Abstract—An automated speech recognizer (ASR) having a large vocabulary is yet to be developed for the Sinhala language because of the time consuming nature of gathering the training data to build a language model. The dictionary and building the language model require non-English text, in our case, Sinhala Unicode, to be transcribed in phonetic English text. Unlike text to speech conversions which only require transcribing the non-English text to phonetic English text, an ASR needs correct reproduction of the original language text when the phonetic English text is produced as the output of the speech recognizer. In the present research, newspaper articles are used to gather a large set of sentences to build a language model having thousands of words for the Sphinx ASR. We present a decoder algorithm that produces phonetic English text from Sinhala Unicode text and an encoder algorithm that produces the correct reproduction of Unicode Sinhala text from phonetic English. For a near phonetic tag set for Sinhala alphabet, results indicate 100% accuracy for the decoder algorithm while for numberless text, accuracy of the encoder algorithm stands at 98.61% for distinct phonetic English words.

Keywords—Sinhala to Phonetic English, Phonetic English to Sinhala, Sinhala ASR, Sinhala Phonetic Tag set, Sphinx

I. INTRODUCTION

Commercial and open source products to automatically recognize spoken English is available in the market today. In terms of performance and the accuracy, ASR systems that are based on acoustic and language models utilize hidden markov model (HMM) or Gaussian mixture models (GMM) for recognition. Most popular such open source ASR systems include Sphinx, HTK and Julius [1-3]. HMM based ASR systems typically have two phases, namely, the training phase and the recognition phase [16]. Onetime training phase is used in which a single or multiple user's voice is utilized to train a proper speech model called an acoustic model. In the recognition phase, the speech model built in the previous phase is recurrently used to recognize spoken words of the speaker.

The process of training a speech model requires a collection of acoustic data, a recorded voice sample of a person, and the correct phonetic text transcription of that voice sample. Adopting this technique for systems built for non-English languages require a production of non-English words in phonetic English text for the speech trainer. In order to build the phonetic text transcription from the original Unicode text, a proper phonetic tag per character is necessary.

Sinhala Unicode text to phonetic English tagging (syllabification) is not a straightforward task because Sinhala characters have several modifiers that are typed after the actual base character but appears before, after, and the bottom of the actual base character. In addition, a non printable zero-width joiner (Unicode U+200D) is inserted between the characters to convert certain character combinations to one of the modifiers named yansaya, rakaranshaya or repaya which are not present in Sinhala Unicode.

For an ASR application to work, both Sinhala Unicode to phonetic text conversion and proper reproduction of Sinhala Unicode from phonetic English output of the speech recognizer is necessary. Phonetic tags for Sinhala alphabet has to be searched and replaced in a specific order to prevent invalid construction of words from phonetic English. For example, if phonetic tag for the characters $ is i and $ is ii with search order i, ii is used to encode a phonetic English word giithaya (meaning song), the word is constructed as (of + ɣ + i + ɣ + $ + a) @ giithaya which is misspelled. Correct spelling of the word giithaya will be constructed if the search order ii,i is followed.

The ASR applications use N-gram based techniques to construct word combinations to build a language model in phonetic English. Therefore, it is necessary for the researchers to identify suitable phonetic tag set to develop a Sinhala to phonetic text decoder algorithm and phonetic English to Sinhala Unicode encoding algorithm.

II. BACKGROUND AND RELATED WORK

Representation of Sinhala characters in digital form commenced in early 1980s to display Sinhala subtitles in television. The Wijesekara keyboard used in the type writers are extended to Sinhala fonts and keyboard drivers used in the computers. The Sinhala alphabet is first standardized by the Sri Lanka standards (SLS) institute as Sinhala Character Code for Information Interchange (SLS 1134:1996) and subsequently revised in 2004 and 2011 (SLS 1134:2004, SLS 1134:2011) [5]. The Sinhala alphabet representation in Unicode is first added to the Unicode standard version 3 based on the ISO/IEC 10646 [4].
The Sinhala language has 18 vowels, 2 half vowels and 40 consonants in modern Sinhala alphabet of 60 characters [6]. The consonant letter ɐ is present to make it 61 characters in the Unicode standard. In addition, there are several modifiers placed after a consonant which corresponds to vowel sounds [7]. The first well documented syllabification effort claims an accuracy of 99.95% and employs a rule based algorithm to produce the output in English characters using the International Phonetic Alphabet (IPA) format [9]. The IPA format for the Sinhala [12] consists of characters that are not part of the 26 letter English alphabet. A different tag set is proposed to represent phonetic sounds for the Sinhala alphabet with an accuracy of 98.21% for the purpose of text to speech (TTS) applications utilizing the festival framework [13].

