ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

MODELING IMPROVED AMHARIC SYLLBIFICATION ALGORITHM

BY

NIRAYO HAILU GEBREEGZIABHER

A THESIS SUBMITED TO

THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN COMPUTER SCIENCE

June, 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

MODELING IMPROVED AMHARIC SYLLBIFICATION ALGORITHM

BY

NIRAYO HAILU GEBREEGZIABHER

Signature of the Board of Examiners for Approval

Name	Signature
1. Dr.Sebsbie H/ Mariam , Advisor	_____
2. _____	_____
3. _____	_____

Acknowledgments

I would like to thank my advisor, Dr.Sebsbie H/Mariam, for his support, guidance, understanding and motivation throughout this thesis work. It is also my pleasure to express my gratitude to my friend Solomon Baye and many of my friends who have helped me.

My heartfelt gratitude goes to Dr. Getahun Amare of the Institute of Language Studies, Addis Ababa University, who was always ready to answer my questions regarding Amharic language. My special thanks go to Dr. Mulugeta Seyoum who was helping me in any queries regarding Amharic syllabification, epenthesis and on the evaluation of the algorithm performance.

Table of Contents

Contents **Page**

List of Figures **Page**

List of Tables Page

Acronyms and Abbreviations

GTP (G2P)	Grapheme-to-phoneme
TTS	Text-to-speech
ASR	Automatic speech recognition
NLP	Natural language processing
CV	Consonant Vowel
L2P	Letter to Phoneme
ONC	Onset Nucleus Coda
LHL	Light Heavy Light syllable
LSH	Light Supper Heavy syllable
HSH	Heavy Supper Heavy syllable
HS	Heavy syllable
HL	Heavy light syllable

Abstract

In this paper, a rule-based automatic syllabification Algorithm for Amharic language is designed using linguistic implementation notions such as, Maximal Onset and Sonority Hierarchy principles. Amharic is a syllabic language in which every grapheme represents consonant-vowel assimilation. However, while reading a text in Amharic, all the CV syllables are not uttered as expected and hence the syllables in the text are not the CV sequence seen in the grapheme sequence. Epenthesis and gemination is also a major challenge in Amharic grapheme-to-phoneme conversion because of the failure of Amharic orthography to show epenthetic vowel and geminated consonants. This limits the performance of many speech systems (Amharic text-to-speech and speech recognition) and other natural language applications. After a thorough study of the syllable structure, identification of linguistic syllabification rules and a survey of the relevant literature, a set of rules were identified and used to design the algorithms. Prior success rates of rule-based methods applied to different languages for instance, Spanish, Dutch, Italian, Catalan and Sinhala are the basis of this work. Before designing the syllabification algorithm, the epenthesis algorithm is designed. Moreover, the benefit of syllables to assign stress is pointed out.

The system was implemented and tested using 1000 carefully selected Amharic words found in the language. The result gave rise to 98.1% word accuracy rate, this result shows rule-based syllabification approach is performing very well and the syllabifier for the language can be rule-driven. Although, comparison with data-driven syllabification approach is not performed in this language, rule-based approach showed a higher accuracy rate in the test set.

Key Words: syllabification, rule-based techniques, grapheme-to-phoneme, Maximal Onset Principle, Sonority Hierarchy principles, CV, text-to-speech, speech recognition.

CHAPTER ONE
INTRODUCTION

1.1 Background

A syllable is a basic unit of word studied on both the phonetic and phonological levels of analysis. It is typically composed of more than one phoneme[1]. No matter how easy it can be for people and even for children to count the number of syllables in a sequence in their native language, still there are no universally agreed upon phonetic definitions of what a syllable is. It is phonologically believed that syllable is a complex unit made up of nuclear and marginal elements. There is general agreement that a syllable consists of a *nucleus* that is almost always a vowel, together with zero or more preceding consonants (the *onset*) and zero or more following consonants (the *coda*) but determining exactly which consonants of a multisyllabic word belong to which syllable is problematic. Nuclear elements are the vowels or syllabic segments, and marginal elements are the consonants or non-syllabic segments. With the *rhyme* we find the *nucleus* and *coda*. Standard dictionaries provide syllabification that is influenced by the morphological structure of words; it is common in such dictionaries to split prefixes and suffixes from stems (Tian, 2004). Table 1.1 presents the different parts of syllable structure.

Table 1.1: Syllable structure parts and their description

Parts	**Description**	**Optionality**
Onset	Initial segment of a syllable	Optional
Rhyme	Core of a syllable, consisting of a nucleus and coda (see below)	Obligatory
– Nucleus	Central segment of a syllable	Obligatory
– Coda	Closing segment of a syllable	Optional

A syllable can be described by a series of grammars. The simplest grammar is the phoneme grammar, where a syllable is tagged with the corresponding phoneme sequence. The consonant-vowel grammar describes a syllable as a consonant-vowel-consonant (CVC) sequence. The syllable structure grammar divides a syllable into onset, nucleus and coda (ONC). For instance,

[1] (linguistics) one of a small set of speech sounds that are distinguished by the speakers of a particular language

the Amharic word /ብልሀት/ is a bi-syllabic and it is transcribed as /**bixlhat/** (the transcription scheme used in this thesis work is shown in Appendix A)**.** In this word, the syllable pattern CVC occurs at word initial position and word final position. The word should, therefore, be syllabified into CVC-CVC, /-/ symbol represents the syllable boundary. Alternatively, the word can be syllabified into CV-CCVC but in attempting to syllabify words into their component syllable, one should take into account the syllable structure of the language under consideration. Hence, the first syllable is CVC (**bixl) /b/** is the *onset*, **/ixl/** is the *rhyme*. Since **/ix/** is the vowel it becomes the *nucleus*. The next syllable (**hat**) has the same syllable structure **/h/** is *onset*, **/at/** is the *rhyme* and the vowel **/a/** is the *nucleus*. Figure 1.1 shows the whole syllable structure of the word /**bixlhat/**.

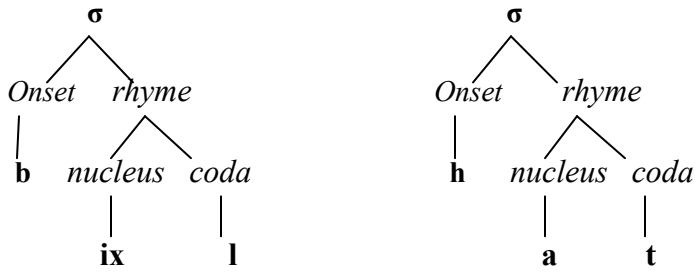

Figure 1.1: Syllable (**σ**) structure for the word (**ብልሀት**) **/bixlhat/** meaning ‗technique'

Some of Amharic grammar books, like (Getahun, 2010) and (Baye, 2010), describe the grammatical syllable structure of Amharic words. There are also some recent works which tries to describe the syllable structure of Amharic words. For instance, (Aster, 1981) and (Mulugeta, 2001) are among those works. These works present the syllable structure of Amharic words and syllabification.

Identification of syllables structures of words play an important role in speech synthesis and recognition apart from their purely linguistic significance. The pronunciation of a given phoneme tends to vary depending on its location within a syllable. While actual implementations vary, text-to-speech (TTS) systems must have, at minimum, three components: a letter-to-phoneme (L2P) module, a prosody[2] module, and a synthesis module. Letter-to-phoneme or grapheme-to-

[2] The patterns of stress and intonation in a language

phoneme conversion is a process which converts a target word from its written form (grapheme) to its pronunciation form (phoneme).

Syllabification can play a role in all the three modules (Bartlett *et al*, 2009). Moreover, in speech recognition syllabification has been used to build recognizer which represents pronunciations in terms of syllables rather than phonemes. In addition, syllabification can help annotate corpora with syllable boundaries for corpus linguistics research (Jurafsky and Martin, 2006).

There are two broad approaches to handle automatic syllabification: *rule-based* and *data-driven*. The traditional approaches to automatic syllabification have been *rule-based* (or knowledge-based), implementing notions such as the **maximal onset principle** and **sonority hierarchy principle**, including ideas about what constitute phonotactically legal sequences in the coda. An alternative to the rule-based methodology is the *data-driven* (or corpus-based) approach, which attempts to infer ~~"n~~ew" syllabifications from an evidence base of already-syllabified words (a dictionary or lexicon), i.e., the corpus acts as the gold standard. Data-driven methods are therefore based on machine learning. There is a small existing literature on data-driven syllabification (Marchand, 2009). The work by (Müller, 2000) describes a hybrid (partly rule-based and partly data-driven) approach in which a form of the expectation-maximization (EM) algorithm is used to cluster example data in three- and five-dimensional syllable classes. The three-dimensional data are onset, nucleus and coda; the five-dimensional data add position of syllables in the word and stress type.

When we explore attempts on Amharic, there is only single published work done so far on modeling automatic syllabification algorithm which is presented by (Sebsbie *et al*., 2004). In this work the researchers have used syllables for the Amharic speech synthesizer.

Since syllabification is helpful in speech synthesizer and speech recognition, the work by (Solomon and Menzel, 2007) presents syllable-based speech recognition for Amharic. In this work the researchers reported 90.43% word recognition accuracy and they have stated that there is a possibility of performance improvement. Furthermore, they have stated that CV syllable is a promising alternative in the development of automatic speech recognition for Amharic which is a motivation for this research work on automatic syllabification for Amharic.

In this paper by using the previous works as benchmark, we tried to see the syllable structure of the language in accordance with acoustic evidences. Moreover, the study looks into the possibility of having automatic syllabification algorithm for Amharic. We also tried to see the epenthesis, gemination and phonetic characteristics of Amharic words in relation with syllable structure.

1.2 Statement of the problem

Automatic syllabification, as it described in Section 1.1, has applications in automatic speech recognition, text-to-speech (TTS) system, and in other natural language processing applications. Syllabification in TTS system is important for two reasons: first, it helps the implementation of letter-to-phoneme rules such as the diphthong[3] generation. Second, syllabification is essential in enhancing the quality of speech produced by synthesizers, since detecting the syllable will help in using them to model phone durations and as carriers of certain acoustic traits like intensity and duration to improve the synthesized speech intonation. Syllabification is also useful in Automatic Speech Recognition (ASR). In speech recognition, syllabification has been used to build recognizer which represents pronunciations in terms of syllables rather than phonemes.

Amharic is a syllabic language in which every of the grapheme (character) represent consonant-vowel assimilation. However, while reading a text in Amharic, all the syllables are not uttered as expected and hence the text syllables are not the CV sequence seen in the text. This limits performance of many speech systems and other NLP applications for Amharic language. A great number of diverse algorithms have been proposed for syllabification in different languages and many researchers have been done on other languages. However, only single published work (Sebsbie *et al*., 2004) has been done on automatic syllabification algorithm in Amharic and its syllable structure. This paper is based on this published article on the only algorithm on Amharic syllabification. Thus, the purpose of this work is to model improved algorithm for classifying Amharic words into syllables, and to share this algorithm so that other researchers in the area of Amharic speech and language processing will use the work. Moreover, this paper looks into epenthesis, gemination, the acoustic evidence (phonetic characteristics) and propose a better model for the syllabification of Amharic text.

[3] A vowel sound that starts near the articulatory position for one vowel and moves toward the position for another

1.3 Objectives

1.3.1 General objectives

The general objective of this research work is to investigate the possibility of having appropriate syllabification model for Amharic text.

1.3.2 Specific objectives

The specific objectives of this research are:

- Review and study the works done so far in the area of automatic syllabification.
- Study Amharic syllable structure and identify rules for automatic syllabification.
- Collect corpus for testing the system performance.
- Study the property of Amharic phonemes in words by looking into the acoustic evidence of the language, to identify the phonetic characteristics of Amharic syllables.
- Asses the various approaches for the development of automatic syllabification in the language.
- Design and model new syllabification algorithms for the language.
- Develop a prototype system for the automatic syllabification in the language.
- Test and analyze the system performance using the collected corpus.
- State conclusion and recommendation based on the experimental results.

