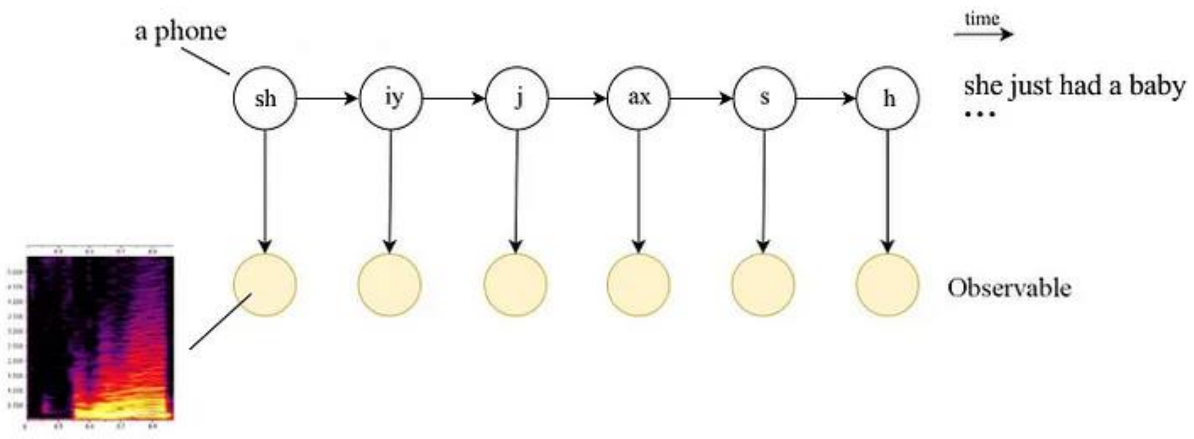


Figure 2: Hidden States and Observable states of HMM

With the HMM model, phone sequence is not searched one by one. Otherwise, when the number of phones increases complexity also grows exponentially. The Viterbi algorithm of HMM helps to find the optimal phone sequence in a polynomial time. This is a good feature of the HMM.

All the above-mentioned model and algorithm works with the Hidden Markov model. Hence it is essential to understand the working procedure of the HMM deeply. The hidden Markov model uses the Markov chain and the first-order Markov chain property (Hidden markov models simply explained, n.d.). However, it is hidden.

As mentioned earlier, there are two types of states of HMM which are hidden states and observable states. Furthermore, there are two types of important probabilities; transition probability, and emission probability. By using these two types of states and probabilities the HMM model makes three important algorithms; namely, the forward algorithm, Viterbi algorithm, and forward-backward algorithm. Understanding these three algorithms is very helpful to get an idea of the application of the hidden Markov model in speech recognition.

1) Forward Algorithm:

The likelihood of observation is calculated by the forward algorithm. The objective is to get the summation of probabilities for all the possible state sequences of an observation. For this calculation, transition probability and emission probability are used.

$$P(X) = \sum_s P(X,S) = \sum_s P(X|S)P(S) \quad (2)$$

where, X : observed events, $\sum_s P(X,S)$: sum over all possible time sequences of internal states, P(X|S) : calculated from emission probability, P(S) : calculated from transition probability

Getting the summation of all possible state sequences, one at a time, has an exponential complexity with the number of states. This problem can be solved with HMM smartly. Since HMM solves the problem at time 't' with the result of time 't-1' or 't+1'. Hence, by expressing the calculations with time recursively, the exponential curse can be broken.

2) *Viterbi Algorithm:*

Then it is essential to find internal states to a given sequence of observations. This can be done using the Viterbi algorithm in HMM. This process is also called decoding. In speech recognition using this algorithm, HMM finds out the most possible phoneme sequence.

Since the current observation only depends on the current state, the joint probability of the observation sequence with the optimal state can be expressed as,

$$V_t(j) = max_{i=1}^{N} V_{t-1}(i) a_{ij} b_j(x_t) \tag{3}$$

After the comparison of the equation of the forward algorithm and the above equation, the summation function of the forward algorithm is replaced by the maximum function of the Viterbi algorithm. That means instead of summing all possible state sequences here Viterbi algorithm takes the most likely path of possible state sequence.

3) *Forward Backward Algorithm:*

Forward-backward algorithm is used to learn the hidden Markov model. This algorithm is an iterative procedure that estimates the maximum likelihood parameters of the HMM based on the observed data. HMM consists of a set of hidden states and observable states, along with the transition probabilities between states and emission probabilities associated with the observable states. The goal of the Baum-Welch algorithm is to find the optimal parameters that maximize the likelihood of the observed data.