A syllabification tool published by the University of Colombo School of Computing (UCSC) also employs a slightly modified tag set to the one that is proposed in [13] to convert Sinhala Unicode text to phonetic English text where the output of each character is separated by a space [14].

Although the tag set in [13] work well with TTS, it presents several problems for ASR applications as it does not suggest different tags for characters with similar sounds but different characters. Therefore, phonetic English text generated by those tag sets make wrong classifications of acoustic sounds in the training phase of the ASR and make erroneous output when building a language model as well. The reason behind this scenario is that the words spelled correctly are replaced by misspelled words during the conversion from Sinhala Unicode to phonetic English producing errors in the ASR output.

This research paper presents an improved tag set to decode Sinhala Unicode text gathered from online Sinhala news paper articles to phonetic English text with a decoder algorithm and an encoder algorithm to convert phonetic English to Sinhala Unicode text in an ASR application.

III. METHODOLOGY

The Sphinx ASR speech trainer tool (training phase) requires Sinhala Unicode sentences to be represented in phonetic English format. In addition, it requires a list of distinct words with its phonetic pronunciations separated by a space as the pronunciation dictionary.

A. Special Features Associated with Phonology of Sinhala

Technical explanation of Sinhala Unicode to phonetic English decoder algorithm is not fruitful without giving specifics of how Sinhala language words are constructed from alphabetical characters. In addition to the occurrences of vowels and base consonants, one or more modifier is typed after a consonant but the appearance of the modifier could be before, after, top and bottom of a particular consonant.

The pronunciation of the derived consonant varies with the associated modifiers. Most of the modifiers are inheriting the sounds of a vowel. The al modifier (ə) does not have a vowel associated to its sound. A word in Sinhala language is constructed from a set of phonetic sounds related to the alphabet. Phonologically a base consonant (termed as pure consonant) is denoted as a consonant with a əl (ə) modifier.

For example the consonant ɐ (ka) consist of ə + q (k + a) sounds where ək (k) is consonant with əl (ə) modifier and q (a) is a hidden vowel according to phonetic pronunciation. Therefore, a consonant written without a modifier is having the hidden vowel sound q (ə) associated with it. The set of derived consonants attaching one or more modifiers after the consonant ɐ (ka) is depicted in figure 1 [9, 10].

Three special modifiers, namely rakaranshaya, repaya, yansaya are used with consonants but do not have a corresponding character code in Unicode standard. When a specific character and modifier combination is encountered after a consonant, such sequences are immediately replaced with a respective special character or modifier by using either a font driver or a keyboard driver of the Sinhala language [17].

There are conjuncts (two or more letters combined to form a single letter) present in some of the words which are not compulsory to write in that manner. If a conjunct is detected in a word, it has to be processed by the decoder to produce correct phonetic English and reproduced in the same manner by the encoder that produces Unicode text.

Those three special modifiers, as well as the conjuncts, require a special non-printing character called a zero-width joiner (ZWJ) inserted appropriately to build a proper character sequence in Sinhala Unicode text [8]. The character combinations used in special modifiers and two common conjuncts are given in the Table I.

When designing an algorithm to decode Sinhala Unicode to phonetic English, the features present in a standard text of Sinhala language need preservation across the decoding process. A sentence is read by the decoder algorithm and Sinhala characters and modifiers are replaced with the relevant phonetic tag. The reconstruction process of the recognized phonetic English text output of ASR to Sinhala Unicode text will fail if those features are overlooked or lost at the decoding process.