1.4 Methodology

In this study, first we thoroughly studied the syllable structure and linguistic rules for syllabification of Amharic words in parallel with a survey of relevant literature. A set of rules were identified and implemented. Moreover, we reviewed linguistics literature to gather the views of scholars from the various linguistic traditions.

1.4.1 Data collection

Amharic words were collected from literatures and Amharic dictionary to study the phonetic characteristics of Amharic phonemes while designing the algorithm. The sonority constraint on syllable structure says that the nucleus of the syllable must be the most sonorous phone in a sequence (the **sonority peak**), and that sonority decreases monotonically out from the nucleus (toward the coda and toward the onset) (Jurafsky and Martin, 2006). Therefore, the collected

data is recorded to look into the acoustic evidence (the waveform or intensity of the sound). Such evidence helps in knowledge acquisition (identifying rules) phase.

The collected data includes words with consonant cluster and gemination at different positions (word initial, media and final). For instance, to study epenthesis vowel insertion we need parallel corpus which contains words with consonant clusters in its phoneme sequence. The other kind of collected data is Amharic words with gemination, to test gemination handling in epenthesis module. Such words were identified by the researcher in consultation with language experts and from literatures. Moreover, the data includes Amharic words with different syllable patterns to test the syllabification algorithm. The specified data were collected from different sources, such as Amharic dictionaries and literatures on syllabification. Hence, a total of 1000 words were collected.

1.4.2 Modeling methodology

There are essentially two possible approaches to automatic syllabification, namely: rule-based and data-driven as it is discussed earlier in Section 1.1. For this study, we have chosen rule-based modeling approach with acoustic evidence. Acoustic evidence is used as rule corrector for both grapheme-to-phoneme conversion and syllabification.

We have selected this approach since it was demonstrated in different languages and has better results than data-driven approaches. To implement the data driven approach it requires large corpus, corpus which consists of already syllabified Amharic words to attain a better accuracy. Moreover, rule-based approach is flexible and fast and the more complete rules produce the better accuracy.

1.4.3 Tools and techniques

In conducting this research work, we have used appropriate tools that can help the study. KTH Wavesurfer, Free Pitch Marker Tool and Praat free speech analyzer tool are used to study the phonological characteristics of words in corpus. We have also used audio editor software such as, Speech Analyzer. Implementation is done using Visual Studio C# programming language. The tools are used as the researcher is convenient with and they are all appropriate for the research.

1.4.4 Analysis and Evaluation

To evaluate the performance of the proposed model, results are computed using word accuracy. Word accuracy is the percentage of words syllabified by the method in exactly the same way as is given by the experts (linguistic experts manually checked the syllabified corpus by the system and put remark on each word in the corpus).

For example, the word (በኃላ) - /behwala/ has six junctures: /b*e*h*w*a*l*a/. If the syllabification according to the rule of the language and experts is /be-hwa-la/ and if the algorithm syllabifies the word as /be-hwal-a/, this is considered entirely wrong in terms of word accuracy. Furthermore, we will also check the accuracy of the epenthetic vowel insertion. The entire data is used to test the algorithms and after analysis was made on the result, word accuracy is generated in percentage and also the distribution syllable patterns are presented.

1.5 Application of Results

Automatic syllabification has varieties of applications like text-to-speech, automatic speech recognition, corpus statistics and also in other NLP applications. The result of this work will improve the performance of different applications in the area of speech and other NLP applications for Amharic language, by incorporating the result with those applications. For instance, in speech recognition for Amharic language instead of representing pronunciations in terms of surface form of the grapheme sequence in each word, we can use the actual phoneme sequence or syllables from the result of this research work, so that we improve the attainable performance. Epenthetic vowel occurs in Amharic language frequently; therefore this work used in the preparation of pronunciation dictionary in handling of epenthetic vowel in ASR. Moreover, in Amharic TTS it can be used in grapheme-to-phoneme (G2P) module and in constructing a pronunciation dictionary.

1.6 Scope of the Study

In this thesis work we only cover the automatic epenthesis and syllabification algorithm for Amharic language. Stress prediction, automatic gemination analysis are not covered in this work. Though gemination is important for syllabification in Amharic language we did not work on automatic gemination because of the time limit, rather we have used corpus prepared by

language experts which contains gemination whenever it is necessary, but our algorithm is designed to handle epenthetic vowel insertion. In this work, it is tried to cover all the syllable templates existing in the language and convert into easily understandable algorithm.

1.7 Organization of the Thesis

This paper is structured as the following: Chapter 2 presents literature review and related works on automatic syllabification. In Chapter 3, we present syllable structure and syllabification of Amharic language. The design of automatic epenthesis and automatic syllabification algorithms for Amharic is presented in Chapter 4. The Test results are shown and discussed in Chapter 5. In Chapter 6, conclusions and future works are pointed out.

CHAPTER TWO

AUTOMATIC SYLLABIFICATION

This chapter presents review of literature and related works on automatic syllabification. The chapter begins with a brief introduction to syllabification and to the two major approaches used for automatic syllabification. Presenting further discussions on various approaches being tested in syllabification, the chapter gives detailed account on rule-based approach which focuses on implementing notions such as the *maximal onset principle* and *sonority hierarchy*. The chapter ends up with presentation of related works on automatic syllabification of words which has been done so far on diverse languages using the different approaches.

2.1 Literature Review

2.1.1 Syllabification

Syllabification is the task of segmenting a sequence of phonemes into syllables. A syllable is a unit of sound composed of a central peak of sonority (usually a vowel), and the consonants that cluster around this central peak. As we have seen in Chapter One, syllabification has importance in a variety of speech applications. For instance, in speech synthesis, syllables are important in predicting prosodic factors like accent. The realization of a phone is also dependent on its position in the syllable (onset is pronounced differently than coda). In speech recognition syllabification has been used to build recognizers which represent pronunciations in terms of syllables rather than phonemes (Jurafsky & Martin, 2006), it also helps to detect out of vocabulary words.

Syllabification includes the separation of a word into syllables, whether spoken or written. Speech is organized into syllables. Although nearly everybody can identify syllables, almost nobody can define them. It is difficult to state an objective procedure for locating the number of syllables in a word or a phrase in any language. There are words difficult to be agreed upon in determining the number of syllables contained, but it is important to remember that there is no doubt about the number of syllables in the majority of words. Syllable is a unit larger than a single segment and smaller than a word, and this characteristics can be described from both a phonetic and a phonological point of view, one of which is distinguished from the other, although the differentiation is not yet agreed upon by all scholars (Duanmu, 2008).

From phonological standpoint syllable is a conventional unit which is a group of sounds that constitute the smallest unit of the rhythm of a language. These phonological syllables differ from language to language. In English, for example, it is theoretically possible to make a single syllable as CCCVCCCC (Duanmu, 2008), where previous studies related to the syllable structure of the standard Amharic have shown that the following syllable types V, VC, VCC, CV, CVC and CVCC occur as part of the phonological system of Amharic (Aster, 1981) and (Mulugeta, 2001). The syllable, in this view, is considered as an important abstract unit explaining the way vowels and consonants are organized within a sound system. Technically, the basic elements of the syllable are the **onset** (zero or more consonants) and the rhyme. The **rhyme** (sometimes written as ‗rime') consists of a vowel, which is treated as the nucleus, plus any following consonant(s), described as the **coda** (Yule, 2006). The internal organization of syllables characterized as in Figure 2.1.

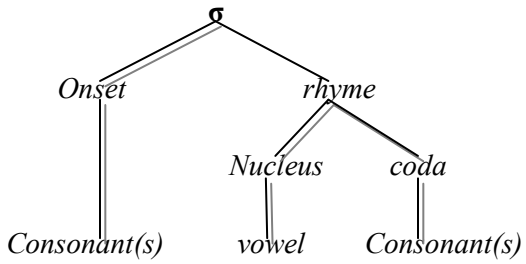

Figure 2.1: Syllable structure **σ** -syllable

Some studies have included basic generalization as follows: all languages have syllables with onsets; many languages require all syllables to have onsets in surface representation; no language requires all syllables to have codas. Each syllable has a nucleus, and language-particular conditions govern the class of possible onsets and codas (Fujisaki, 1995).

Languages differ considerably in the syllable structures that they permit. For most languages, syllabification can be achieved by writing a set of declarative grammatical rules which explain the location of syllable boundaries of words step-by-step. It has been adhered to the well known principles *"the Maximum Onset Principle"* and *"the sonority hierarchy principle"*. In the following subtopics we will give details on these principles.

a) Sonority Hierarchy Principle (SHP)

The most notable phonological principle which comes into play here is known as the Sonority Hierarchy Principle, which governs the shape of onsets, nucleus and codas. Sonority is related to the difference between sonorants (sounds which are typically voiced, like approximants, nasal stops and vowels) and obstruents (oral stops and fricatives, which may be either voiced or voiceless). Sonorants are more sonorous, that is, their acoustic properties give them greater carrying power. If we stood at the front of a large room and said one sound as clearly as we could, a listener at the back would be much more likely to be able to identify a highly sonorous sound like /ix/ [ɨ] than a sound at the other end of the sonority range, such as /t/ [ት].

The general rule expressed by the Sonority Hierarchy Principle is that syllables should show the sonority curve which means the nucleus constitutes the sonority peak of the syllable, with sonority decreasing gradually towards the margins (syllable boundaries). Example for the word ምልክት - /**mixlkixt**/ as shown on Figure 2.2, the onset /**m**/ rises to the nucleus, since /**ix**/ is the vowel it has high sonority in the curve and the sonority decreases towards the coda /**l**/. Again the other onset /**k**/ rises to the other nucleus /**ix**/ (high peak) and falls towards the coda /**t**/. As a broad generalization, we can say that onsets must rise in sonority, and codas must fall in sonority and the nucleus becomes the peak of the curve.

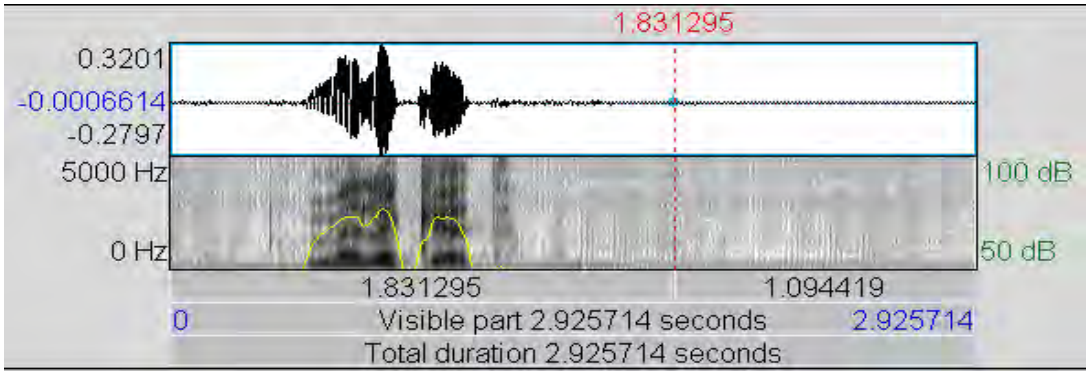

Figure 2.2: Waveform and sonority curve for the word ምልክት - /**mixlkixt**/ meaning ‗sign'

b) Maximal Onset Principle (MOP)

Sonority Hierarchy used as one guide for drawing syllable boundaries; there is another, equally important principle governing syllable division, namely Onset Maximalism (also known as Initial Maximalism or Maximum onset principle), which is stated out as follows:

Where there is a choice, always assign as many consonants as possible to the onset, and as few as possible to the coda. However, remember that every word must also consist of a sequence of well formed syllables.