The backward algorithm is the complement of the forward algorithm. The probability of seeing all the coming observations is calculated as a forward algorithm but in reverse order.

$$P(Y|\lambda) = \sum_{j=1}^{N} \pi_j \, b_j(y_1) \beta_j(1) \tag{4}$$

4) *Acoustic Model:*

The acoustic model is an important component in the flow of speech recognition using the Hidden Markov model. This model represents the acoustic properties of speech sounds. This is responsible for mapping the audio input to a sequence of phonetic units. This model is typically trained with a large amount of labeled speech data and this model is capable of capturing statistical characteristics of speech sounds. Model parameters like mean vectors covariance matrices all are estimated based on speech features.

The Gaussian Mixture Model (GMM) is a type of acoustic model used in speech recognition systems. This type of acoustic model uses Mel-frequency cepstral coefficients (MFCCs) extracted features as the input. GMM consists of a mixture of Gaussian distributions.

Using an acoustic model, the likelihood of an audio feature vector X given the phone P (X phone) can be determined. Suppose that there is only one feature for each audio frame. Then the

**Correspondence: rgn@sjp.ac.lk*

© University of Sri Jayewardenepura

value of this feature can be modeled by using a normal distribution. Here given different phones, corresponding probability density values can be calculated and then classified as the phone with the highest value. However, real-world scenarios are very different from this situation. Since this acoustic model gets inputs from the MFCC, there are 39 features of an audio frame; hence, given the state the value of the features cannot be modeled from the normal distribution.

Therefore, the Gaussian Mixture Model is suitable. That means a few possible values are allowed since the flexibility of variants in speech is provided.

By considering near one of the m nodes, the feature value of a specific phone can be obtained. Sometimes, there might be values more likely than others hence, to indicate more likely values, weights are introduced (Speech recognition — gmm, hmm, n.d.).

5) MFCC:

In the computation of MFCC, the first thing is windowing, windowing is splitting the speech signals into frames. High frequencies are prioritized in order to achieve similar amplitude for all formants because high-frequency formants process less amplitude than low-frequency formants do. After the widowing process, the next step is applying the Fast Fourier Transform (FFT) to determine each frame's power spectrum. Subesequently, the filter bank processing is done using Mel-scale on the power spectrum. To get the MFCC coefficients, the power spectrum is converted to a log domain before the DCT (Discrete Cosine Transform) is applied to the speech signal.

It is possible to compute formants that are in the low- frequency range and represent the resonances of the vocal tract using MFCC since it can effectively denote the low- frequency zone more efficiently than the high-frequency region. Additionally, it is an ideal representation of sound when the source qualities (music and voice) are constant and reliable, Additionally, it can record data from sampled signals up to a maximum frequency of 5 kHz, which encompasses most of the energy in human-generated sounds. The block diagram shown in Figure 3 summarizes and shows the steps to obtain the needed coefficient.

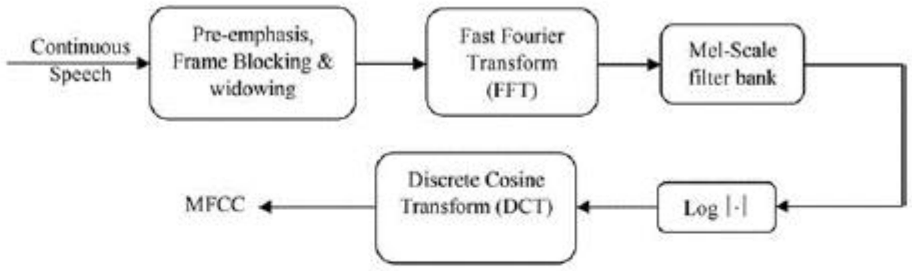

Continuous Speech
Pre-emphasis, Frame Blocking & widowing
Fast Fourier Transform (FFT)
Mel-Scale filter bank
Log |·|
Discrete Cosine Transform (DCT)
MFCC

Figure 3: Process of extracting features from speech by MFCC

2.3. Method

CMUSphinx toolkit, which is an open-source collection of speech recognition tools and libraries heavily relies on Hidden Markov model is used to make a Realtime Sinhala subtitle generator for Sinhala speech.

