Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 5107–5113
Marseille, 20-25 June 2022

A Systematic Approach to Derive a Refined Speech Corpus for Sinhala

Disura Warusawithana, Nilmani Kulaweera, Lakshan Weerasinghe, Buddhika Karunarathne
Department of Computer Science and Engineering
University of Moratuwa
Sri Lanka
{disurawaru.17, rukshilakulaweera.17, lakshan.17}@cse.mrt.ac.lk, buddhikak@uom.lk

Abstract

Speech Recognition is an active research area where advances of technology have continuously driven the development of research work. However, due to the lack of adequate resources, certain languages such as Sinhala, are left to underutilize the technology. With techniques such as crowdsourcing and web scraping, several Sinhala corpora have been created and made publicly available. Despite them being large and generic, the correctness and consistency in their text data remain questionable, especially due to the lack of uniformity in the language used in the different sources of web scraped text. Addressing that requires a thorough understanding of technical and linguistic particulars pertaining to the language, which often leaves the issue unattended. We have followed a systematic approach to derive a refined corpus using a publicly available corpus for Sinhala speech recognition. In particular, we standardized the transcriptions of the corpus by removing noise in the text. Further, we applied corrections based on Sinhala linguistics. A comparative experiment shows a promising effect of the linguistic corrections by having a relative reduction of the Word-Error-Rate by 15.9%.

Keywords: speech recognition, Sinhala speech corpora, refined corpus

1. Introduction

With the advent of artificial intelligence (AI) in computer science, the conventional techniques of human-computer interaction (HCI) have been redefined with more natural human behaviours. Among such techniques, conversational interfaces driven by speech recognition have gained great attention, since speech is the most natural way of human communication (Gunasekara and Meegama, 2015). Since the 1950s (Malik et al., 2021), research and development in this area has evolved over the years and has produced many applications powered by the speech recognition technology. Examples of such applications include virtual assistants such as Google Assistant, Apple Siri, and Amazon Alexa (Karunathilaka et al., 2020) which are widely used in everyday life.

A typical automatic speech recognition (ASR) system comprises three main models, namely, acoustic model, language model, and pronunciation model. The models use a probabilistic approach to collaboratively decode a sequence of text that matches to a sequence of features extracted from the audio input. Having an abundance of resources, widely used languages such as English are leading in the development of speech recognition. On the contrary, languages used by relatively smaller populations remain as low-resourced in this field, making them still remain in their early stage of development. Sinhala, a language used by the majority of the population in Sri Lanka, is recognized as such a language (de Silva, 2021; Dilshani et al., 2018). Being a member of the Indo-Aryan language family, Sinhala is a phonetically rich language and has a large lexical diversity (Karunathilaka et al., 2020). Therefore, developing an ASR system for Sinhala with competitive performance is a challenging task.

In this paper, we discuss our approach of deriving a refined corpus for Sinhala speech recognition. The rest of the paper is organized as follows. In section 2, we discuss work related to Sinhala ASR. We formalize our motivation in this project in section 3. Section 4 contains the specifications of the corpus[1] (hereinafter referred to as 'OpenSLR-52') which we have used in this project and elaborates on the issues we have identified in that corpus. The systematic approach we have followed to address those issues is described in section 5. Using the open-source Kaldi ASR toolkit[2], we have conducted an experiment to see the effectiveness of our approach. The setup and results of the experiment are explained in section 6. Finally, in section 7, we conclude our discussion highlighting the advantages of our approach.

2. Related Work

Despite Sinhala being a low-resourced language, speech recognition for Sinhala can be noticed as an active and rapidly growing research domain. There are several examples that can be noted as attempts to develop ASR systems for the Sinhala language. Discrete speech recognition systems which can recognize isolated words, have been implemented by Amarasinghe and Gamini (2012), and Gunasekara and Meegama (2015). An initiative to develop continuous speech recognition systems for Sinhala has been taken by Nadungodage and Weerasinghe (2011). In that, they have generated a speech corpus of 106 sentences, but with a single speaker. However, it is appreciable that they have given attention to important aspects of the corpus such as phonetic balance. Their transcriptions have been prepared using content from newspaper articles, which has limited the language style to written Sinhala (in contrast to spoken Sinhala, which has a large variety), but not constrained to any particular domain. Works by Manamperi et al. (2018) and Dinushika et al. (2019) are some more examples of simple continuous speech recognition systems for Sinhala. In all above works, the methods of data preprocessing only focus on the speech data. Those methods include removing recordings that contain noise in audio and removing or re-transcribing utterances which do not match with their original prompts. According to de Silva (2021), having sufficiently large corpora is a vital necessity for the development of Natural Language Processing (NLP) applications for any language.