Onset Maximalism tells us that, in a word like /mefelfeya/ meaning (_pulper‘), the medial /l/ must belong to the third syllable –lfe- , where it can be located in the onset, rather than the second syllable, where it would have to be assigned to the less favoured coda but here we should also consider the other guiding principle namely sonority hierarchy. According to the waveform shown in the Figure 2.3 for the word /mefelfeya/ the sonority for consonant /l/ is decreasing starting from the previous vowel /e/ (with high sonority) and the next consonant /f/ goes up towards the next vowel /e/ with high sonority. In this case if we directly apply maximum onset principle without considering the sonority hierarchy of the consonant cluster /lf/ it leads us into wrong syllable because, the cluster should belong to the third syllable /me-fe-**lf**e-ya/ to assign more consonants to the onset of the third syllable. The same goes for a word like /wenber/ (_chair‘), where both parts of the medial /nb/ cluster belong to the onset of the second syllable, while the initial CV forms a syllable on its own. For such CVCCVC words CVC-CVC, CV-CCVC or CVCC-VC are possible syllabifications but according to the onset maximalism principle the correct one is CV-CCVC Figure 2.4 shows the waveform for the word /wenber/.

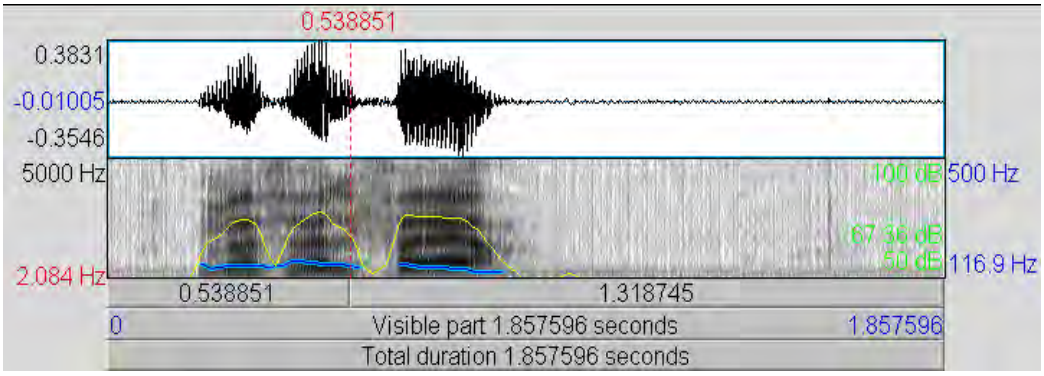

Figure 2.3: Waveform for the word መፈልፈያ– /mefelfeya/ meaning _pulper‘

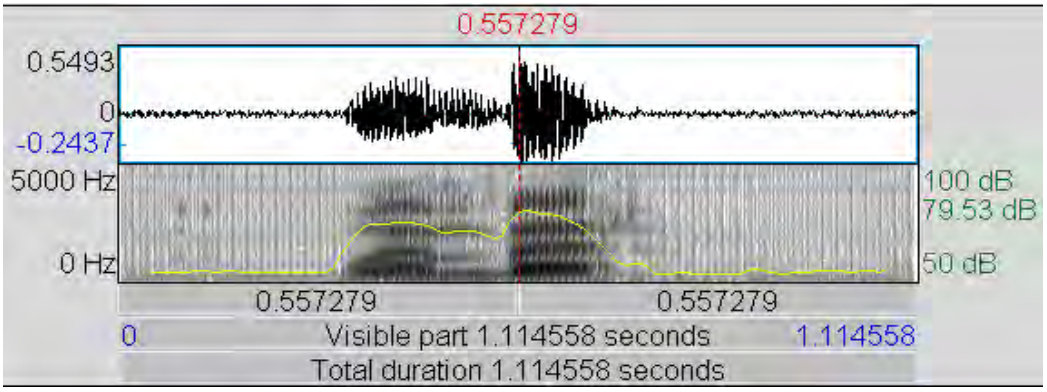

Figure 2.4: Waveform for the word ወንበር - /wenber/ meaning _chair‘

There are many other guiding principles for division of words into syllables. For example according to (Aster, 1981) if there are cluster of two consonants in Amharic word position one of the consonants, i.e the first member of the consonant cluster most likely belongs to the preceding syllable and the second member belongs to the following syllable. In advance, she stated that in attempting to divide words in to their component syllable, one should take into account the syllable structure of the language. For instance, in the Amharic word (ዐድባር) /axedbar/ if we just directly apply maximum onset principle the syllabification will be CV-CCVC /axe-dbar/, more codas to the onset of the second syllable CCVC. But, the correct syllabification for this word is CVC-CVC /axed-bar/ according to the syllable structure of the language, /ax/ is digraph for the glottal consonant [ዕ].

Once we have the syllable templates of the language the given word can be syllabified into different possible sequences of syllables; but there is only single legal sequence of syllables in the given word. For instance, in English word like falter /foltor/, we cannot straightforwardly assign the medial /lt/ to the second syllable. The Sonority Hierarchy Principle would allow the syllable boundary to follow /lt/ (compare fault, a well-formed monosyllabic word), but Onset Maximalism forces the /t/ at least into the onset of the next syllable. The syllable boundary cannot, however, precede the /l/ because /lt/ is not a possible word-initial cluster in English, and it consequently cannot be a word-medial, syllable-initial cluster either. On the other hand, in bottle if it is pronounced as /bottle/ our immediate reaction might be to propose bo-ttle, which fits both the Sonority Hierarchy Principle and Onset Maximalism. However, we then face a problem with the first syllable, which would on this analysis consist only of /bo/; and a single

short vowel cannot make up the rhyme of a stressed syllable in English. Therefore, the first syllable clearly needs a coda and it is syllabified as /bot-tle/.

2.1.2 Approaches to automatic Syllabification

There are a number of researches done so far related to automatic syllabification. Those works implement different approaches to study the structure of syllables and for automatic syllabification of words in the language. In this Section we present the rule-based and data-driven approaches.

a) Data-driven methods

Data-driven (or corpus-based) approach is an alternative approach in syllabifying words, which attempts to infer ~~"n~~ew" syllabifications from an evidence based on already-syllabified words (a dictionary or lexicon), i.e., the corpus acts as the gold standard. Data-driven methods are therefore based on machine learning. There are few research done so far on syllabification using this approach (for example see (Bartlett *et al.,* 2008), (Brigitte Bigi *et al.*, 2009) and (Bartlett *et al*., 2009)). Although there are few literatures in data-driven approach, in this paper we will see some of data-driven approaches namely Look-up procedure and Decision tree-based approach.

i) Look-up Procedure

The Look-Up Procedure was also originally used for grapheme to phoneme transcription. It has since been modified to perform automatic syllabification (Marchand *et al.,* 2009). This method uses N-grams (each consisting of a left context, right context and central letter) to learn and determine syllable boundaries.

During training, an N-gram (a total number of phonemes that can be considered as a focus phoneme) is generated for each possible syllable boundary location in a word. Each N-gram is stored in a table along with how often a syllable occurs and does not occur following the central letter. Table 2.2 shows the table entries for the word s*y*l|l*a|b*l*e, using a left and right context of three letters (N = 7).

As it is shown in Table 2.2 the central letter is taken and to the left of that letter is considered and if there is a syllable boundary the frequency to the syllable boundary mark (|) is set to 1and 0 to the other juncture (*). During testing, the closest matches to the N-grams from the test words are found in the table. For a given N-gram, if the value of a syllable boundary occurring after the

central letter is 1 and the value of no syllable boundary is 0, a syllable boundary is placed in the test word.

Table 2.1: Table entries for the word –s*y*l | l*a | b*l*e" used by the Look-Up Procedure

N-grams	Syllable boundary information	
	| (syllable boundary)	* (no syllable boundary)
---***s****yll*	0	1
--*s****y****lla*	0	1
-*syl****l****lab*	1	0
syl*labl*	0	1
*yll****a****ble*	1	0
*lla****b****le-*	0	1
*lab****l****e--*	0	1

For example, using the 7-grams stored in Table 2.1 to syllabify the word –able" requires finding the closest match to each of four 7-grams *(---able, --able- and -able--)* within the table. The value of a syllable boundary occurring after the central letter in this pattern is greater than the value of no syllable boundary and therefore a syllable boundary is placed following the /a/ in –able".

The Look-Up Procedure was tested in the comparison of automatic syllabification methods for English (Marchand *et al*., 2009).

ii) Decision tree-based syllabification

Decision tree-based syllabification is another type of data-driven approach; a separate decision tree is trained for each of the different phonemes. The ONC (Onset Nucleus Coda) tag for a phoneme is obtained by –asking" a series of questions about the context of the phoneme in question as defined by the corresponding decision tree. A decision tree is composed of a root node, internal nodes and leaves. In the trees used here, the context is defined by the neighboring phonemes. Each node contains information about the attribute and ONC identity.

In the decoding phase, a ONC tag sequence is generated by going through the pronunciation phoneme by phoneme from left to right. The decision tree corresponding to the current phoneme is climbed based on the context information until a leaf is reached. The ONC tag that

corresponds to the current phoneme is read from the leaf. Then the process moves on to the next phoneme and the ONC tag for this phoneme is found in a similar way.

Example: training word: /**kixf-fixtt**/ and if it is syllabified as /**kixf-fixtt**/ this turn produce ONC-ONCC. (W)- represents the starting and ending of a word. The central phoneme is taken as a focus character.

Phoneme sequence	Classification
W k ix	O
k ix f	N
ix f f	C
f f ix	O
f ix t	N
t t W	C

When a decision tree is trained for a given phoneme, all the training cases for the phoneme are considered. A training case for the phoneme is composed of the phoneme context and the corresponding ONC tag of the pronunciation. During training, the decision tree is grown and the nodes of the decision tree are split into child nodes according to an information theoretic optimization criterion. Details about decision tree training can be found in (MacKinney, 2006) and (Quinlan, 1993).

b) Rule Based

Another alternative approach rather than the data-driven methodology is rule-based approach. The rule-based method effectively embodies some theoretical position regarding the syllable. It uses implementing notions such as *maximal onset principle* and *sonority hierarchy,* those principles we have already seen in the previous sections, Section 2.1.1.1 and 2.1.1.2.

Rule-based approaches have traditionally been the preferred method of determining the syllabification of unknown words (for example, see (Kishore and Black, 2003), (Oliveira *et al.*, 2005), (Libossek and Schiel, 2000), (Beck *et al*, 2009), (Bigi *et al.*, 2009) and (Burileanu and Negrescu, 2006). Once defined by linguists, rules are straightforward to implement and apply. However, this approach is often time-consuming and requires expert knowledge. More importantly, several studies have raised the question of whether rule-based methods are actually the best approach to these tasks given the high performance of data-driven methods (Daelemans

and van den Bosch, 1992), (Adsett and Marchand , 2007) and (Marchand *et al*., 2009). In particular, it has been demonstrated that, for syllabification of English words, data-driven methods perform significantly better than rule-based methods (Marchand *et al*., 2009). The success of data-driven methods on this language may be due to the fact that English is a language with a complex and irregular syllable structure (Bruck *et al.*, 1995), (Conrad and Jacobs, 2004) which is challenging (and perhaps impossible) to fully capture with traditional linguistic rules. Italian and Spanish are considered to exhibit simple syllablic structure (Conrad and Jacobs, 2004), (Bartlett *et al.,* 2008). However, according to (Adsett and Marchand , 2007), when a syllabification procedure is included as a component of a TTS system, a data-driven method is a more appropriate choice than a rule-based approach, even for languages with low syllabic complexity. Unfortunately, to cope with statistical syllabification of words, these techniques are effective only if a syllabified corpus exists for the training. More difficulties arise when the task is the syllabification of spoken data and casual speech (Bigi *et al.*, 2009). On the other hand, papers like (Weerasinghe *et al*., 2005), (Beck *et al*, 2009), (Bigi *et al.*, 2009) used rule-based approach. For instance, in the work by (Weerasinghe *et al*., 2005), they have tested their algorithm with 30,000 distinct words in a corpus and compared with the same words that are manually syllabified. The algorithm performs with 99.95 % accuracy defined as the proportion of correctly syllabified words.

2.2 Related works

In this section, different works which are related with automatic syllabification algorithms are presented. The papers presented are not only on Amharic language rather different works in other languages are also selected and presented based on the relevance and the similarities of the papers with this thesis work.