**Correspondence: rgn@sjp.ac.lk*

© University of Sri Jayewardenepura

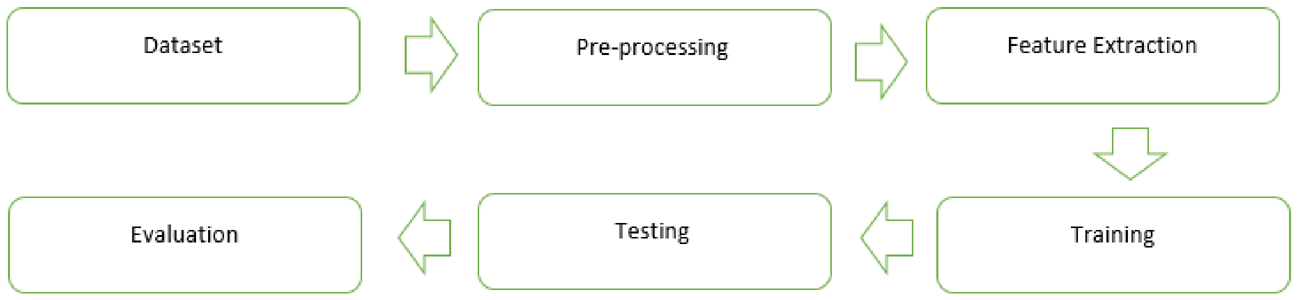


Figure 4: Main flow of the method

1) *Dataset:*

Here, the selected data set consists of nearly 3300 Sinhala voices and a transcribed file for each voice. Those voices are in both ’split-flac’ and ’wav’ format. This data set is created on behalf of the training algorithm for speech recognition hence the recordings are very clear and high quality. All the records are recorded by a single speaker. Specially, there are some recordings of utterances of Sanskrit and Pali. There could be some errors like the recording not matching with the transcribed file. Some statistics of the data set (Foundation, 2021) are;

- Number of recordings – 3300
- Total Length - 7 hours 5 minutes
- Maximum length – 33.4 seconds
- Minimum length – 1 seconds
- 'wav' -format – sample rate 22050Hz and 16-bit PCM encoded
- Silences have been removed from both the beginning and end

2) *Data Pre-processing & Feature Extraction:*

The previously selected data set contains ‘wav’ files with sample rates 22050 and 16-bit. However, for the development of the acoustic model using the CMUSphinx toolkit it is required to have all the recordings in ‘wav’ format with sample rate -16Khz, 16bit mono. Hence, at the pre-processing stage, must convert all the ‘wav’ files in the data set into the required format. Furthermore, the file name of the recordings is not in a specific format. This might be a problem in the future while creating other required files. Hence, a new Python script is written to rename all the converted recordings.

In the selected data set, the transcribed file contains all the utterances which are recorded. However, there are some specific symbols (! , ., ?), and sometimes numbers are included in the sentence, All these symbols and numbers are removed by a separate new Python script. The output is a cleaned transcribed file. Features are extracted from both converted ‘wav’ files and cleaned transcribed file.

As previously mentioned MFCC method (2.2. 5)) is used here to extract features from ‘wav’ files. Those extracted features are taken as inputs for the creation of a new acoustic model.

Then using a cleaned transcribed file, specific types of files must be created, which are essential for the creation of an acoustic model according to CMUSphinx documentation which are,

1) '.dic' file
2) '.phone' file
3) '.lm.bin' file
4) '.filler' file
5) ' train.fileids' file
6) ' train.transcription' file
7) ' test.fileids' file
8) ' test.transcription' file

Since the amount of data set is very large, manual creation of the above files is a tedious task. Hence, new Python scripts are written to automate this process.

The flow chart shown in Figure 5 shows how data pre- processing and feature extraction are done by using Python scripts which are mentioned above (python scripts are shown by numbers in the flow chart; *eg:- by running python script 1 on the original transcribed file output is a cleaned transcribed file).