### Table I. Special Modifiers and Conjunctions

<table>
<thead>
<tr>
<th>Derived consonants</th>
<th>Type</th>
<th>Required Character Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>ɐ</td>
<td>Modifier = rakaranshaya</td>
<td>əf + ZWJ + 5 + q</td>
</tr>
<tr>
<td>ɐ</td>
<td>Modifier = rakaranshaya, ə</td>
<td>əf + ZWJ + 5 + ə</td>
</tr>
<tr>
<td>ɐ</td>
<td>Modifier = yansaya</td>
<td>əf + ZWJ + u</td>
</tr>
<tr>
<td>ɐ</td>
<td>Modifier = repaya</td>
<td>5 + ZWJ + ə</td>
</tr>
<tr>
<td>Conjunct</td>
<td></td>
<td>Z + ZWJ + ə</td>
</tr>
<tr>
<td>Conjunct</td>
<td></td>
<td>Z + ZWJ + ə</td>
</tr>
</tbody>
</table>

Fig. 1. Derived consonants from the consonant ɐ.
B. Improved Phonetic Tag Set

Based on the initial tag set present in [13, 14], several tags are changed to eliminate ambiguity in characters with similar phonetic sounds but having different Unicode character codes. The near phonetic tag set used in this research is given in Table II.

Several new tags are introduced to prevent actual vowels being misinterpreted as modifiers. The phonetic tag used for a particular modifier for a corresponding vowel is given in Table III. The character form derived by adding al modifier ( aç ) to a consonant is considered as the base form (pure consonant) for breaking down a Sinhala word to its phonetic pronunciation.

The Sinhala to phonetic English decoder algorithm is planned to generate the phonetic form automatically from an input character sequence so that it will fulfill the requirement of the Sphinx ASR trainer tools. The encoder algorithm used to reconstruct Sinhala Unicode word from phonetic English output of the speech recognizer also utilizes the same phonetic pronunciation format.

<table>
<thead>
<tr>
<th>Table II. Sinhala Alphabet with Identified Phonetic Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Vowels</td>
</tr>
<tr>
<td>q</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>zri</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>o</td>
</tr>
<tr>
<td>ZWJ</td>
</tr>
<tr>
<td>Consonants and Zero Width Joiner</td>
</tr>
<tr>
<td>z</td>
</tr>
<tr>
<td>g</td>
</tr>
<tr>
<td>n</td>
</tr>
<tr>
<td>jh</td>
</tr>
<tr>
<td>qndh</td>
</tr>
<tr>
<td>zon</td>
</tr>
<tr>
<td>zh</td>
</tr>
<tr>
<td>zsh</td>
</tr>
<tr>
<td>z</td>
</tr>
<tr>
<td>jh</td>
</tr>
<tr>
<td>zg</td>
</tr>
<tr>
<td>m</td>
</tr>
<tr>
<td>l</td>
</tr>
<tr>
<td>f</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table III. Phonetic Tags for Modifiers and Matching Vowels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vowel</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q1</td>
</tr>
<tr>
<td>q</td>
</tr>
<tr>
<td>c</td>
</tr>
<tr>
<td>e</td>
</tr>
<tr>
<td>o</td>
</tr>
<tr>
<td>aix</td>
</tr>
</tbody>
</table>

C. Sinhala Unicode to Phonetic English Decoder Algorithm

The numbers in digit form (0-9) and HTML tags present in the Sinhala Unicode sentences that are copied from the online newspaper articles are removed first. The cleaned up text is processed using the following decoder algorithm. Numbers written in textual form such as "01" (one) in text are allowed.