2.2.1 Automatic syllabification Algorithm for Amharic

The first attempt on automatic syllabification of Amharic words is done by (Sebsbie *et al.,* 2004). In this work, the researchers tried to describe the issues to be considered in developing a concatenative speech synthesizer for Amharic language and, as a component they worked on automatic syllabification algorithm for Amharic words. They have used syllables while they are

creating their speech database for the speech synthesizer. While doing on the syllabification they have identified and presented 8 basic syllabification rules of the language and they developed a recursive algorithm to identify the set of syllables in Amharic words.

Regarding the words which need epenthesis vowel they mentioned only one rule which works on a word consisting of only CC phonemes, in which their algorithm inserts the epenthetic vowel and form a monosyllable word CVC. Moreover, regarding to stress they used simple stress pattern of 1 (primary stress) for initial syllable and 0 (secondary stress) for all of the remaining syllables in the word.

In this work, it is defined a transliteration scheme to work with Amharic scripts and incorporated Amharic phone set, syllabification rules, letter to sound rules into Festvox.

For the training they have used a corpus consists of a total of 29,480 diphone instances made up of 801 unique diphones. Their corpus consists of a total of 12,724 syllable instances and 1317 unique syllables. From these unique syllables high frequency syllables cover 70% of the total distribution. Moreover, among the 12,724 syllable instances: 316 are monosyllables, 3752 are front syllable (word initial), 4904 are middle syllable (word middle) and 3752 are back syllable (word final).

The researchers showed their perceptual evaluation of Amharic voice. The result shows the average score of the Amharic synthesizer is 2.9 out of 5 (which is categorized as good). Though the work implemented automatic syllabification, the performance of the syllabifier is not reported in the paper and as a limitation gemination is not considered in the syllabification algorithm. The performance of the epenthetic vowel is also not reported and it is not handled fully.

In conclusion, the researchers gave detailed account on the issues to be considered to improve the performance of Amharic synthesizer. One component they put as future work is on the syllabification of consonant clusters (e.g **ትምህርት** –education", **ትንኝ** –gnat", **ምልክት** –sign") where epenthetic vowel is used to split those impermissible words according to the basic rules of the language. There is scope for further improving the algorithm in handling of epenthesis, gemination handling and in the overall performance of the syllabification algorithm.

2.2.2 A Rule based Syllabification Algorithm for Sinhala

There are different papers which implement rule-based approach for automatic syllabification. (Weerasinghe *et al.,* 2005) presented a study of Sinhala (Indic language spoken by the people of Sri Lanka) syllable structure and an algorithm for identifying syllables in Sinhala words. Before they develop the algorithm, they studied the syllable structure and linguistic rules for syllabification of words in this language. They have also made a survey of the relevant literature. A set of rules were identified and implemented as an algorithm. They identified a total of eight basic syllabification rules in the language. They have used linguistic principles for driving those rules, namely maximum onset and sonority hierarchy which are the well known principles in linguistics for syllabification.

The researchers tested their algorithm on 30,000 distinct words extracted from the Sinhala Corpus. They compared the output with correctly hand syllabified words. The corpus is obtained from the category of news paper, feature articles and others. The authors tried to make the dataset heterogeneous and hence they perceived better representation of the language in this section of the corpus. In addition, they extracted a list of distinct words, 30,000 most frequently occurring words chosen for testing the algorithm. The 30,000 words yielded some 78,775 syllables excluding the word final syllable of each word. The algorithm achieved an overall accuracy of 99.95% when compared with the same words manually syllabified by an expert. They have made an error analysis and their analysis revealed the following two types sources of errors:

> *1. Words composed by joining two or more words (i.e. single words formed by combining two or more distinct words such as in the case of the English word "thereafter"). In this case, syllabification needs to be carried out separately for each word of the compound word, and then concatenated to form a single syllabified word.*
>
> *2. Foreign (mainly English) words directly encoded in Sinhala.*

2.2.3 Automatic detection of syllable boundaries in spontaneous speech

This work is done by (Bigi *et al.*, 2009) for detection of syllable boundaries in spontaneous speech of French language. The proposed solution deals with the syllabification of a phoneme sequences. Like the works presented in previous sections, this work also implemented rule-based approach. Their Rule-based System (RBS) phoneme-to-syllable segmentation system is based on 2 main principles:

1. a syllable contains a vowel and only one.

2. a pause is a syllable boundary. The CID (Corpus of Interactional Data) corpus was first automatically segmented into inter-pausal units (IPU) delimited by silent pauses of 200 ms and more. Human transcribers marked shorter pauses they perceived. In both cases, a pause signals a syllable boundary.

They used grouping phonemes into classes for establishing rules dealing with the grouped classes. Subsequently, they defined general rules followed by a small number of exceptions. They proposed the following classes for the all phonemes exist in the language: V for Vowels, G for Glides, L for Liquids, O for Occlusives, F for Fricatives, N for Nasals and they assigned those French phonemes into their corresponding classes.

The following tables (Table 2.2 & Table 2.3) show the general syllabification rules and additional exception rules identified by the authors for French language respectively. To derive the rules they used maximum onset principle and sonority hierarchy principles. The letter X refers to one of the five phoneme classes: G, L, O, N or F (i.e., a non-vowel phoneme).

The paper gives detail account on how the algorithm is implemented. They have used the program *LPL-Syllabeur*[4] which is implemented in Java 1.6 and tested under Linux and windows®. The object of the *LPL-Syllabeur* output files are: the syllable start time (which is the start time of its first phoneme), the syllable end time (which is the end time of its last phoneme), and the syllable string label (which is the phoneme string labels concatenation).

The researchers used test corpus which is 1.6% of the CID (Corpus of Interactional Data). The CID is an audio-visual recording of 8 hours of spontaneous French dialogues (1 hour of recording per session). 1.6% of CID corpus is about 7 minutes of a dialogue, which represents 653 words for speaker 1 and 1,238 words for speaker 2. This test corpus contains 2,068 syllables.

[4] The Syllabeur 'syllabifier' developped at LPL (Laboratoire Parole et Langage)

Table 2.2: General Rules

	Observed Sequence	Segmentation Rule
1	VV	V-V
2	VXV	V-XV
3	VXXV	VX-XV
4	VXXXV	VX-XXV
5	VXXXXV	VX-XXXV
6	VXXXXXV	VXX-XXXV

Table 2.3: Exception Rules

	Observed Sequence	Segmentation by the Exception rule	Segmentation by the general rule
1	VXGV	V-XGV	VX-GV
2	VFLV	V-FLV	VF-LV
3	VOLV	V-OLV	VO-LV
4	VFLGV	V-FLGV	VF-LGV
5	VOLGV	V-OLGV	VO-LGV
6	VOLOV	VOL-OV	VO-LOV

According to the performance report the test corpus was manually segmented by two experts, and 23 boundary mismatches are observed (this means 46 different syllables). A boundary mismatch means that one expert does not propose the same segmentation between two syllables as the other expert. Therefore, the syllable agreement rate is 97.77% (2022 over 2068). The authors agreed on that improvement is when the automatic syllabification is nearest to the manual one. Finally, the rules identified and the algorithm developed by the researchers seems to be well-adapted to the syllable boundary detection in the specified corpus.

2.2.4 Automatic Word Stress Marking and Syllabification for Catalan TTS

This paper presents three linguistically rule-based automatic algorithms for Catalan text-to-speech conversion: a word stress marker, an orthographic syllabification algorithm and a phonological syllabification algorithm. The work is done by authors (Rustullet *et al*, 2008) in Microsoft Language Development Center.

In this work the researchers tried to address both the orthographic and phonological syllabification algorithm. Moreover, they included stress marker in their work. For the

orthographic syllabification the authors listed a set of linguistic rules for the Catalan language. In the same way they identified the rules for phonological syllabification too.

To discuss the general architecture of their algorithm: for the orthographic syllabification they designed a pipeline where the input orthography is processed and gives the final syllable boundary marked orthography. Similar pipeline is also designed for the phonological syllabification. The stress marker exits in both orthographic syllabification pipeline and the pronunciation syllabification pipeline. Moreover, after the orthographic syllable marker is processed in the pipeline the output can be the input for the phonologic stress marker.

The major modules exist in the pipelines are vowels parser, digraphs parser (exist only in the orthographic stress and syllable marker), glides parser, sonority scale and the pronounceable groups identifier. In the end, two outputs are expected: an orthographic output with stress and syllable boundaries and a phonologic one, only with syllable boundaries. In this work, symbols that help to identify each class of phonemes are used in the algorithm. For the orthographic syllabification the researchers identified a total of 16 rules and used them in their algorithm.

The algorithms were implemented and tested using selected words from the language. The researchers carried out two tests in order to assess the performance of the proposed algorithms: test 1 used 1000 words randomly selected from a Catalan phonologically transcribed lexicon and test 2 used 223 words (newspaper text). This lexicon does not include foreign words, abbreviations, or acronyms. According to the results report, the result gave rise to the following word accuracy rates: 100% for the stress marker algorithm, 99.7% for the orthographic syllabification algorithm and 99.8% for the phonological syllabification algorithm.

Based on the test carried out the researchers come up to the conclusion that, the stress and syllabification can be rule-driven in languages which have a phonologic-based orthography. We can see that the rule-driven approach performs nearly 100% for the orthographic syllabification for this language in the specified test set.

2.2.5 Automatic Syllabification for Danish Text-to-Speech Systems

Prior success rates of rule-based methods applied to Portuguese and Catalan syllabification modules were on the basis of this work. The paper is done in Microsoft Language Development Center Portugal by the researchers (Beck *et al.*, 2009). In this paper, a rule-based automatic syllabifier for Danish is presented. The authors used Maximal Onset Principle to derive the syllabification rules and automatic phonological syllabification algorithm for Danish words is presented. The general architecture of the algorithm is shown in Figure 2.5.

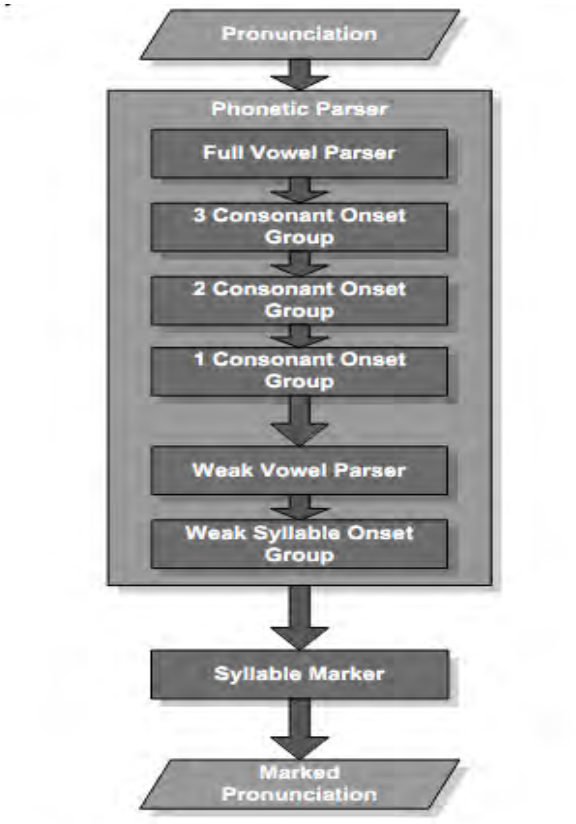

Figure 2.5: Automatic syllabification algorithm architecture for Danish language

The input for the syllabification algorithm is a phonetic representation rendered in ASCII characters from a 100k Danish lexicon without syllable boundary information. The parser module is activated and it tags (in order) full vowels, valid three, two and one consonant clusters, weak vowels and weak vowel onset groups. Syllable boundaries are introduced according to the

specifications of the algorithm. The expected output is an identical phonetic representation as input but including syllable boundary markers. They identified and present the rules for automatic syllabification of Danish specifically 9 rules for syllabification of full vowels and one rule for syllabification of weak vowels.