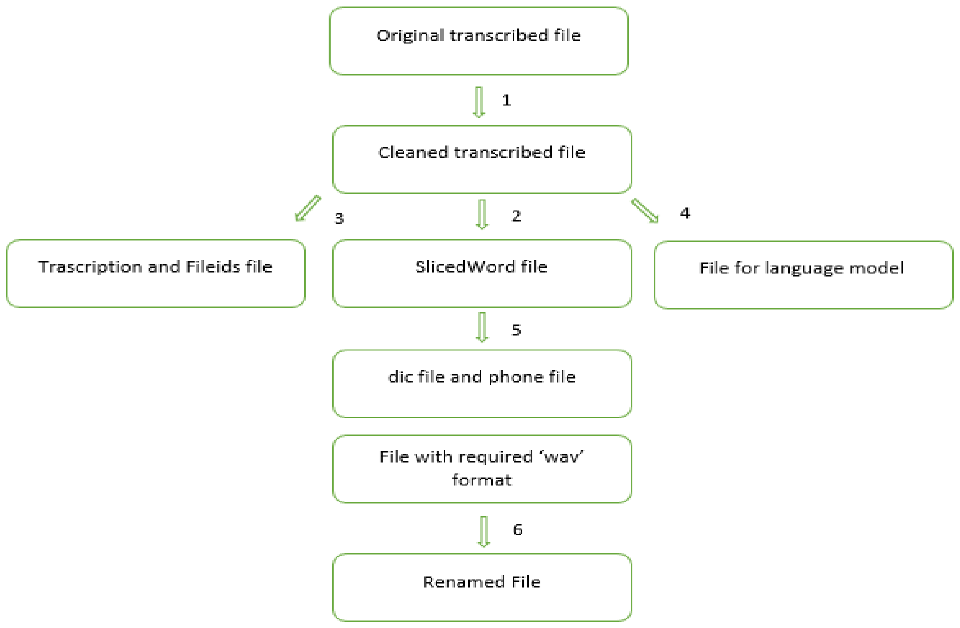


Figure 5: Process of Data Pre-processing and Feature Extraction

3) Training:

At the training stage, using the previously described forward-backward algorithm (2.22 3)), a new acoustic model is created.

4) Testing:

Using the newly created acoustic model and generated language model, the testing stage is done previously described Viterbi algorithm (2.2. 2)) according to the 'test.transcription' and 'test.fileids' files. After this process ".align" file is created which contains the overall results of the testing.

**Correspondence: rgn@sjp.ac.lk*

© University of Sri Jayewardenepura

Thereafter, using the acoustic model, a realtime subtitle generator is created using OpenCV and PocketSphinx library in Python. This created application recognizes Sinhala voices and converts them into Sinhala text and feeds those texts into a video while all the recognized texts are saved in a separate file.

5) Evaluation:

The evaluation part is done using the accuracy of speech recognition. The accuracy of speech recognition systems is typically measured using metrics such as Word Error Rate (WER). WER measures the percentage of words that are incorrectly recognized compared to the 'test.transcription'.

3. Results

In this study, accuracy has been used as a performance matrix. Since this is a subtitle-generating model, basically the accuracy is defined by evaluating its performance against a set of reference subtitles. In other words, measuring its ability to generate correct and meaningful subtitles for a given input.

The Word Error Rate (WER) is a matrix used in speech recognition and transcription tasks to evaluate accuracy. WER measures the difference between the generated subtitles and the reference subtitles at the word level. WER calculates the minimum number of word insertions, deletions, and substitutions needed to transform the generated subtitles into reference subtitles (Papineni, Roukos, Ward, & Zhu, 2002), (Mpora, Ganchev, Kotinas, & Fakotakis, 2007). Lower WER values indicate higher accuracy (Srun, Ny, Cheat, Ching, & Ny, 2021).

$$WER = \frac{I+D+S}{N} \tag{5}$$

$$\text{Accuracy} = 100 - \text{WER} \tag{6}$$

where, I: Number of incorrectly inserted words, D: Number of undetected words, S : Number of substituted words between reference & recognized script, N: Number of counted words in transcription

4.1 Training Result

Without doing any configuration for the default configuration file, the following results can be recorded (Table 1). The '_train' transcription' file consists of 3300 sentences, and from them, 20% (main dataset) of sentences are selected for the '_test.transcription' file. Since, the tested file contains approximately 660 sentences, in other words, it has 7882 words approximately. Table 1 shows the obtained training accuracy from the main dataset without doing any modification for the default configuration file. According to that, accuracy is 67.05% and WER is 32.95% for 7882 words.

Table 1: Results with Default Configuration for Main Dataset

WER	Accuracy	Number of Words
32.95%	67.05%	7882

4.2 Tunning Hyperparameter

The principal goal of machine learning, including subtitle-generating models is to create a model that performs well and gives accurate predictions in a particular set of cases. In order to achieve that, machine learning optimization is needed. Machine learning optimization is the process of adjusting hyperparameters in order to minimize the cost function. In other words, the training process typically involves optimizing model parameters and hyperparameters using training data. The model is trained to maximize the likelihood of generating the correct subtitles given the input data. Hyperparameters are configuration settings that are not learned from data but are set before starting to train the model. When it comes to hyperparameter optimization for a subtitle generation model, several hyperparameters were tuned manually to improve the performance of the model.