[1] https://openslr.org/52

[2] https://kaldi-asr.org/

Various attempts have been made to generate large text corpora which can be used in Sinhala NLP applications including ASR. A common approach taken in those attempts is web crawling on sites such as Sinhala news sites (de Silva, 2015; Jayawickrama et al., 2021). A large text corpus using Facebook posts has been made available by Wijeratne and de Silva (2020). It is admirable that these attempts have resulted in significantly large corpora. Also, every corpus of them is domain independent, since they are generated by web crawling. However, there is no notion of preprocessing in any of those corpora.

3. Motivation

We have identified the potential of development in the research area of Sinhala speech recognition by making available a well-produced, readily-usable corpus. When addressing the 'well-produced' clause there, we focus on four conditions related to the generation of the corpus. The conditions are that the corpus needs to (i) be sufficiently large in size, (ii) be phonetically balanced (particularly if the corpus is small), (iii) have recordings of multiple speakers with a balanced gender distribution, and (iv) have a vocabulary which is domain independent.

Next, we focus on two properties of the generated corpus, with regard to the 'readily-usable' clause mentioned above. The first one is that the corpus needs to be refined. Since an ASR training corpus is a combination of both speech and text data, we are targeting them both. We are especially concerned about the correctness and consistency in terms of spelling and syntax, which we believe is often overlooked when it comes to lexically diverse languages. When investigating the related work, we noticed that such a level of refinement has not been made on any of those corpora, possibly because that requires a great deal of study on the linguistics related to the particular language. The next property of this clause would be, of course, the availability to the public.

The OpenSLR-52 corpus is a good candidate to produce our target deliverable. It satisfies all conditions under the 'well-produced' clause and is publicly available. As one might correctly guess, it lacks the property of being refined. Therefore, we will be putting our effort into refining that corpus efficiently, focusing on specific considerations when using the Sinhala language.

4. Dataset

4.1. Corpus Statistics

OpenSLR-52 is a publicly available, annotated corpus, generated using crowdsourced speech (Kjartansson et al., 2018). It contains separate recordings of 185,293 utterances taken from 478 speakers. The speech data adds up to a total of 224 hours. These recordings are in the *wav* format and the average duration of a recording is about 4 seconds. In addition to the recordings, the corpus provides a *tsv* (Tab-separated values) file which contains the recording ID, the anonymized speaker ID, and the transcription in Sinhala Unicode, for each recording.

4.2. Issues in the Corpus

In this section, we describe the issues we identified with respect to the content of the OpenSLR-52 corpus.

4.2.1. Unavailable Metadata

To conduct experiments in the widely used Kaldi toolkit, the '*spk2gender*' file which maps speakers to their genders is required. Each entry in this file should be in the form of <*speaker ID*> <*gender*>. Unfortunately, the OpenSLR-52 corpus does not contain this information.

4.2.2. Issues Related to Textual Characters

Punctuation Marks Automatic speech recognition systems normally generate the textual format of the spoken utterance without punctuation marks (Fu et al., 2021), and hence there is no need for punctuation marks in corpora used to train speech recognition systems. However, the OpenSLR-52 corpus contains punctuation marks in the transcriptions of the utterances and, further, they are not consistent. Hence, we identified that having these punctuation marks is a challenge when preparing the required files especially when conducting experiments for speech recognition as it introduces additional noise. In Figure 1, we have shown a few examples where punctuation marks are present in the transcriptions followed by an explanation on how they can negatively affect the ASR performance.

```
053392f7c5  බැංකු ඉදිරියේ පෙළපාළි ගියා.
020a9e8c8b  තමන්ගේ අනන්‍යතාවකුත් එක්කම ඉස්සරහට ගියා
```

Figure 1: Inconsistent punctuation marks in transcriptions

When using an ASR toolkit, a list of distinct words is needed to generate a lexicon. If we get the distinct words by splitting the sentences using white spaces and listing each character sequence as a word, in the two examples in Figure 1, we will get 'ගියා.' and 'ගියා' as two distinct words, which is an incorrect behaviour. This introduces additional noise, and the ASR system may predict either one of these two words as the output. Since the outputs are compared to the actual transcriptions when evaluating the accuracy of the ASR system, the above behaviour can introduce false negatives, resulting in a lower accuracy.