In addition to the rule-based approached they have also tested the Danish automatic syllabification using Artificial Neural Network (ANN), which is one of the data driven approach for automatic syllabification. The authors preferred this approach for a comparison with rule-based approach. The algorithm was tested in order to assess the performance of ANN for automatic syllabification. Test 1 corpus set consisted of 1000 randomly selected word forms from a phonetically annotated lexicon of 100k words in Danish, excluding abbreviations, acronyms. Test 2 corpus set consisted of 1000 words (604 unique words) from running text extracted from a Danish newspaper article. Words in the test set were first phonetically transcribed in order to serve as input of the rule-based syllabifier. Both the results of Test 1 and Test 2 were manually checked by a Danish Linguist. The result rises to 96.9% and 98.7% of word accuracy rate respectively for each test set. The result seems to prove that syllabification can be rule-driven in languages like Danish with more syllabic complexity. The authors used the same test corpus set and tested using an ANN based syllabification system adapted to Danish and the results showed a lower accuracy rate: 94.1% and 94.5% respectively for each test set. Rule-based approaches have been proven to be computationally lighter than data-driven methods and equally or more efficient.

2.2.6 A Syllabification Algorithm for Spanish

(Cuayåhuitl, 2004) presents an algorithm for dividing Spanish words into syllables. This algorithm is based on grammatical rules of the language which were found in the literature and converted into algorithm. The researcher categories the Spanish letters as either vowels (a,e,i,o,u) or consonants (b,c,d,f,g,h,j,k,l,ll,m,ň,n,p,q,r,rr,s,t,v,w,x,y,z), and the vowels are further classified as either weak (i,u) or strong (a,e,o). Letters like ch, ll, and rr which exists in the Spanish are considered as single consonants. In order to illustrate the syllabification process the author proposes the following step by step algorithm:

1. Scan the word from left to right

2. If the word begins with a prefix, divide between the word and the prefix

3. Ignore one or two consonants if they begin a word

4. Skip over vowels

5. When you come to a consonant, see how many consonants are between vowels

a) If there is only one, divide to the left of it;

b) If there are two, divide to the left of the second one, but if the second one is l or r, divide to the left of the first one;

c) If there are three, divide to the left of the third one, but if the third one is l or r, divide to the left of the second one;

d) If there are four, the fourth one will always be l or r, so divide before the third consonant.

6. If the consonant ends the word, ignore it

7. Scan the word a second time to see if two or more vowels are together

a) If two vowels together are both weak, ignore them;

b) If one of the vowels is weak, ignore it, but if the u or i has an accent mark, divide between the two vowels;

c) If only one of the vowels is weak, and there is an accent mark which is not on the u or i, ignore them;

d) If both vowels are strong, divide between the vowels;

e) If there are three vowels together, ignore them if two of them are weak even if there is an accent mark; if two of the three vowels are strong, separate the two strong vowels if they are side by side

The author collected 316 words ranging from one to six syllables from other previous works for evaluation of the algorithm. He performed the evaluation with simple division between the number of correctly syllabified words and the total number of words. The overall syllabification accuracy became 98.4% on the specified test corpus. Though his algorithm scores higher performance in his evaluation the test corpus seems few.

CHAPTER THREE

SYLLABLE STRUCTURE AND SYLLABIFICATION IN AMHARIC

This chapter presents the structure and the properties of Amharic syllables as of related studies carried on Amharic syllabification. The chapter begins by introducing the well known syllable structure of Amharic words. Presenting further discussion on various issues related with Amharic syllables, the chapter gives detailed account on Amharic syllable structure, syllabification, consonant clusters handling, gemination handling and on the issue of epenthetic vowel insertion of Amharic words. The chapter ends with presenting the benefit of Amharic syllables to work on stress pattern in the language.

3.1 Amharic Language Script

Amharic, the official language of Ethiopia, is a Semitic language. According to the 1998 census, Amharic has 17.4 million speaker as a mother tongue language and 5.1 million speakers as a second language (Ethnologue, 2004).

A set of 34 phones, seven vowels and 27 consonants, makes up the complete inventory of sounds for the Amharic language (Baye, 2010). Consonants are generally classified as stops, fricatives, nasals, liquids, and semi-vowels. Table 3.2 shows the phonetic representation of the consonants of Amharic as to their manner of articulation, voicing, and place of articulation.

Amharic has seven vowels. In addition to the five vowels common among many languages, Amharic has two central vowels, /e/ and /ix/, the latter with a mainly epenthetic function. The epenthetic vowel /ix/ plays a key role in syllabification. Moreover, epenthetic vowel is crucial for proper pronunciation in Amharic language. Table 3.1 shows the seven vowels and consonants along with their representation in Ge‘ez characters and their place of articulation respectively.

Table 3.1: Categories of Amharic Vowels

	Front	Central	Back
High	ii (ኢ)	ix(ኧ)	u(ኡ)
Mid	ie(ኤ)	e (ኧ)	o(ኦ)
Low		a(ኣ)	

Table 3.2: Categories of Amharic Consonants

		Labials		Alveolar		Palatals		Velars		Labio-Velar		Glottals	
Stops	Voiceless	p	ፕ	t	ት			k	ክ	kwa	ኳ	ax	ዕ
	Voiced	b	ብ	d	ድ			g	ግ	gwa	ጓ		
	Glottalized	px	ጵ	tx	ጥ			q	ቅ	qwa	ቋ		
Fricatives	Voiceless	f	ፍ	s	ስ	sx	ሽ					h	ሀ
	Voiced	v	ቭ	z	ዝ	zx	ዥ						
	Glottalized			xx	ጽ							hwa	ኋ
Africatives	Voiceless					c	ች						
	Voiced					j	ጅ						
	Glottalized					cx	ጭ						
Nasals	Voiced	m	ም	n	ን	nx	ኝ						
Liquids	Voiced			l	ል								
	Voiced			r	ር								
Glides		w	ው			y	ይ						

Amharic has its own typical phonological and morphological features that characterize it. The following are some of the striking features of Amharic phonology that gives the language its characteristic when one listen to the sound: the weak indeterminate stress; the presence of glottalic, palatal, and labialized consonants; the frequent gemination of consonants and central vowels; and the use of the epenthetic vowel (Tadesse, 2011). Among these, we found gemination of consonants and the use epenthetic vowel to be very critical for naturalness of pronunciation. Therefore, these features are the main issues to be considered while syllabifying Amharic words.

3.2 Syllable structure of Amharic words

Syllable structure, which is the combination of allowable segments and typical sound sequences, is language specific. As we have perceived from different literatures, the syllable type of a language can be defined in terms of underlying syllable templates. All the syllable type of Amharic can be also defined at the phonological representation. In Amharic language, there are six main syllable templates (Mulugeta, 2001). These templates accounts for all other possible syllable patterns. Moreover, the longest possible syllable is CVCC (Mulugeta, 2001). The main syllable templates of Amharic language are:

1. V
2. VC
3. VCC
4. CV
5. CVC
6. CVCC

We can classify these templates into different classes based on the grammatical structure of the syllable templates. In a simple weight distinction, there are heavy and light syllables in Amharic. Heavy syllable is a syllable that either ends in a consonant or has a long vowel or diphthong. Light syllable is a syllable that ends in a short vowel. A closed syllable is one that ends in a consonant and an open syllable is one that ends in a short vowel or diphthong. Thus we can restate the definition above: short voweled open syllables are light, all others are heavy. Table 3.3 shows the summary of different kinds of Amharic syllable structure.

Table 3.3: Different kinds of Amharic Syllable templates

Kind	Description	Example
Heavy	Has a branching rhyme. All syllables with a branching nucleus (long vowels) are considered heavy. Some languages treat syllables with a short vowel (nucleus followed by a consonant (coda) as heavy.	CVCC, CVC
Light	Has a non-branching rhyme (short vowel). Some languages treat syllables with a short vowel (nucleus) followed by a consonant (coda) as light.	CV,VC, VCC
Closed	Ends with a consonant coda.	CVC, VC, CVCC, VCC
Open	Has no final consonant	CV, V

In Amharic language we can found syllable structure containing the different kinds of syllable templates. For instance, syllable structure found in different parts of speech can be generalized, which means the same pattern of syllable structure can be observed for words in the same part of speech. For example, if we see the syllable structure in nouns there are well known observed sequence of syllables pattern, such as CV-CV/CV-CV-CV like in ውሃ /**wix-ha**/ meaning –water".

We can also observe the syllable structure in nouns in a simple weight distinction. For instance, Light heavy Light (LHL) syllable CV-CVC-CVC like in /**mix-lixk-kixt/** meaning –sign". In this example, the first half of the geminated consonant /k/ as the segment of the preceding syllable while the second half of the geminate represented as the onset of the second syllable. Heavy syllable (H) syllable (CVCC), like in /**berr/** –door" and /**lixbb/** –heart". Heavy-Light *(*CVC-CVC) like in /**fixy-yel/** meaning –goat". Heavy super heavy (HSH) syllable (CVC-CVCC), like in /**men-gixst/** meaning –government" and /**tixm-hixrt/** meaning –education". Light super heavy (LSH) syllable (CV-CVCC), like in /**me-lixkt/** –message". Such kinds of classification of syllable templates by weight distinction help us in assigning stress.

As it is pointed out in (Mulugeta, 2001), the syllable types in nouns stems are CV, CVC, and CVCC. The superheavy syllable CVCC is more often found in the underlying form of noun in word final position while the superheavy syllable CVVC type is found mainly in the verb roots. Stem finally, the permitted sequence are CVC, CV and CVCC. The CVCC string is the maximally allowed syllable in Amharic syllable structure.

We can also observe the syllable structure of Amharic nouns when the words are found in plural forms. Amharic plural nouns can be formed by using two options. The first is by suffixation, most often suffix –occ (-ኦች) is used. The second option is by internal changes to the stem which referred to as formation of broken plurals'. The other option is by reduplication all parts of the stem noun. The sound plural, which also called external plural ending in Amharic is –occ. It is a regular plural marker in the language and always suffixed to a noun. If a noun ends in a consonant, the plural is formed by suffixing –occ (-ኦች) . But, if the noun ends in a vowel, the plural is basically formed by suffixing –wocc (-ዎች) or by deletion of the stem-final vowel.

Examples:	**Singular**	**Plural**	**Meaning**
	ቤት (bet)	be-tocc	"house"
	ሰው(sew)	se-wocc	"man"
	ክር (kixrr)	kixr-rocc	"thread"
	በሬ (be-re)	be-re-wocc	"ox"
	ውሻ (wisx-sxa)	wisx-sxa-occ	"dog"
	ተማሪ(te-ma-rii)	te-ma-rii-occ	"student"

As it is presented in the example the CVCV syllable structure becomes CVCVCVCC type and the inserted glide consonant acts as the onset of the next syllable, which is the CVCC type that is formed as the result of pluralization (the /-/ (hyphen) represent the syllable boundary). On the

other hand, the CVCC type singular noun is reduced to the CVC type as a result of the plural suffix.

Examples:

Singular	**Meaning**
ይስበር (yix-se-ber)	"let him break"
ሰበረ (seb-be-re)	"he broke"
ተሰባበረ (te-seb-ba-be-re)	~~–it~~ was broken"
ሰብሮ (seb-ro)	"he having broken"
ስበር (six-ber)	"break!"
መስበር (mes-ber)	"to break"

Amharic verbs also have their own syllable structure. Gemination is one of the issues in Amharic verbs and it is common phenomenon in Amharic. For instance, type A verbs geminate the second radical in certain pattern while type B geminate the second radical in all patterns (Baye, 2010). This feature has its own impact on the syllable structure of the words. For instance, in some Amharic words like **/fixy-yel/**, the geminated consonant may have position in two different syllable. Accordingly the syllabification pattern may change because of the gemination of consonant.