DECODER ALGORITHM

//This algorithm converts Sinhala Unicode text to phonetic
//English text in order to use with sphinx based Sinhala ASR
//abbreviations: zero width joiner - ZWJ, carriage return - CR
tr = input_string
repeat until end of vowel list
tr = tr.replace(vowel,"@ + phoneticvowel + ")
repeat until end of consonent list
tr = tr.replace(consonent,"$ & phoneticconsonent & ")
repeat until end of modifier list
tr = tr.replace(modifier,"~ & phoneticmodifier & ")
//zero width joiner replacement
tr = tr.replace(ZWJ,"$qx")
//introduce "a" sound for non al consonents
tr = tr.replace("i", "-/a-")
tr = tr.replace("!@", "!-a-@")
tr = tr.replace("i", "!-a-")
tr = tr.replace("!" & CR, "!-a-" & CR)
tr = tr.replace("!@", "!-a-@")
//remove all special characters except /, <, >
add < to the start of tr
remove last < from the tr
return tr
Decoder algorithm is capable of handling multiple sentences separated by linefeed. The tags \(<s>\) and \(<</s>\) is a requirement to build sentence transcription of sphinx language model building tool.

**D. Phonetic English to Sinhala Unicode Encoder Algorithm**

A longer algorithm required to encode phonetic English to Sinhala Unicode because the features mentioned in 3A has to be considered.

//This algorithm converts phonetic English to Sinhala Unicode
//Phonetic English text is without numbers, but number written in text is possible. eg. 1 - is not allowed, one - is allowed
ra = input_phonetic_english_text

//data preparation section
vowelO = vowel list in sinhala unicode
modifierO = modifier list sinhala unicode
consonentO = consonant list sinhala unicode
csingle = Unicode character list with single letter phonetic tag
cdual = Unicode character list with two letter phonetic tag
ctri = Unicode character list with three letter phonetic tag
clarge = Unicode character list with large phonetic tag
singlephone = single letter phonetic tag list
dualphone = two letter phonetic tag list
triphone = three letter phonetic tag list
largephone = phonetic tag list with more than 3 letters for tag

//replace tag with Unicode character in the order of tag size
repeat until eof largephonelist
ra = ra.replace(largephone, clarge)
repeat until eof triphonelist
ra = ra.replace(triphone, ctriple)
repeat until eof dualphonelist
ra = ra.replace(dualphone, cdual)
repeat until eof singlephonelist
ra = ra.replace(singlephone, cingle)

//make a character array to do linear search to convert vowels that immediately follow a consonant to modifiers
tca() = ra.ToCharArray()

//ch - current char, cc - counter, kk, kkr - tempcharacter
while not end of input_phonetic_english_text
if current char = space then
    sout += space
end if
if ch is a consonant then
    check next character tca(cc+1) for in vowel()
if vowel found
    next character = modifier
    kkr = modifier
else
    check next character for modifier without vowel
if found then
    kk = current char & next char
    Increment character count by 1
else if current character is not in ("Oo", "o%") then
    if next 3 characters are "wqx" Then
        GoTo rakaransha_section
    end if
    kk = current char & "d"
else
    kk = current char
end if
rakaransha_section:
//rakaranshaya handling section
If next 4 characters are "wqos" Then
    kk = current char + "dd"
    If fifth character from current char = "q" Then
        Increment character count by 5
    end if
else // otherwise check the additional modi
    check 5th char from current position for vowel
if found fifth character from current = modifier
    kk += modifier
    Increment character count by 3
end if
    Increment character count by 1
end if
//end of rakaransaya handling section
repeat rakaransha section changing δ to ω, υ for yansaya and conjuncts of υ, ω respectively

//repaya handling
if next 4 characters are "wqdx" Then
    kk = "5" + cta(cc+4)
    If fifth character from current char = "q" Then
        Increment character count by 5
    end if
else // otherwise check the additional modi
    check 5th char from current position for vowel
if vowel found
    Fifth character from current char = modifier
    kk += modifier
    Increment character count by 4
end if
    Increment character count by 1
end If
    sout += kk
else
    sout += current char + kkr
end if

IV. RESULTS AND DISCUSSION

Sinhala to phonetic English algorithm (decoder) and phonetic English to Sinhala Unicode (encoder) algorithm given in sections 3C and 3D are implemented and tested with different test cases.
A. Sinhala Unicode to Phonetic English Decoder Algorithm

The functionality of the Sinhala Unicode to phonetic English decoder algorithm is tested with several phases. In the first phase, it is checked for accurate generation of phonetic English text from a given sentence. Test Results are described without <s> and </s> tags which is a format requirement of the Sphinx trainer.