In this project, we have used *Subasa*[3] transliteration tool to obtain phoneme sequences for the words in the lexicon. The obtained phoneme sequences for each word in the utterance "අමුම අමු 'තුප්පහියෙක්'." are '*amum@*', '*amu*', '*t^upp@hiyekk*'. Here, the correct phoneme sequence for the word 'තුප්පහියෙක්' should be '*t^upp@hiyek*' (having a single '*k*'), but *Subasa* is producing an incorrect transcription due to the presence of punctuation marks.

English Utterances In the corpus, there were 6,865 English utterances in the Latin script. When compared with the total number of utterances in the corpus, this is a small percentage of utterances. As the use of this corpus is Sinhala speech recognition, ideally it should not contain such utterances. Some examples of such utterances are shown in Figure 2.

```
1c336338c7  winners of miss world
fc0e596656  tea party ideas
```

Figure 2: Transcriptions of English utterances

[3] http://transliteration.sinhala.subasa.lk/

Numeric Characters The transcriptions of the corpus contain numeric characters to represent values such as quantities, years, and dates. An issue we identified in this is that numbers are spoken differently depending on what they represent. Further, in Sinhala, the nouns representing numbers have inflections subject to the case (i.e., nominative, dative etc.).

```
d27179e3bc  මූලික වන කාරණා 4 නම්
640d9670d4  පැය 4 ක් පාගමනින් ගොස්
98ce602b0c  මෙම පිළිම 4 න් නැගෙනහිර දිශාවට මුහුණ ලා ඇති පිළිමය
```

Figure 3: Transcriptions having the character '4'

Consider the transcriptions shown in Figure 3. Even though they contain the same numeric character, the cases of the numbers are different. In the first utterance, the case is definite-nominative, in the second, it is indefinite-nominative, and in the third, it is instrumental. As a result, the three numbers are spoken as /hʌθərə/, /hʌθərʌk/, /hʌθərɛn/. Therefore, by having numeric characters to represent numbers in the transcriptions, the ASR system cannot get an idea of what has actually been spoken, leading to a suboptimal training. This in turn will reduce the performance of the ASR system.

```
38f51660ab  මුල් පියවර 04 අසාර්ථක වූ විට
d27179e3bc  මූලික වන කාරණා 4 නම්
036d076500  මහරජ මෙම කාරණා හතර
```

Figure 4: Transcriptions in which the same number is represented in different ways

Another issue with numbers is that, as shown in Figure 4, all '**04**', '**4**', '**හතර**' are spoken in the same way. If we have the transcriptions in this way, the ASR system may output any of these as the prediction. However, when measuring the performance of the ASR system, since the output is compared with the actual transcription, this might give false negative results thereby leading to reducing the accuracy. Moreover, transliteration tools may not output the correct phoneme sequence for numbers. For example, as shown in Figure 5, *Subasa* simply outputs the numeric characters themselves as the phoneme sequences corresponding to the numbers included in the text. It is completely pointless since they are not able to represent any phonetic information.

```
1948    1948
04  04
4   4
15  15
```

Figure 5: *Subasa* giving the same character sequence as the output for numbers

The above scenarios are sufficient to understand that having numeric characters in the transcriptions is an issue and it will downgrade the performance of the ASR system.

Unnecessarily Applied Non-printable Characters In the Sinhala Unicode implementation, Zero-Width Joiner or ZWJ (U+200D), a non-printable character, is used to represent the modifiers '*rakaranshaya*' and '*yanshaya*', by combining ZWJ with certain character sequences (Punchimudiyanse and Meegama, 2015).