Changed variables and description:

- Liftering coefficient - It is used to apply liftering to MFCCs during feature extraction which involves applying a windowing function to the MFCC. The value of this parameter determines the degree of emphasis or attenuation on the spectral components, with higher values improving the discrimination of specific phonetic features potentially leading to better recognition accuracy.
- Phonetic decision tree clustering - Enabling this feature multiple Gaussian context-independent models can be created. This technique is used in speech recognition to enhance the accuracy and performance of the system. This trains separate models for each phoneme. By training a dedicated Gaussian model for each phoneme the system captures acoustic characteristics more accurately.
- State tying - This determines the number of tied states used in the acoustic modeling process. These states allow for efficient modeling by sharing statistical information among similar acoustic units such as phonemes. This defines the granularity of the acoustic model Higher values lead to more distinct tied states. Experimentation is necessary to determine the suitable value because it significantly impacts system accuracy.
- Number of Gaussians - It determines the number of Gaussian components used to model the acoustic properties of each tied state or acoustic unit which can be changed for the continuous models for large vocabulary. It can be any power of 2, allowing flexibility in choosing the appropriate number of Gaussians for the specific task or dataset.

Since hyperparameter optimization can be time-consuming and computationally expensive, it is advisable to use techniques like early stopping and parallelization to make the process more efficient. Out of those techniques, here early stopping technique was used.

The idea behind early stopping is to monitor the model's performance on a separate validation dataset during training and stop the training process when the performance on the validation set starts to deteriorate. Here, 10% of the training dataset was taken as the validation set, and have been tuned the hyperparameters manually and tried to increase the accuracy of the model. Table 2 shows the obtained results before beginning the hyperparameter tuning for the validation set. 78.81% of accuracy can be obtained with 21.19% WER.

Table 2: Results with Default Configuration for Validation Set

WER	Accuracy
21.19%	78.81%

The outcomes of changing the Liftering coefficient's value are shown in Table 3. Interestingly, regardless of the adjustments made to the coefficient, the accuracy values consistently revolve around 78.69%, which corresponds to the accuracy achieved using the default configurations. According to these results, altering the liftering coefficient does not result in appreciable increases in accuracy. In other words, it seems that the liftering coefficient does not significantly affect the system's overall accuracy.

Table 3: Obtained Results from Validation Set by Only Changing the Lifting Coefficient

Value	WER	Accuracy
12	21.19%	78.81%
22(Default)	21.31%	78.69%
30	21.31%	78.69%
35	20.94%	79.06%
45	21.19%	78.81%
60	21.19%	78.81%
70	21.19%	78.81%
90	21.19%	78.81%
100	21.31%	78.69%
110	21.31%	78.69%
130	20.82%	79.18%

Table 4: Obtained Results from Validation Set by Changing the Number of Gaussians

Value	WER	Accuracy
8(Default)	21.19%	78.81%
16	21.19%	78.81%
32	21.19%	78.81%
64	21.19%	78.81%
128	21.19%	78.81%

The results of changing the number of Gaussians employed in the system are shown in Table 4. The accuracy was 78.81% with the default value of 8 which remained constant regardless of the number of Gaussians examined. This finding implies that the overall accuracy is not much affected by the number of Gaussians used in an iteration.

*Correspondence: rgn@sjp.ac.lk

© University of Sri Jayewardenepura

Table 5: Obtained Results from Validation Set by Changing the Number of Tying States

Value	WER	Accuracy
100	21.31%	78.69%
200(Default)	21.19%	78.81%
250	21.31%	78.69%
300	21.31%	78.69%
350	21.31%	78.69%
400	21.31%	78.69%
4200	21.31%	78.69%

Table 5 shows the outcomes of modifying the number of tie states, another hyperparameter intended to increase the system's accuracy. The best accuracy was obtained at 78.81% with the default value of 200. Intriguingly, the accuracy remained marginally lower at 78.69% for all other entries where the number of tie states was changed.

The results achieved by allowing the phonetic decision tree clustering parameter, which enables the training of multiple Gaussian models, are shown in Table 6. This method resulted in a validation set accuracy of 93.56%, which is outstanding. When compared to the default accuracy number of 78.81%, this appreciable increase in accuracy is noticeable. It is clear from the findings shown in the previous tables that turning on the phonetic decision tree clustering parameter has a noticeable impact on accuracy. The accuracy, however, does not seem to be significantly impacted by the other variables. According to the findings, changing parameters like the liftering coefficient, the number of Gaussians, or the number of tying states does not significantly affect the accuracy. Since the phonetic decision tree clustering parameter is enabled for the main data set the result in Table 7 were obtained.