3.2.1 Geminates and cluster of consonants

One of the important features of Amharic phonology that should be handled in automatic syllabification is gemination of consonants. Traditionally, it is represented either as /C:/ or /CC/ to indicate its length. In phonetics, gemination happens when a spoken consonant is pronounced for an audibly longer period of time than a short consonant. Although double consonants (sequence of the same consonants or consonant clusters) and geminates are phonetically (relating to speech sounds) the same, they are phonologically different. Gemination is distinct from stress and may appear independently of it, it is a doubling of consonants.

a) Geminates and their syllable structure

Gemination in Amharic is one of the most distinctive characteristics of the cadence of the speech, and also carries a very heavy semantic and syntactic functional weight. Unlike English language in which the rhythm of the speech is mainly characterized by stress (loudness), rhythm in Amharic is mainly marked by longer and shorter syllables depending on gemination of consonants, and by certain features of phrasing. In Amharic, all consonants except (ሀ) /h/ and (ዕ) /ax/ may occur in either a geminated or a non-geminated form. Amharic gemination is either

lexical or morphological. As a lexical feature it usually cannot be predicted. The failure of the orthography of Amharic to show geminates is the main challenge in Grapheme-To-Phoneme (GTP) conversion (Tadesse, 2011).

Gemination occurs in many of Amharic words for example; we can observe the difference between /**kixft/** and /**kixffixtt/** because of the geminate consonant /**f/** (ፍ) and /**t/ (**ት**).** Figure 3.2.1 and Figure 3.2.2 shows the difference of the two words in waveform respectively. When /**f/** and /**t/** are geminated there is epenthesis vowel /**ix/** inserted between them therefore syllabification for the two words becomes completely different. Without gemination the word contains only single syllable type CVCC. Therefore, the whole phoneme sequence, /kixft/, is taken as a single syllable. But when it is geminated, the word will have two CVC syllables, /kixf-fixtt/. The epenthetic vowel is clearly seen in the waveform (Figure 3.2), similarly in Figure 3.1 we can observe the missing epenthetic vowel between the two consonants /f/ and /t/.

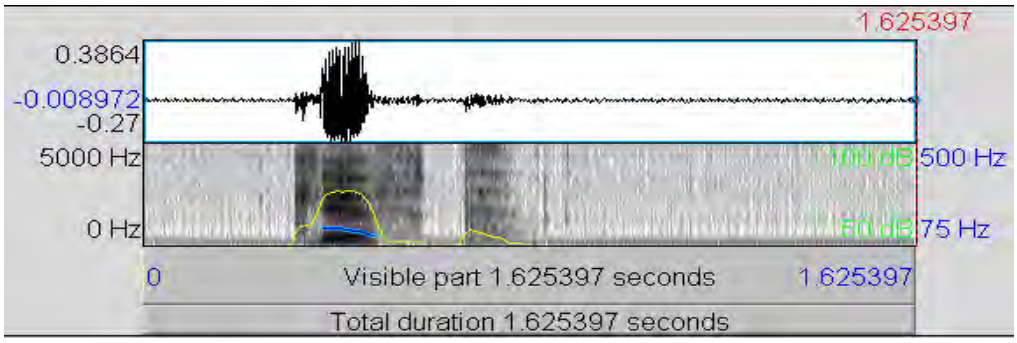

Figure 3.1: Waveform for the word (ክፍት) without gemination effect /kixft/

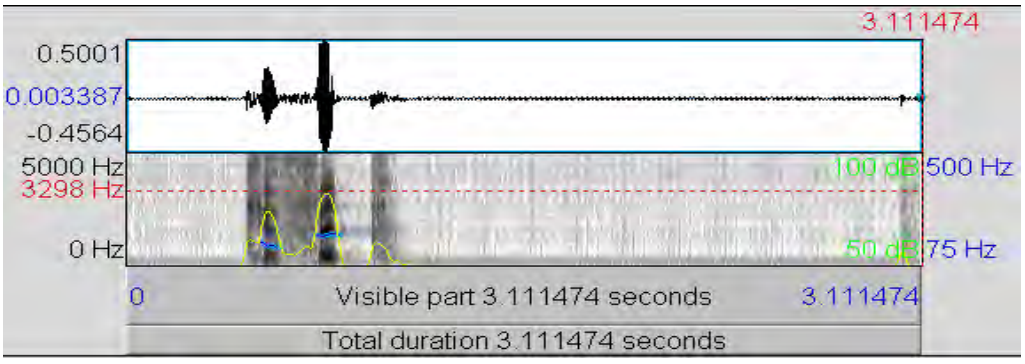

Figure 3.2: Waveform for the word (ክፍት) with gemination effect /kixffixtt/

b) Cluster of consonants and their syllable structure

A cluster of consonants is nothing but a succession of two consonants without a splitting by any vowel. In Amharic, the maximum number of allowable consonant sequences in a cluster is two (Mulugeta, 2001).

Hudson (2000) proposes three possibilities for consonant sequences in Amharic. According to his analysis Amharic word structure allows the following consonant sequence.

a. No word-initial consonant sequences except C+w as in hwala ~~–b~~ack" and qwanqwa ~~–l~~anguage".
b. Word-final sequence of at most two consonants of which, typically, the sonority of the first equal or greater than the second, including final long consonant (thought of as CC) , for example /**bixltx/** _clever', /**wixdd/** ~~–e~~xpensive" , /**hixgg/** ~~–l~~aw".
c. Medial sequence of at most two consonants, including cases with long consonants counted (geminated consonants) as two consonants: /**alle/** ~~–h~~e is present", /**metta/** ~~–h~~e came" /**wesdo/** ~~–h~~e may take", /**beqlo/** ~~–m~~ule".

Thus the maximum number of consonants in Amharic cluster is two (Mulugeta, 2001). Consonant clusters are not permitted at the beginning of a word. However, some speakers pronounce an initial cluster when the second element is a liquid.

Examples: /**kixremt/** → **/kremt/** ~~–r~~ainy season"

/**fixrash/** → **/frash/** ~~–m~~attress"

/**bixlatta/** → **/blatta/** ~~–a~~n honoric title"

Onset cluster are not permitted in Amharic. The epenthesis vowel should be inserted between the liquid and the preceding consonants. Therefore, whenever we found consonant clusters at initial position we have to insert epenthetic vowel except for the phoneme /w/; if it occurs next to the initial phoneme. But, final clusters of consonants are permitted in Amharic. However, if the hierarchy of sonority sequence is not satisfied, the final cluster takes an epenthesis vowel and the cluster splits. Examples: /**brd/** → /**bixrd/** ~~–e~~oldness"

/**trf/** → /**tixrf/** ~~–p~~rofit"

/**atr/** → /**atixr/** ~~–f~~ence"

/**tfr/** → /**tixfixr/** –nail"

In those Ethio-semetic languages, which allow CVCC type syllable, the sonority of the first consonant must be higher than that of the second (Rose, 1997). This might be universal principle too.

According to the sonority hierarchy principle, the sonority of a syllable increases from the beginning of the syllable onwards, and decreases from the beginning of peaks onwards. The sonority of speech sound can be ranked in terms of their relative sonority from lowest to highest as Oral stops, fricatives, nasals, liquids, semi-vowels and vowels. Hence, oral stops have minimal sonority while vowels have the highest degree of sonority (Fujisaki, 1995). The sonority hierarchy helps us to determine what sequence of sounds can occur in syllable. As the result, when stops and liquids appear together in cluster word finally, an epenthesis vowel is inserted between them; because the sonority of the final liquid is greater than that of the preceding phoneme.

3.3 The issue of epenthesis in Amharic words

The process of epenthesis is common in Amharic. It can occur word-initially or medially. As (Hudson, 2000) stated epenthesis is extensive in word-formation in the Ethiopian Semitic languages, since many morphemes, both roots and affixes, consists only consonants. Amharic epenthesis vowel may be said to provide almost all occurrences of the high central vowel /**ix/** (**ɨ**).

There are two general rules concerning automatic insertion of an epenthetic vowel in Amharic.

1. Word-initially no consonant clusters are allowed.
2. Elsewhere clusters of no more than two consonants are tolerated.

Hudson (2000) also proposes three environments for epenthesis corresponding to the possibilities of consonants sequence word initial, medial and final position in Amharic.

1. #CC as in words like /**tsebr/** →/**tixsber/**, /**sber/** →/**sixber/** (word initial consonant clusters are impermissible) (the (#) indicates the position of the cluster, word initial or word final).

2. CC# as in word like/**mkr/→ /mixkixr/**. In this case, the sonority of the final consonant, /r/, is greater than that of the preceding consonant, /k/. Thus, to split up the final cluster epenthetic vowel /ix/ is inserted. On the other hand, if the sonority of the first is equal or greater than that of the second consonant, epenthesis will not be applied.
3. CCC, in the case of this environment Hudson (2000) proposes three types of CCC violation where epenthesis /**ix/** is required.
    a. CCC→ CCixC, in a word like /**fendto/→ /fendixto/** –exploit"
    b. C:C→ C:ixC, in a word like /**fellgo/→/fellixgo/** –want"
    c. CC:→CixC:, in a word like /**sebrre/→/sebixrre/** –break"

Mulugeta (2001) further claims, if the first and the second consonant pair are different long consonants; the epenthetic vowel is inserted between them as shown below:

C:C:→ C:ixC:, in words like like /**sbbrr/ → /sixbbixrr/** –break", /**kfftt/→/kixffixtt/** –open".

Mulugeta (2001) also discussed further about the process of epenthetic vowel in Amharic by dividing into six sections, we present all of them with examples as follows.

1. Word initially no consonant cluster—the sonority of the initial consonant is greater than that of the following consonants for word initial cluster, the epenthesis is inserted before the cluster.
2. If a word medial cluster of consonants contains the geminate and singleton in sequence, the epenthesis vowel is inserted after the geminate consonants. Examples: **/fel:go/→/fel:ixgo/**,**/mel:so/→ /mel:ixso/, /lem:no/→/lem:ixno/** etc.
3. If word medial cluster of consonants contains a singleton and geminate in sequence, the epenthesis is inserted before the geminate consonant. Examples**: /sebr:e/→/sebixr:e/, /gedy:ie/→/gedixy:ie/, /txrg:ie/→ /txrixg:ie/.**
4. If word medial or final cluster of consonants contain two geminate consonants in sequence, the epenthesis inserted between the two different geminates. Examples: **/sbbrr/→/sixbbixrr/, /kfftt/→/kixffixtt/.**
5. If three consonants are appeared in sequence word medially, the epenthesis vowel is inserted before the third consonant. Examples: **/sentxqo/→sentixqo, /fendto/→ /fendixto/, /bergdo/→ /bergixdo/.**

6. If the sonority of the final consonant is greater than the preceding consonant, the epenthesis can be inserted between the final clusters. Example: /**dngl/→/dixngixl/** –~~v~~irgin", /**tsebr/→ tixsbixr** –~~m~~ay you break", /**mkr/→ /mixkixr/** –~~a~~dvice".

3.4 Syllabification

Having gemination handling rules, syllabification rules, epenthesis (epenthetic vowels insertion) rules and syllable templates of the language it is possible to syllabify (mark syllable boundaries) given the Amharic text. In the previous sections we have seen all the rules of the language in relation with epenthetic vowel insertion and gemination handling. Moreover, we have the syllable templates of the language from different linguistic literatures in accordance with empirical experiment. At this level we can develop rule-based automatic syllabification algorithm applying all the rules.

Furthermore, we can have general syllabification model at this level. Figure 3.3 shows the general syllabification model for Amharic language.