In Figure 6, we have shown those combinations for those modifiers.

```
Rakaranshaya (්‍ර) → ්(U+0DCA) + U+200D + ර(U+0DBB)
Yanshaya (්‍ය) → ්(U+0DCA) + U+200D + ය(U+0DBA)
```

Figure 6: Character combinations containing ZWJ used to denote modifiers

In words where those modifiers are used, applying ZWJ is necessary. However, in the OpenSLR-52 corpus, we found many instances where this character has been applied unnecessarily. Some examples are shown in Figure 7.

```
97bdea9202  අපි වැඩ කරන කාලේ
9e546ef018  කතාවෙන් වගේම ක්‍රියාවෙන්.
```

Figure 7: Examples of transcriptions which include ZWJ unnecessarily

In the above two transcriptions, ZWJ is present in the character sequences which represent the words 'කාලේ' and 'වගේම', even though neither of them contains either of the modifiers mentioned above. Further, we also found some other unnecessarily applied non-printable characters appearing in the text. They are Zero-Width Space (U+200B) and Zero-Width Non-Joiner (U+200C). The issue here is that when any of these non-printable characters are present in a character sequence which they are not supposed to be in, that character sequence is considered as a distinct word in addition to the same word which does not contain non-printable characters. This incorrect behaviour creates inconsistencies, affecting the overall performance of the ASR system.

4.2.3. Issues Related to Linguistics

In Sinhala, spelling is tricky due to two reasons: (i) the Sinhala alphabet contains more than one character which have similar (almost the same) pronunciations, (ii) as mentioned in section 1, Sinhala has a significant lexical diversity, meaning that the spelling of a word is sometimes determined according to its contextual meaning. This can be explained using the following example. There are two characters as 'ල' and 'ළ' which correspond to the lateral consonant of /l/. Ideally, they are supposed to have dental and alveolar articulations respectively, even though practically they are never pronounced differently. However, the use of characters 'ල' and 'ළ' in words 'කල' *(at the time)* and 'කළ' *(done)* differentiates not only the meanings but also the pronunciation of the first character 'ක' as /kʌ/ and /kə/ respectively. When going through the transcriptions of the corpus, we found many similar instances, of course, in addition to obviously misspelled words. Table 1 shows some examples from the corpus where words have contextually incorrect spellings and Table 2 shows words having obviously incorrect spellings. We also focused on Sinhala grammar rules which define spacing between words and prepositions. Just like with spelling, ambiguity also comes around with spacing due to the same reason of having a complex lexical diversity. The same character (or set of characters) can be a valid word or a preposition if used in isolation, as well as a suffix if combined with another word. For instance, the character 'ගේ' appears in the word 'නෑදෑයින්ගේ' *(of relations)* as a suffix whereas it is a separate word in the phrase 'කලින් ගේ නැඟුවා' *(built the house early)*. The effect of this on the ASR

system is indeed subtle because, if we carefully observe, we can see that in both cases, the non-space character before 'ගේ' is 'න්', which means that the acoustic model would be capturing the same phone sequence. Therefore, whether to include a space or not can be only determined according to the context. Table 3 shows some examples from the corpus where spaces are omitted contextually incorrectly, and Table 4 and 5 shows instances having obviously incorrect spacing.

Transcription and the intended meaning in English	Contextually incorrect spelling pattern of the underlined word	Contextually correct spelling pattern
<u>එකේ</u> සුන්දරත්වය සංචාරක සටහනකින් විස්තර කරන්න බැහැ *(The beauty of that cannot be explained using a travel log)*	එකේ /ɛkɛː/ *(of [something declared by a preceding noun])*	ඒකේ /ɛːkɛː/ *(of that)*
ඉතිං <u>දරුව</u> ලොකු වෙද්දිත් මේ තාත්තා කොච්චර හොඳ මනුස්සයෙක් උනත් *(Even when the child grows up, though this father is a good man)*	දරුව /ðʌruvə/ *(child* [vocative]*)*	දරුවා /ðʌruvɑː/ *(child* [nominative]*)*

Table 1: Transcriptions containing words having contextually incorrect spellings

Incorrect spelling	Correct spelling	English translation
පුලුවන්	පුළුවන් /puluvʌn/	*possible*
මිණිමැරුම	මිනීමැරුම /mɪniːmæruməˈ/	*murder*
ව්‍යවස්තාව	ව්‍යවස්ථාව /vyəvʌsθɑːvə/	*constitution*
පන	පණ /pʌnə/	*life*

Table 2: Obviously misspelled words, their correct spellings, and corresponding English translations