Table 6: Obtained Results from Validation Set by Enabling Phonetic Decision Tree Clustering Parameter

	WER	Accuracy
Multiple Gaussians context	6.44%	93.56%
Single Gaussian context	21.19%	78.81%

Table 7: Obtained Results from Main Dataset by Enabling Phonetic Decision Tree Clustering Parameter

WER	Accuracy	Number of Words	Number of Sentences
11.72%	88.28%	7882	660

The results of using the phonetic decision tree clustering parameter for the primary data set are shown in Table 7. It indicates that an outstanding accuracy of 88.28% can be obtained by training several Gaussian context-independent models. A single Gaussian context-independent model, however, produces a much lower accuracy of 67.05%.

Furthermore, to assess whether training accuracy is influenced by the specific choice of the test data set, various 'test.transcription' files were generated. It is important to note that all of these alternative files contain an equal number of words compared to the original 'test.transcription' file. By maintaining an equal word count, the impact of the test data set itself can be isolated and analyzed independently from other factors. This approach allows for a fair and unbiased evaluation of the training accuracy across different test data sets (Table 8).

Table 8: Results for Main Dataset by Changing the 'test.transcription' File

Flod	WER	Accuracy	Number of Sentences
Flod 1	32.95%	67.05%	660
Flod 2	26.96%	73.04%	588
Flod 3	36.91%	63.09%	433
Flod 4	33.33%	66.67%	445
Flod 5	28.28%	71.72%	622

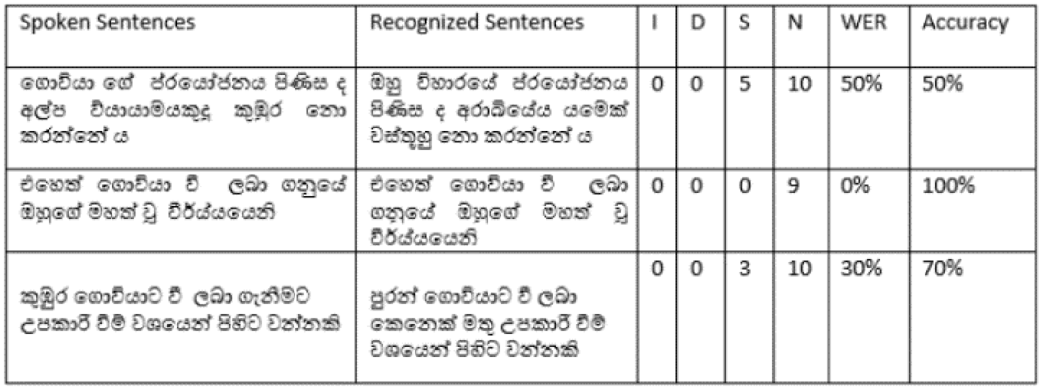
According Table 8, variations in training accuracy were seen despite keeping the number of words in the chosen dataset constant. This suggests that occasionally certain terms in the selected data set are misclassified or in- correctly identified by the trained model. Despite the constant word count, the accuracy varies, indicating that the model can have trouble correctly identifying and understanding particular terms.

The methods discussed aim to improve the training accuracy of the acoustic model. Consequently, the acoustic model with the highest training accuracy (88.28%) was chosen for developing a real-time subtitle generator.

4.3 Testing Result

Using the acoustic model with the highest training accuracy for the real-time subtitle generator, the following test results have been obtained.

Table 9: Results of Realtime Application using the above acoustic model



Spoken Sentences	Recognized Sentences	I	D	S	N	WER	Accuracy
ගොවියා ගේ ප්රයෝජනය පිණිස ද අල්ප ව්යායාමයකුදු කුඹුර නො කරන්නේ ය	ඔහු විහාරයේ ප්රයෝජනය පිණිස ද අරාබියේය යමෙක් වස්තූහු නො කරන්නේ ය	0	0	5	10	50%	50%
එහෙත් ගොවියා වී ලබා ගනුයේ ඔහුගේ මහත් වූ වීර්ය්යයෙනි	එහෙත් ගොවියා වී ලබා ගනුයේ ඔහුගේ මහත් වූ වීර්ය්යයෙනි	0	0	0	9	0%	100%
කුඹුර ගොවියාට වී ලබා ගැනීමට උපකාරී වීම් වශයෙන් පිහිට වන්නකි	පුරන් ගොවියාට වී ලබා කෙනෙක් මතු උපකාරී වීම් වශයෙන් පිහිට වන්නකි	0	0	3	10	30%	70%

The test results obtained from the real-time subtitle generator application in a real-world setting are represented in Table 9. For testing purposes, three Sinhala sentences were spoken to the system. The second column displays the recognized Sinhala sentence based on the input provided. The variables I, D, S, and N, stand for Insertions, Deletions, Substitutions, and Total number of words respectively, and are represented in their respective columns.