A sentence \( \text{sixonhalenw katalahaxdka hagqendhunaxa gxcenaximeze mizrundhukaxaxongayya saqnddahaxa mema xcaelwgorixhama dheka upakaxarixi veze} \) (meaning: these two algorithms helpful in Sinhala ASR software) is encoded using phonetic English to Sinhala Unicode algorithm. It produces the accurate output \( "\text{it is risky to walk in the rail track}\) is processed using the decoder algorithm to produce the output as rezelw (meaning: It is risky to walk in the rail track) is processed using the decoder algorithm to produce the phonetic English text.

The functionality of the Sinhala Unicode to phonetic English decoder algorithm is tested with several phases. In the first phase, it is checked for accurate generation of phonetic English text from a given sentence. Test Results are described without <s> and </s> tags which is a format requirement of the Sphinx trainer.

A sentence \( \text{paxareze payinw gamanw kirixima anatxhurudhaxayakaya} \) (meaning: It becomes cf + w in the word ©&cf. The tag w implies that the consonant does not have a hidden vowel a present. Therefore, the word ©®af becomes ©®af and subsequently, rezew and the rest of the words are processed with the same notation.

The second phase of testing the decoder algorithm involves generating phonetic text in the presence of special modifiers (repaya, rakaranshaya and yansaya) and conjuncts (combined letters). Table IV depicts the decoding process of the words ©®af (curve), ©®af (truthful), ©®af (colors) and ©®af (letters) tested with online Sinhala typing application "real time Unicode converter" by UCSC [15]. This algorithm converts the words to proper phonetic English format.

The initial results indicate that the decoder algorithm generates phonetic English from Sinhala Unicode in the expected format of the Sphinx ASR trainer. A test set of 500 sentences is gathered from the Divaina, Silumina and Lankadeepa online editions of Sinhala newspapers which are published in the Sinhala Unicode font to test bidirectional conversion between phonetic English and Sinhala Unicode text. The numbers present in the text in digits (0-9) are manually replaced by the word form of the numbers.

A test set of 500 Sinhala Unicode sentences is processed using the decoder algorithm to produce the phonetic English transcription within 3 seconds in a personal computer having a processing speed of 2.7 GHz.

### Table IV

<table>
<thead>
<tr>
<th>Derived consonants</th>
<th>Modifier</th>
<th>Phonetic Form</th>
<th>Phonetic tag separated</th>
</tr>
</thead>
<tbody>
<tr>
<td>©®af</td>
<td>rakaranshaya</td>
<td>vakwqaxraya</td>
<td>v a k w q x r a y a</td>
</tr>
<tr>
<td>©®af</td>
<td>yansaya</td>
<td>a w v w q x y a s a jh a</td>
<td></td>
</tr>
<tr>
<td>©®af</td>
<td>repaya</td>
<td>varwqaxna</td>
<td>v a r w q x z n a</td>
</tr>
<tr>
<td>©®af</td>
<td>conjunct</td>
<td>akwqaxshara</td>
<td>a k w q x z h a r a</td>
</tr>
</tbody>
</table>

B. Phonetic English to Sinhala Unicode Encoder Algorithm

The phonetic English to Sinhala encoder algorithm is tested in several ways. The first test involves carrying out proper reproduction of 61 alphabetical letters from their respective phonetic tags each separated by a space character. The results indicate an accuracy of 100% in this regard.

The next test is carried out for a sentence containing words without special modifiers. The sentence "sixonhalenw katalahaxdka haqendhunaxa gxcenaximeze mizrundhukaxaxongayya saqnddahaxa mema xcaelwgorixhama dheka upakaxarixi veze" (meaning: these two algorithms helpful in Sinhala ASR software) is encoded using phonetic English to Sinhala Unicode algorithm. It produces the accurate output "it is risky to walk in the rail track" (meaning: It is risky to walk in the rail track). A sentence with special modifiers is tested using the same technique. The sentence "paxareze payinw gamanw kirixima anatxhurudhaxayakaya. It is processed using the decoder algorithm to produce the output as rezelw (meaning: It is risky to walk in the rail track) is processed using the decoder algorithm to produce the phonetic English text.