Transcription and the intended meaning in English	Contextually incorrect spacing of the underlined token, and the corresponding meaning	Correct spacing according to the context
චිත්‍රපටියක් බලන්නට <u>යාමය</u>. *(Is [the act of] going to watch a movie)*	යාමය *(time)*	යාම ය *(is going* [gerund]*)*
අවස්ථාව <u>උදාවේ</u> *(The opportunity is arising)*	උදාවේ *(of the dawn)*	උදා වේ *(is/are arising)*

Table 3: Transcriptions having contextually incorrect omission of spaces

Incorrect, since a space is omitted	Correction by including a space
ඔබවටා	ඔබ වටා *(around you)*
මොනවගේ	මොන වගේ *(like what)*
මේකරන	මේ කරන *(being done like this)*

Table 4: Obviously incorrect omission of spaces and corresponding corrections

Incorrect, since a space is included	Correction by omitting the space
ලාංකිකයන් ගේ	ලාංකිකයන්ගේ *(of Sri Lankans)*
අපිව ත්	අපිවත් *(also us)* [colloquial]
ඔයා ට	ඔයාට *(to you)*

Table 5: Obviously incorrect inclusion of spaces and corresponding corrections

Since proper use of spelling and spacing determines valid words and their meanings according to the context they appear, we can understand that their effects will get reflected on all three models in the ASR system.

5. Methodology

This section describes the approaches we took to address the issues we identified in section 4.

5.1. Completing Required Metadata

To find out the gender information of each speaker, initially we extracted the unique speaker IDs of all the speakers using the *tsv* file provided with the corpus. There were 478 unique speaker IDs corresponding to distinct speakers. Then, we listened to at least one of their recordings and, depending on the tone of each speaker's voice, manually labelled the speaker as female or male. When doing this, we came across some speakers who were difficult to characterize as female or male based on their vocal tones (e.g., young boys having high-pitched voices). In those instances, we labelled them considering the pitch range of the voice. Speakers who have high-pitched voices were labelled as females while those who have low-pitched voices were labelled as males.

5.2. Treating Character-wise Errors

Punctuation Marks The percentage (%) mark was replaced by inserting the word 'සියට' *(percent)* preceding the numerical word, since it is the way to express percentages in Sinhala. All other punctuation marks (e.g., . , " " ? / : etc.) were replaced by empty strings. For example, 'මෙම වසරේ **15%ක්** හා පසුගිය…' was replaced by 'මෙම වසරේ **සියට පහළවක්** හා පසුගිය'.

English Utterances The English utterances in the Latin script were removed from the OpenSLR-52 corpus by filtering them out. This was done by running a Python script on the *tsv* file.

Numeric Characters First, the utterances containing numeric characters were filtered out from the total set of utterances. Then, by listening to each of their corresponding recordings, the numbers were manually replaced by their textual formats according to the context. Some examples are shown in Table 6.

Original	Replaced by text
මූලික වන කාරණා **4** නම්	මූලික වන කාරණා **හතර** නම් *(The main four factors are)*
1818 දී උඩරට කැරැල්ල හට ගැනීමයි	**එක්දාස් අටසිය දහ අටේ** දී උඩරට කැරැල්ල හට ගැනීමයි *(Is the formation of the 'upcountry' rebellion in [year] eighteen-eighteen)*

Table 6: Utterances containing numeric characters, and their textual replacements

Unnecessary Non-printable Characters Our approach of treating these characters was as follows. First, the ZWJ characters (U+200D) in all the transcriptions (regardless of whether it is necessary or not) were removed by replacing them with empty strings. Then, as mentioned earlier, since ZWJ is needed to represent *rakaranshaya*, a ZWJ character was added in between each ් (U+0DCA) + ර (U+0DBB) combination using a Python script. However, after that, some words became incorrect due to the following reason. In Sinhala, there are words which have ් (U+0DCA) + ර (U+0DBB) combination where the *rakaranshaya* is not used (e.g., දුම්රිය *(train)* → ද + ු + ම + ් (U+0DCA) + ර (U+0DBB) + ි + ය). Therefore, we filtered out such words and corrected them manually. Using a similar fashion, a ZWJ character was inserted in between each ් (U+0DCA) + ය (U+0DBA) combination as it was needed to represent *yanshaya*. Here too, similar to the peculiarity with *rakaranshaya*, Sinhala language contains words which have the ් (U+0DCA) + ය (U+0DBA) combination where the *yanshaya* is not used (e.g., කාර්යය *(work)* → ක + ා + ර +් (U+0DCA) + ය (U+0DBA) + ය). These were corrected following a similar approach as we did for *rakaranshaya*.