**Correspondence: rgn@sjp.ac.lk*
© University of Sri Jayewardenepura

The accuracy percentages obtained for the spoken sentences were 50%, 100%, and 70%. Taking the average of these accuracy values, the overall accuracy achieved is 73.33%. These results reflect the performance of the real-time subtitle generator in accurately transcribing the spoken Sinhala sentences. While there is variability in the accuracy for different sentences, the average accuracy indicates the system's ability to capture the content of the spoken words and provide corresponding subtitles.

4. Discussion

The initial accuracy of the acoustic model, without any modifications to the default configuration, was 67.05%. To optimize the training accuracy, parameter tuning was performed using a validation set, which identified influential parameters. Using the values of these parameters, a new acoustic model was trained, resulting in a higher training accuracy of 88.28%.

An experiment was conducted to investigate the impact of the selected data set for testing on training accuracy. The results showed that the average accuracy varied around the default accuracy value. The experiment revealed challenges in correctly identifying and comprehending specific terms within the training data set, leading to varying training accuracies when tested with different 'test.transcription' files.

These findings emphasize the importance of meticulous parameter tuning and data selection for achieving optimal training accuracy in acoustic models. The experiments highlighted the influence of specific parameters and the model's ability to interpret and recognize various terms in the training data.

The developed application, using the acoustic model with a training accuracy of 88.28%, was tested in a real-world environment, achieving an overall accuracy of 73.33%. Considering the challenges posed by real-world conditions such as background noise, this accuracy percentage is considered fair and acceptable. Despite the difficulties in mitigating these effects, the system demonstrates a reasonable level of accuracy in transcribing and interpreting speech.

The current limitation of the subtitle generator lies in its exclusive reliance on the provided dataset for subtitle generation. A crucial enhancement would involve expanding the dataset to encompass recordings from multiple speakers, thereby significantly augmenting the generator's adaptability to diverse speech patterns. Furthermore, the application boasts cross-platform independence, ensuring its compatibility across various operating systems. To further refine its capabilities, a key improvement would be the seamless and continuous creation of subtitles in real-time. These advancements collectively position the real-time subtitle generator as a versatile and inclusive tool, extending its utility to a broader user base and diverse scenarios.

During the development of the Real-time Subtitle Generator for Sinhala speech, several limitations were encountered. Building the language model proved challenging as it required a Unix system, whereas installing a relevant toolkit for Windows had limited documentation. To overcome this, language model files were created using a Unix system, while the rest of the development was done in a Windows environment.

Outdated pages in the CMUSphinx documentation presented another obstacle, leading to the utilization of an older version of the toolkit. Newer versions proved challenging due to insufficient instructions and guidance.

Additionally, there was a scarcity of data sets for Sinhala speech and transcription. One identified data set had significant background noise, so an alternative data set with reduced noise levels was found and Python scripts were modified accordingly. This new data set featured recordings from a single speaker.

*Correspondence: rgn@sjp.ac.lk

© University of Sri Jayewardenepura

These limitations highlight the complexities faced during the development process. However, appropriate solutions were devised to address issues related to toolkit compatibility, documentation, and the availability of suitable data sets.

5. Conclusion

This research aimed to develop a real-time subtitle generator for Sinhala speech using speech recognition techniques. By employing methods like MFCC for feature extraction and Hidden Markov Models for statistical modeling, a new acoustic model with a training accuracy of 88.28% was created. This model formed the foundation for real-time subtitle generation of Sinhala speeches.

The application demonstrated satisfactory accuracy in real-world scenarios, showcasing its practical viability. The significance of this study lies in its novelty, as there were no existing offline real-time subtitle generators for Sinhala speech before this work. This development opens up possibilities for further advancements in Sinhala speech processing and the digital era. While the real-time subtitle generator accurately generates Sinhala subtitles once it identifies speech sentences, it does have disadvantages such as, inability to generate new subtitles beyond those provided in the dataset, being limited to recognizing and transcribing voices present in the training dataset and being limited to a single speaker. These disadvantages highlight the need for future improvements and adaptations to expand the generator's applicability to a wider range of voices and subtitles.