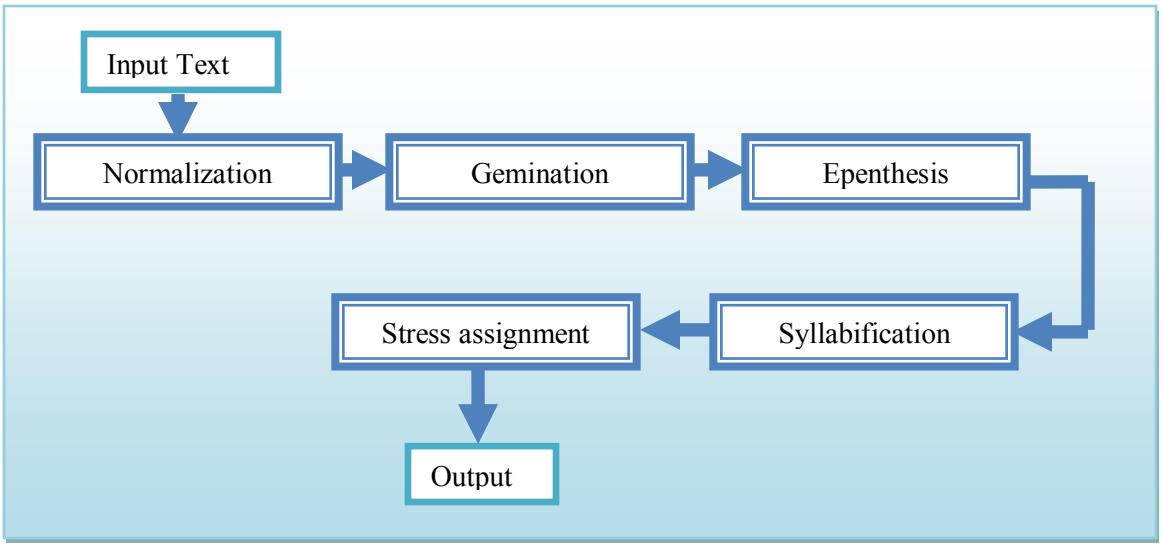

Figure 3.3: General automatic syllabification model for Amharic text

As it is shown in the model (Figure 3.3), as an input to activate the module, Amharic text (written in Ge'ez alphabets or transcribed text) is used. Then, in the normalization module normalization is carried out. Normalization means checking for the input text and transliterate if the input text is written in Ge'ez alphabets or checking for the transliterated input text to confirm

the input is correct Amharic text. The normalized text is again passed to the gemination module as an input and gemination is takes place applying the gemination rule of Amharic language. At this point, epenthesis can be carried out; the gemination module final result is used as an input for this module. After applying insertion of epenthetic vowel based on the rule of epenthesis in the language, syllabification is done by the syllabifier module. The syllabifier applies syllabification using an algorithm to syllabify texts in their legal sequence. Finally, by examining the syllable sequences in accordance with their syllable weight and the stress assignment rules, stress assignment is takes place in the final module. The final output will be stress and syllable boundary marked transcribed Amharic text.

3.5 Stress and Syllables

There has never been agreement among linguistics on the topic of stress assignment in Amharic. Stress in Amharic words is complex (Alemayehu, 1987). However, there are some systems proposed in relation with stress and syllable structure. In many stress languages, stress is sensitive to a distinction called syllable weight. In a simple weight distinction, there are heavy and light syllables, defined as follows:

Heavy syllable: syllable that either ends in a consonant or has a long vowel or diphthongs.

Light syllable: syllable that ends in a short vowel.

Regarding the stress assignment rules of Amharic, we get the following rules form different literature. There are also other methods proposed by different scholars but the following rules have direct relation with syllables and syllable weight.

a. Stress falls on a heavy final syllable only in bisyllabic words when the first syllable is light.
b. Otherwise, the final syllable is skipped and the right most heavy syllable is stressed.
c. In the absence of any heavy syllables, the left most of a string syllables is stressed.

Although stress assignment is beyond the scope of this thesis work, once we have the syllables we can use the benefit of syllabification algorithm in order to have syllables and using rules of syllable weight assignment to assign stress based on the rules defined in relation with syllables

and their corresponding weight. Therefore, having syllables and syllable weight for each syllable in the given word we can assign stress based on the rules specified.

CHAPTER FOUR

DESIGN OF AUTOMATIC SYLLABIFICATION ALGORITHM FOR AMHARIC

A detail description of design issues and techniques used for the Amharic automatic syllabification algorithm is dealt in this chapter. The general architecture of the model is present. Moreover, a rule-based epenthesis insertion algorithm design, automatic syllabification algorithm design, the techniques and approaches used are described in detail.

4.1 Approaches and Techniques

The process of automatic syllabification given Amharic word is the process of segmenting a sequence of phonemes into syllables. A syllable is a unit of sound composed of a central peak of sonority (usually a vowel), and the consonants that cluster around this central peak. The process of automatic syllabification takes a transliterated word as an input. Then, it inserts epenthetic vowel if there is impermissible consonant clusters. Again the output of epenthesis module is parsed once more and the syllable boundary marker (-) is inserted whenever it finds a legal syllable boundary in the input word. Finally, it produces a syllabified word as an output.

To handle automatic syllabification, there are two broad different techniques: rule-based (or already-syllabified knowledge-based) and data-driven (corpus-based) approach which attempts to syllabify new words from evidence based on words.

As it is mentioned earlier, data-driven approaches need large amount of already syllabified words to attain a better accuracy syllabification. Moreover, rule-based approaches show a better performance in most languages for automatic syllabification. Therefore, the design and implementation of this thesis work is based on the rule-based approaches in order to have improved syllabification algorithm. The rule based algorithm is designed having all the rules of the language. After wards, the given word is parsed and if it has consonant clusters epenthesis is performed to split the impermissible consonant clusters. Once more, the word is parsed and the syllable boundary marker is inserted applying the syllabification rules and by matching the legal syllable templates of the language. Finally, the output will be syllable boundary marked word or text.

4.2 Design Goals

The general goal in developing the automatic syllabification algorithm for Amharic is to attain better accuracy syllabification. Accuracy is nothing but the closeness of the agreement between the test result and the accepted reference value (manually syllabified Amharic words (by linguist expert) of the test set). The main goal of the design of automatic syllabification is, therefore, to achieve a better performance in syllabifying Amharic words in terms of accuracy.

4.3 Designing the syllabification Algorithm

4.3.1 Syllabification Architecture

Figure 4.1 shows the architecture of the automatic syllabification algorithm for Amharic. Syllabification of foreign words, abbreviations and acronyms were not considered at this point of our system development, since these words do not follow the standard syllabic structure of Amharic language. In the next paragraphs we will explain each components of the architecture.

The input for the syllabification algorithm is a phonetic representation rendered in ASCII characters from Amharic lexicon without syllable boundary information (direct transliteration is done manually). Then, gemination is performed using expert‘s knowledge which is done manually. After gemination information is incorporated in the text, the parser module is activated and it parses for vowels and consonants. After the valid consonant clusters and geminated consonants groups are identified, epenthetic vowel is inserted to split illegal consonant clusters.

The next step is parsing again to identify vowels and consonants, which help to identify syllable templates, and match the syllable templates of the language considering the syllabification rules. Finally, syllable boundaries are introduced according to the specifications of the algorithm. A word may be syllabified in different syllable structure but the algorithm selects the legal syllable structure sequence for the input word. Here, the well known linguistic syllabification implementation principles namely; *maximum onset principle* and *sonority hierarchy principle* are also implemented. Expected output is an identical phonetic representation as input but including epenthetic vowel and syllable boundary markers.

The final output of the syllabification becomes input for the stress marker. The stress marker identifies the syllable weight of each syllable in the word. Then, applying the stress marking

rules it assigns stress. However stress assignment in Amharic is beyond the scope of this thesis work we included stress assignment module in our architecture.

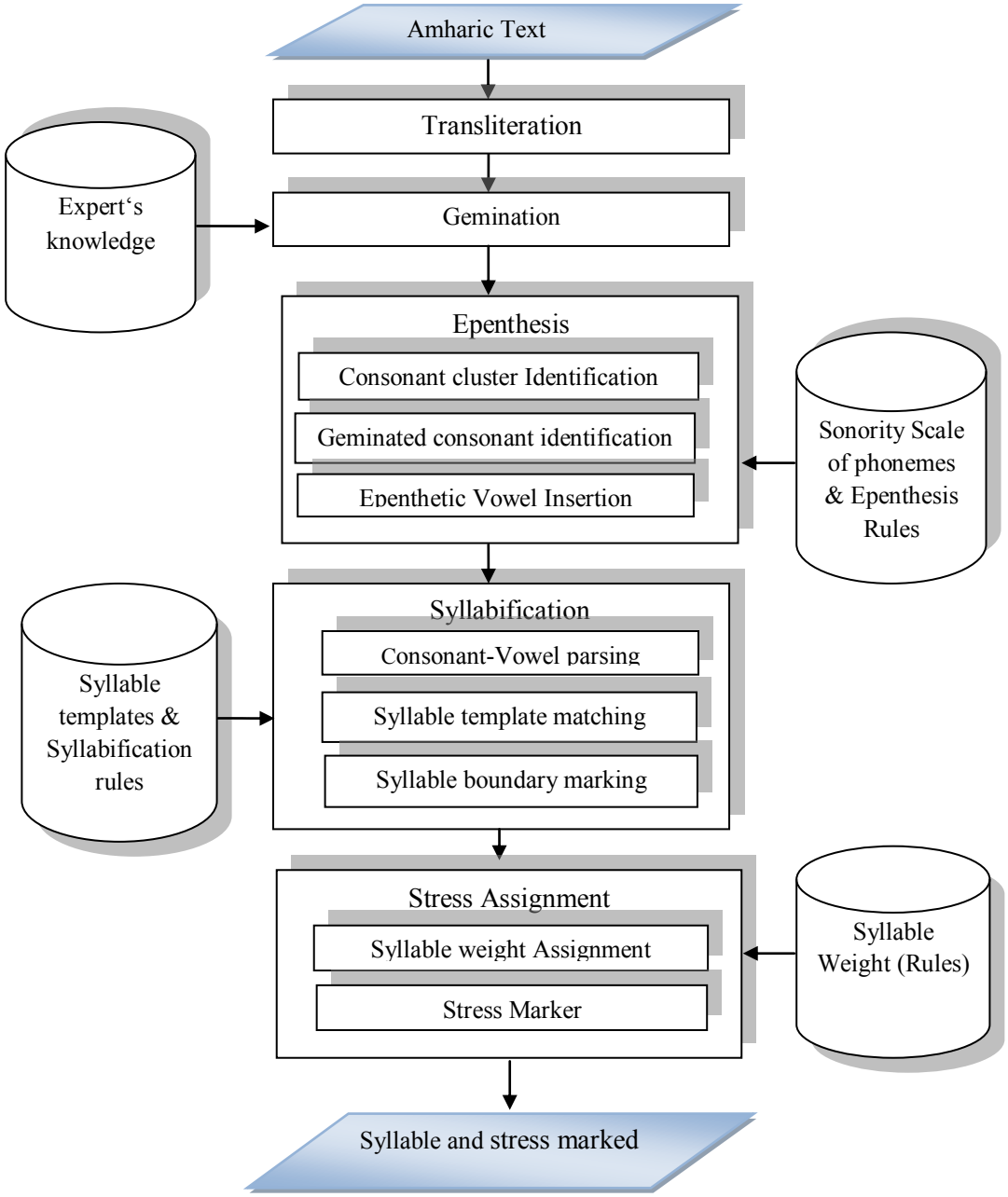

Figure 4.1: Automatic syllabification algorithm architecture

Moreover, we would like to mention the benefit of syllable structures to assign stress in Amharic words. Since stress is sensitive to a distinction called syllable weight (heavy and light) once we

have syllable weight assigned syllables, we can perform stress assignment by directly applying the rule of the language, like rules mentioned in Section 3.4.

4.3.2 Rules and Algorithms

It has been identified that there are six legal syllable structures (templates) in Amharic, namely V, VC, VCC, CVC and CVCC for words which belong to the Amharic language (Mulugeta, 2001). Though a number of examples for syllabified words belonging to each of the above structures are presented in the literature (Mulugeta, 2001) (Aster, 1981), the methodology or grammatical rules describing how to syllabify a given word has not been presented. A word can be syllabified in many ways retaining the permitted structures, but only a single correct combination of structures is accepted in a properly syllabified word. For example, a word having the consonant-vowel structure VCVCVC can be syllabified in the following different ways, retaining the valid syllable structures described in the literature: V-CVC-VC, VC-VC-VC, VC-V-CVC. However, only one of these forms represents the properly syllabified word.

The determination of a proper mechanism leading to the identification of the correct combination and sequence of syllable structures in syllabifying a given word became the major challenge in this research.

Further review of the literature and empirical observation led to the following model with regard to Amharic syllabification; a fundamental assumption that the accurately syllabified form of a word can be uniquely obtained by formulating a set of rules (set of rules and procedures mentioned in the next paragraphs), rules can be empirically shown to be effective.