According to Punchimudiyanse and Meegama (2015), the ZWJ is also used to represent the modifier '*repaya*' (e.g., කර්‍ය, වර්‍ණය) and conjunct character pairs (e.g., ක්‍ෂ, ක්‍ව). By removing the ZWJ, the character 'ර්' was retained in places having *repaya* (e.g., ධර්‍මයට → ධර්මයට). The conjunct character pairs got separated, leaving a '්' character added to the first character in each such pair (e.g., භික්‍ෂූන් → භික්ෂූන්). In both cases, we did not attempt to replace them with their original characters because the resulting formats are perfectly valid and accepted in Sinhala.

Other than the ZWJ, we also replaced the other non-printable characters Zero-Width Space (U+200B) and Zero-Width Non-Joiner (U+200C) with empty strings. However, corrections were not required since those characters were not necessary in any case.

5.3. Applying Linguistic Corrections

We followed a find-and-replace approach to address the issues explained in 4.2.3. That is, we first filtered occurrences where erroneous text can exist, and extracted the distinct words/word-pairs in those occurrences. Then we prepared dictionaries (i.e., key-value structures) having the incorrect text as keys and manually added the correct text as the respective values. Finally, using those dictionaries, we replaced the incorrect texts with the corresponding correct texts. The filtration and replacement steps were automated using Python scripts. The advantage of following the above approach is that the artifacts (i.e., scripts and dictionaries) can be reused to apply corrections on any new data.

We created a non-exhaustive list of obviously misspelled words by going through the transcriptions. For words that are spelled contextually incorrectly, we added a field in each respective dictionary to denote the specific occurrences to be corrected. In that way, we were able to ensure that the same words which were originally spelled contextually correctly are left unchanged. Using this approach, we were able to correct most of the commonly misspelled words.

To address the spacing issues, we listed 61 Sinhala grammar-based rules for proper spacing, referring to textbooks (National Institute of Education, 2001). Those rules were listed in a way that (i) they ensure consistency throughout the whole set of transcriptions and (ii) they are independent of each other so that the order of applying corrections for each rule does not affect the final outcome. Many of the rules required human decision as they rely on the meaning of words.

When applying corrections on both spelling and spacing errors, we listened to the original recordings of those utterances which we suspected were misaligned with their transcriptions. In cases where the speaker has spoken valid words, we modified the mismatching words in the transcriptions by replacing them with what the speaker has uttered. We removed the utterances where the speaker has spoken invalid words (i.e., gibberish) because the prompt contained misspelled words.

The exercise of applying refinements described in this section will result in properly distinguishing words and prepositions, along with their proper associations with suffixes. This can reduce the variance (i.e., scatter) in the vocabulary and allow the models to fit properly. On the other hand, since the models get trained by the lexically and grammatically correct usage of the language, the ASR system, when applied in practice, will be able to predict its outcome in correct Sinhala. That would be a meaningful contribution we can make to the digital transformation of use of the Sinhala language.

Version of the corpus	Original	Refinements in 5.2 applied	Refinements in 5.3 applied
Total utterances	185,293	178,409	178,096
Unique utterances	102,576	98,435	98,127
Unique words	69,581	63,376	57,029

Table 7: Statistics of the corpus before and after application of refinements

Statistics of the corpus before and after application of refinements[4] described in 5.2 and 5.3 are shown in Table 7. From that, we can see the reduction of utterances and words in the corpus after sequentially applying the refinements.

6. Evaluation

We designed two experiments to analyze the effectiveness of the linguistic corrections applied.