A sentence with special modifiers is tested using the same technique. The sentence "paxareze payinw gamanw kirixima anatxhurudhaxayakaya. It is processed using the decoder algorithm to produce the output as rezelw (meaning: It is risky to walk in the rail track) is processed using the decoder algorithm to produce the phonetic English text.

Researchers have observed that the phonetic English text generated by decoder algorithm using the Sinhala text typed from real time Unicode converter application [15] and the Google input tools provided a 100% accurate output in the encoder algorithm for the three special modifiers yansaya, rakaranshaya and repaya. The encoding algorithm accurately supports one conjunct that is the combined letter ©® which is ©® and ©® added together. Testing with the text converted from other online sources produced errors which will be discussed in the next section.

The output of Sinhala Unicode text from phonetic English source sentence is a continuous effort of experimental comparisons in identifying a proper near phonetic tag set for the Sinhala alphabet. It is also observed that the search order of the phonetic tags has to be from the largest to the smallest.

C. Testing the Bi-directional Conversion

At the final stage, the bidirectional capability of the two algorithms is tested using the following three steps.

**Step 1:** A test set of Sinhala Unicode text is processed by the decoder algorithm to obtain phonetic English text. **Step 2:** The phonetic text is fed as the input to the encoder algorithm to obtain Sinhala Unicode text. **Step 3:** The sentences in the original test set are compared with the regenerated Sinhala Unicode text.

Online editions of the Silumina, Divaina and Lankadeepa Sinhala newspapers possess the least number of typographical errors. As such, Sinhala Unicode text of the articles published in such newspapers are selected for building the test set containing 500 sentences to be used in the bidirectional conversion. Note that the test set of 500 sentences are stripped of all the HTML tags.
The regenerated Sinhala Unicode output of the encoder had 46 errors from a total of 7324 words present in the original test set and the accuracy stands at 99.37%. On the other hand, if the accuracy is taken as the percentage of errors from the number of distinct words, 38 errors out of 2743 distinct words measure up to the encoding accuracy of 98.61%.

It is observed that the encoding errors occur due to the placement of zero-width joiners in the wrong sequence or placed twice in the places with special modifiers rakaransha, yansa and repaya in the original Sinhala text, which is used to generate phonetic English text. This is due to excessive use of zero-width joiners when typing the Sinhala text by the user or the way different keyboard/font driver interprets rakaransha, yansa and repaya by the Sinhala typing application.

Having identified the cause of errors, an accuracy of 100% can be reached after running the set of raw sentence through the decoder and the encoder and replacing the words containing invalid zero-width joiners with correctly typed Sinhala words.

The Sphinx ASR trainer is used to build the Sinhala language model using the phonetic English transcription generated by decoder algorithm taking Sinhala sentences as an Input. Then, the language model, along with the acoustic model, is used with the Sphinx ASR to produce continuous Sinhala voice recognition output in phonetic English. Subsequently, the phonetic English output is fed into the encoder algorithm to generate Sinhala Unicode text via a wrapper application.

V. CONCLUSIONS

Building a language model in phonetic English from Sinhala Unicode text is a tedious task in Sphinx based Sinhala ASR. In this regard, online newspapers are a valuable source of obtaining a large set of Sinhala sentences that can be used to generate a language model.

The main contributions of the proposed technique are the decoder algorithm having an accuracy of 100% for numberless text and the encoder algorithm with an accuracy of 98.61% for distinct words in making successful bidirectional conversion between Sinhala Unicode text and phonetic English.

Two future improvements are suggested. First improvement is enhancement of the decoder algorithm to support inline number to text conversion with appropriate sensing of suffixes used in specifying days, positions and numbers. Second improvement is adding a touching letter (two consonants touched together without al modifier in the first consonant) conversion support as an option which are commonly used in classical Buddhist and Pali literature.

ACKNOWLEDGMENT

This research is funded by the National Science Foundation of Sri Lanka research grant number TG/2014/Tech-D/02.