Several suggestions for further research can enhance the capabilities of the real-time subtitle generator.

- The generator is currently limited to generating subtitles from the provided dataset, expanding the dataset and including recordings from multiple speakers would be a valuable direction for future research.
- The previous experiments have indicated that the model may face challenges in accurately recognizing and interpreting specific words. Investigating these variations and exploring methods to improve the model's accuracy in such cases would present an interesting avenue for further research.
- The sensitivity of the toolkit to background noises poses a significant limitation. Exploring techniques to reduce the sensitivity to noise and enhance the model's robustness in real-world environments would be a promising area of research.

By addressing these areas, future research endeavors can contribute to the advancement of the real-time subtitle generator, enabling it to generate subtitles for a wider range of content, improve word recognition accuracy, and enhance noise robustness for more reliable performance in various environments.

References

Crabb M, J. R. (2015). Online news videos: The ux of subtitle position. 215-222.

De Castro, M., Carrero, D., Puente, L., & Ruiz, B. (2011). Real-time subtitle synchronization in live television programs. *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, (pp. 1 - 6).

Downey, G. (2008). Closed captioning: Subtitling, stenography, and the . *JHU Press*.

Foundation, P. N. (2021). *Path nirvana sinhala tts dataset*. Retrieved from https://github.com/pnfo/sinhala-tts-dataset/

Gaida, C., Lange, P., Petrick, R., Proba, P., Malatawy, A., & Suendermann-Oeft, D. (2014). Comparing open-source speech recognition toolkits. *11th International Workshop on Natural Language Processing and Cognitive Science.*

Gunarathne, W., Ramasinghe, T., Wimalarathne, D., Balasuriya, B., & Hettige, B. (2017). Sinhala speech to text library using sphinx. *2017 KDU IRC.*

Gunasekara, M., & Meegama, R. (2015). Real-time translation of discrete sinhala speech to unicode text. *Fifteenth International Conference on Advances in ICT for Emerging Regions(ICTer)* (pp. 140-145). IEEE.

Hidden markov models simply explained. (n.d.). Retrieved 05 21, 2023, from https://towardsdatascience.com/hidden-markov-models-simply-explained-d7b4a4494c50#:∼:text=Intuition%20and%20Example%20Model,through%20its%20given%20observed%20states/.

Lu , L., Zhang, X., Cho, K., & Renals, S. (2015). A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. *Sixteenth Annual Conference of the International Speech Communication Association.*

Mpora, I., Ganchev, T., Kotinas, E., & Fakotakis, N. (2007). Examining the influence of speech frame size and number of cepstral coefficients on the speech recognition performance. *12th International Conference on Speech and Computer*, (pp. 1-6).

Nallathamby, J., Kariyawasam, K., Pullaperuma, H., Vithana, D., & Jayasena, S. (2011). Debasa sinhala interactive voice response system.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation. *40th annual meeting of the Association for Computational Linguistics*, (pp. 311-318).

Punchimudiyanse, M., & Meegama, R. (2015). Unicode sinhala and phonetic english bi-directional conversion for sinhala speech recognizer. *IEEE 10th International Conference on Industrial and Information Systems (ICIIS)* (pp. 296-301). IEEE.

Punchimudiyanse, M., & Meegama, R. (2016). Web based automated speechto-text translator for the sinhala language. *National Information Technology Conference (NITC), Colombo, Sri Lanka.*

Ramani, A., Rao, A., Vidya, V., & Prasad, V. B. (2020). Automatic subtitle gen- eration for videos. *6th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 132-135). IEEE.

Sandasarani, N. (2015). Sinhala speech recognition. *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERI), 04*(10).

Speech recognition — gmm, hmm. (n.d.). Retrieved 05 22, 2023, from https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196/.

Srun, C., Ny, V., Cheat, C., Ching, S., & Ny, P. (2021, 08). Development of speech recognition system based on cmusphinx for khmer language. *International Journal of Innovative Research in Science Engineering and Technology, 6*, 770-775.

Toshniwal, S., Kannan, A., Chiu, C. C., Wu, Y., Sainath, T. N., & Livescu, K. (2018). A comparison of techniques for language model integration in encoder-decoder speech recognition. *2018 IEEE spoken language technology workshop (SLT)* (pp. 369-375). IEEE.

*Correspondence: rgn@sjp.ac.lk

© University of Sri Jayewardenepura