[4] https://github.com/SinSpeech-Development/Refined-OpenSLR-52-Corpus

- Experiment 1: Using the corpus with corrections mentioned in section 5.2 applied for training
- Experiment 2: Using the corpus with corrections mentioned in both sections 5.2 and 5.3 for training

The experiments were conducted on a GMM-HMM based architecture using a recipe available in the Kaldi toolkit.
To ensure no bias in experiments, we designed and implemented a splitting algorithm such that there were no overlapping utterances between train and test sets. However, for fair comparison, the same test set having all the refinements (explained in section 5.2 and 5.3) must be used in both experiments 1 and 2. For that, we first applied the splitting algorithm on the completely refined corpus and obtained the train and test sets for Experiment 2. Then we created the train set for Experiment 1 using the same utterances that correspond to those in the train set of Experiment 2, but without linguistic corrections applied. The test set was the same in both experiments. The proportions of the train and test sets were 80% and 20% respectively. Also, to ensure that each set contains a balanced set of words, we shuffled the dataset before splitting. The two sets had a balanced gender distribution. Further, utterances in the test set were not used when preparing the language model and the lexicon as it may bias the results.

6.1. Experimental Results

Training Pass	Experiment 1	Experiment 2
Monophone	64.17	**59.25**
Triphone pass 1	49.20	**42.72**
Triphone pass 2	47.24	**40.67**
Triphone pass 3	43.21	**36.34**

Table 8: Word-Error-Rates (WERs) of each training pass in the two experiments

From the results shown in Table 8, it is clear that in all passes of training, Experiment 2 has lower WERs, proving that our refinements have increased the performance. Considering the final training pass (Triphone pass 3), we can observe a relative reduction of 15.9% in the WER.

6.2. Comparison of Decoded Texts

In this subsection, we provide a comparison of the texts decoded by the models trained in the two experiments. While Table 9 and Table 10 shows the effect of applying grammatically correct and consistent spacing, Table 11 highlights the effect of introducing the spelling corrections. The set of examples shown in the tables are utterances taken from the test set. We have underlined the occurrences which contribute to the changes in WER.

Original transcription	Decoded text	
	Experiment 1	Experiment 2
ඒ ඒ කාලවලට ඒ ඒ විදිහට අනුව *(According to method for each period)*	ඒ ඒ කාල වලට ඒ ඒ විදිහට අනුව	ඒ ඒ කාලවලට ඒ ඒ විදිහට අනුව
මේ තොරතුරුවලට *(For these details)*	මේ තොරතුරු වලට	මේ තොරතුරුවලට

Table 9: Effect of proper removal of spaces

Original transcription	Decoded text	
	Experiment 1	Experiment 2
පෙර බුදුවරුන්ගේ ශාසනවල දී ද *(In the Sāsana [regimes] of previous Buddhas too)*	පෙර බුදුවරුන්ගේ ශාසනවලදී ද	පෙර බුදුවරුන්ගේ ශාසනවල දී ද
නිදසුන් වශයෙන් දැක්විය හැකි ය *(Can give as examples)*	නිදසුන් වශයෙන් දැක්විය හැකිය	නිදසුන් වශයෙන් දැක්විය හැකි ය

Table 10: Effect of proper inclusion of spaces

Original transcription	Decoded text	
	Experiment 1	Experiment 2
එක්දාස් නමසිය හැට හතරේ දී ගැසට් කර ඇති මේ පූජනීය ස්ථානය කෙරෙහි *(Regarding this sacred place which has been gazetted in 1964)*	එක්දාස් නමසිය හැට හතරේ දී ගැසට් කර ඇති මේ පූජණීය ස්ථානය කෙරෙහි	එක්දාස් නමසිය හැට හතරේ දී ගැසට් කර ඇති මේ පූජනීය ස්ථානය කෙරෙහි
එකක් තමයි යම් දියුණුවක් ලබමින් තිබුණ අපේ පොඩි පොඩි කර්මාන්තවලට *(One thing is for our small-scale industries which were growing)*	එකක් තමයි යම් දියුණුවක් ලබමින් තිබුන අපේ පොඩි පොඩි කර්මාන්තවලට	එකක් තමයි යම් දියුණුවක් ලබමින් තිබුණ අපේ පොඩි පොඩි කර්මාන්තවලට

Table 11: Effect of spelling corrections

7. Conclusion

In this paper, we have discussed a systematic approach to derive a refined corpus for Sinhala speech recognition. In our approach, we have not only removed misaligning audio files, but more importantly, we have cleaned the text corpus by removing non-Sinhala characters and punctuation marks, and replacing digits with their textual format. Further, by studying the linguistic characteristics of Sinhala, we have corrected the words which are misspelled, and have enforced a consistent and grammatically-correct way of spacing between tokens. Our experimental results show a promising effect of those refinements.
Our systematic approach of refining is not only able to elegantly cover a significant number of errors, but also scalable to extend the text corpus with new data. The scalability is achieved by retaining the correction definitions applied to the existing set of data, which allows us to reuse those when adding new data. It will substantially reduce the amount of corrections for a new set of data, as we are only required to define corrections for errors which uniquely exist in the new set of data. This approach substantially reduces the effort required to preprocess a corpus by avoiding the need to manually go through the complete set of text data.
We believe that we have not only contributed to the development of research on Sinhala ASR by deriving a readily-usable corpus, but also proposed guidelines to improve the quality of any corpus so that ASR applications can perform well in the language they are used upon.

8. Bibliographical References

Amarasingha, W., and Gamini, D. (2012). Speaker Independent Sinhala Speech Recognition for Voice Dialling. In *Proceedings of the 13th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 3-6, Colombo, Sri Lanka.

de Silva, N. (2015). Sinhala Text Classification: Observations from the Perspective of a Resource Poor Language.

de Silva, N. (2021). Survey on Publicly Available Sinhala Natural Language Processing Tools and Research.

Dilshani, W. S. N., Yashothara, S., Uthayasanker, R. T., and Jayasena, S. (2018). Linguistic Divergence of Sinhala and Tamil Languages in Machine Translation. In *Proceedings of the International Conference on Asian Language Processing (IALP)*, *pages* 13-18, Bandung, Indonesia.

Dinushika, T., Kavmini, L., Abeyawardhana, P., Thayasivam, U., and Jayasena, S. (2019). Speech Command Classification System for Sinhala Language based on Automatic Speech Recognition. In *Proceedings of the International Conference on Asian Language Processing (IALP)*, pages 205-210, Shanghai, China.

Fu, X. Y., Chen, C., Laskar, M. T. R., Bhushan, S., and Oliver, S. C. (2021). Improving Punctuation Restoration for Speech Transcripts via External Data.

Gunasekara, M. K. H., and Meegama, R. G. N. (2015). Real-time translation of discrete Sinhala speech to Unicode text. In *Proceedings of the 15th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 140-145, Colombo, Sri Lanka.

Jayawickrama, V., Ranasinghe, A., Attanayake, D. C., and Wijeratne, Y. (2021). A Corpus and Machine Learning Models for Fake News Classification in Sinhala.

Karunathilaka, H., Welgama, V., Nadungodage, T., and Weerasinghe, R. (2020). Low-resource Sinhala Speech Recognition using Deep Learning. In *Proceedings of the 20th International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 196-201, Colombo, Sri Lanka.

Malik, M., Malik, M. K., Mehmood, K., and Makhdoom, I. (2021). Automatic speech recognition: a survey. *Multimedia Tools and Applications*, 80(3):1-47.

Manamperi, W., Karunathilake, D., Madhushani, T., Galagedara, D., and Dias, D. (2018). Sinhala Speech Recognition for Interactive Voice Response Systems Accessed Through Mobile Phones. In *Proceedings of the Moratuwa Engineering Research Conference (MERCon)*, pages 241-246, Moratuwa, Sri Lanka.

Nadungodage, T., and Weerasinghe, R. (2011). Continuous Sinhala Speech Recognizer. In *Proceedings of the Conference on Human Language Technology for Development*, pages 141-147, Alexandria, Egypt.

National Institute of Education. (2001). Sinhala Lekana Reethiya. Maharagama, Sri Lanka.

Punchimudiyanse, M. and Meegama, R. G. N. (2015). Unicode Sinhala and Phonetic English Bi-directional Conversion for Sinhala Speech Recognizer. In *Proceedings of the IEEE 10th International Conference on Industrial and Information Systems*, pages 296-301. Sri Lanka

Wijeratne, Y. and de Silva, N. (2020). Sinhala Language Corpora and Stopwords from a Decade of Sri Lankan Facebook.

9. Language Resource References

Kjartansson, O., Sarin, S., Pipatsrisawat, K., Jansche, M. and Ha, L. (2018). Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali. In *Proceedings of the 6th International Workshop on Spoken Language Technologies for Under-Resourced Languages*, pages 52-55, Gurugram, India.