Modeling identification of the epenthetic vowel improves speech synthesis process (Sebsbie *et al*., 2004). We also understood from our empirical observation epenthetic vowels has great role in syllabification of Amharic words, and it is a common phenomenon in the language. It can occur at word initial, word medial and word final position. Moreover, while implementing grapheme-to-phoneme conversion the written form and the spoken form is one to one except the epenthetic vowel. The rules of epenthesis will decide the presence or absence of such vowel in the spoken form of the language. Therefore, before we design the model of syllabification algorithm we first model the insertion of the epenthetic vowel (automatic epenthesis) in separate

module (the module is shown in the syllabification architecture, Figure 4.1 and the rules applied in this module are summarized in Table 4.1).

Table 4.1: Summary of the epenthesis vowel insertion procedure

Rule #	Position	Observed Sequence	Epenthesis	Exception
1	final	#CC	#CixC	If the first phoneme is consonant and the next consonant is glide /w/
2	medial or initial	CCC	CCixC	If sonority of the middle consonant is greater than The rest (CixCC)
3	medial or initial	C_1C_1C (CC:)	C_1C_1ixC (C:ixC)	
4	medial or initial	CC_1C_1(CC:)	CixC_1C_1 (CixC:)	
5	medial or initial	$C_1C_1C_2C_2$ (C:C:)	C_1C_1ixC_2C_2 (C:ixC:)	
6	final	CC#	CixC#	If the sonority of the last phoneme is less or equal to the preceding

Both the syllabification and epenthetic vowel insertion algorithm reads input from left-to-right, since any syllable requires a vowel and since onset is filled up before coda.

Epenthetic Vowel insertion procedure:

1. *Accept input word and scan from left to right.*
2. *If consonant cluster occurs at word initial position, insert epenthetic vowel between them.*
   ***Exception**: If the first phoneme is consonant and the next consonant is glide /w/.*
   (Rule #1)
3. *If three consonants are appeared in sequence word medially or word final position, insert epenthetic vowel before the third consonant.(Rule #2)*
   ***Exception**: If the middle consonant sonority is greater than the rest insert epenthetic vowel after the first consonant in the cluster.*
4. *If a cluster of consonants contains the geminate and singleton in sequence, insert epenthetic vowel after the geminated consonants.(Rule #3)*
5. *If a cluster of consonants contains the singleton and geminate in sequence, insert epenthetic vowel after the singleton consonants. (Rule #4)*

6. *If a cluster of consonants contains two different geminates in sequence, insert epenthetic vowel between the two geminate consonants. (Rule #5)*
7. *If the sonority of the final consonant is greater than that of the preceding consonant, the epenthetic vowel is inserted between the final consonant clusters. (Rule #6)*
8. *Repeat 2 up to 7 until all the phonemes are parsed in the phonemes list.*

The accuracy of this model was first tested by recording words and look into the acoustic evidence (waveform and spectrogram). Convinced that, the results were consistent with the descriptive treatment of the subject in the literature, it was concluded the above set of rules could describe an accurate epenthesis algorithm for words belonging to Amharic words. Figure 4.2, 4.3 and 4.4 show examples of words which exhibit epenthetic vowel as of the rules.

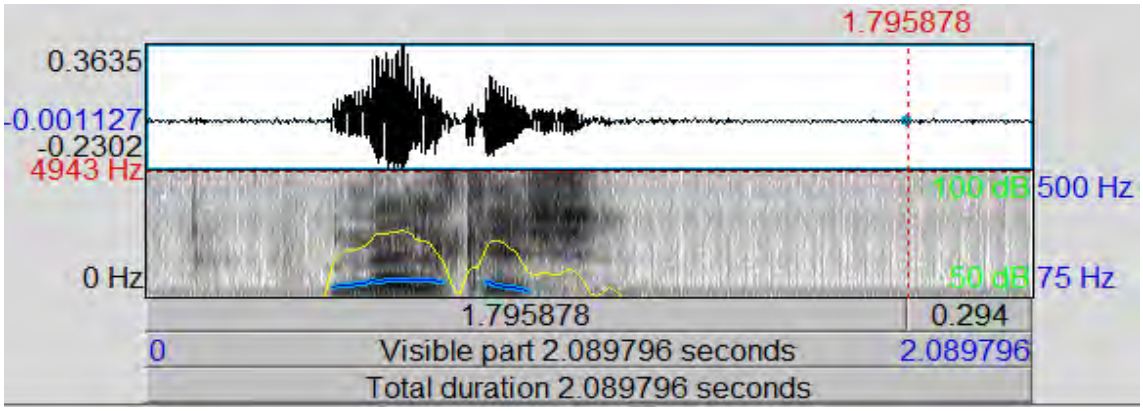

Figure 4.2: Waveform for the word /melkixsx/

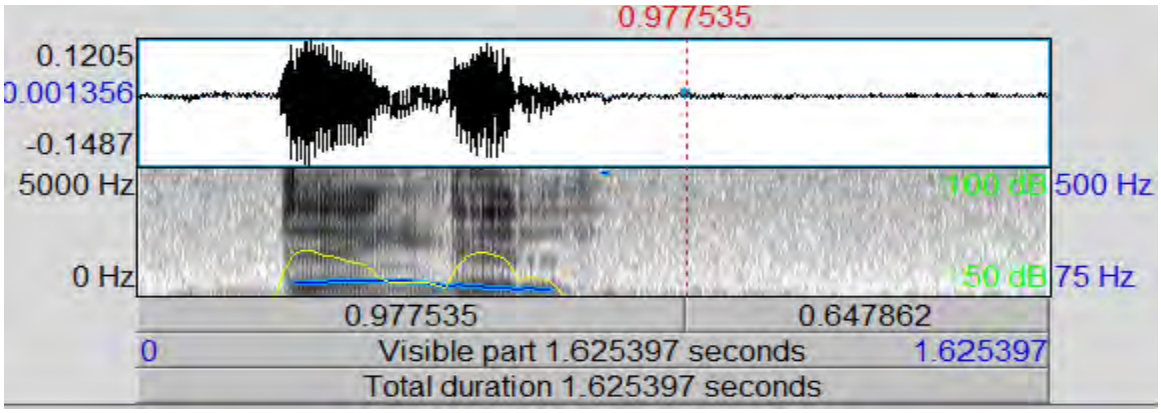

Figure 4.3: Waveform for the word /dixngixl/

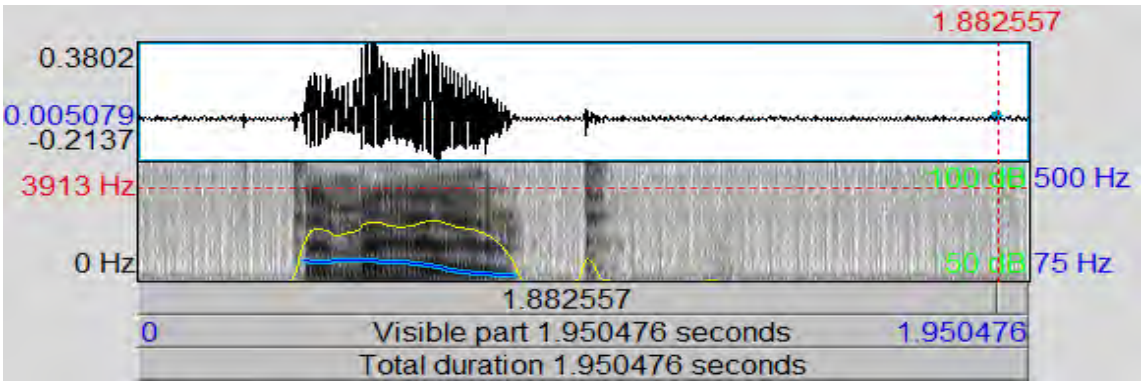

Figure 4.4: Waveform for the word /tixmhixrt/

The final output of the epenthesis module directly becomes the input for the syllabification model. As it is mentioned earlier, the syllabification algorithm reads the given phonemes sequence from left-to-right and the rules are applied repeating the template matching operation for each phoneme sequence in the given word.

Proposed syllabification procedure in Amharic:

1. *Accept the input from epenthesis algorithm and scan from left to right.*
2. *At word initial position if two vowels phonemes (VV) occurs in sequence, mark syllable boundary between them.*
3. *If the initial phoneme is vowel and the next two phonemes are consonant and vowels respectively; mark the syllable boundary just at the second*
4. *If (VCCV) pattern occurs at any position, mark syllable boundary between the two consonant clusters.*
5. *If (VCVC) pattern occurs at word initial position, mark syllable boundary before the second vowel.*
6. *If (CVV) type sequence occurs at any position, mark syllable boundary between the two vowels.*
7. *If (CVCCV) phoneme sequence occurs at word initial position mark syllable boundary between the middle consonant clusters (CVC- CV).*
8. *If (CVCC) pattern occurs at word final position and if there is phoneme before the first consonant mark syllable boundary before the initial consonant in this pattern.*

9. *If (CVCV) pattern occurs at any position, mark syllable boundary after the vowels, but if it occurs at word final position the syllable boundary becomes CV - CV pattern.*
10. *If (CVC_1C_1VC or CVCCVC) pattern occurs in a word mark syllable boundary between the geminated consonants. (CVC_1- C_1VC).*
11. *If (VVCC) syllable pattern occurs at word final or initial position mark syllable boundary between the two vowels.*
12. *Repeat 2 up to11 until all phonemes are parsed*.

Having marked the first syllable boundary, continue the same procedure for the rest of the phonemes as in. Then, the algorithm repeat the step for all syllable patterns, except for patterns found at initial position, until the whole word is syllabified. Syllable pattern V, VV, VC, VC_1C_1, CVCC occurs at word initial position. CV, CVC syllable pattern occurs in word initial, word medial and word final position. CVVCC, CVC_1C_1 syllable pattern occurs at word initial and final position. Moreover, CV syllable template occurs in a larger portion of the syllable distribution of the language. The syllabification algorithm makes into consideration the positions of syllable at which they occurs and it uses the information to decide the legal syllable pattern, and to avoid the confusion of syllabification of words in such patterns. Furthermore, the algorithm makes use of the universal sonority hierarchy principle in deciding the proper position of syllable boundary. Finally, digraph replacement is performed, to make the given phoneme sequence readable according to the transliteration scheme used in this thesis work.

The procedures identified, epenthesis procedure and syllabification procedure, are sensitive to the sequence since they interact with each other. The Amharic epenthesis and syllabification procedure (algorithm) identified above are presented in the form of a formal algorithm implemented in C# Programming language. The algorithm accepts an array of phonemes read from a list of words (a notepad text in ASCII encoding, one word in single line) in a text, and then the algorithm begins to process, first the epenthesis, and then the syllabification. The final output of the system is also a text file within ASCII encoding and the default file name is *Syllabified_amharic_words.txt*, which is given by the system.

In the character list the index starts from zero for initial phoneme and increment to the next phoneme in the list, when we read the character list from left to right. The function *InsertEpentheticVowel(inputStr)* parses from left to right applying the rules and it inserts the

epenthetic vowel /ix/. The function *Syllabify(inputStr*) will mark the syllable boundaries of an accepted array of phonemes from the epenthesis function .

We have also used other supporting functions to support the main functions namely, the epenthesis (*InsertEpentheticVowel(inputStr)*) and the syllabification function *syllabify(inputStr)*. The functions are described as follows:

- *isVowel(char Phoneme):* it is a Boolean function which accepts character phonemes and returns true if the phoneme is a vowel.
- *isConsonant(char Phoneme)*: accepts a phoneme and returns true if the given phoneme is a consonant.
- *SonorityOfConsonant(char Phoneme)*: accepts a phoneme and returns the sonority scale of the input phoneme. We assigned sonority scale for each Amharic consonant as it is shown in table 4.1. Sonority scale is almost universal for all languages or it is language independent (Jany *et al*, 2007) (Fujisaki, 1995). Therefore, as in state-of-the-art systems, we propose grouping of phonemes into classes and we assigned sonority scale dealing with each classes. Stops have got the least sonority scale (number) and glides are more sonorous as it is shown in the table 4.1.
- *Replace_digraphs(string inputStr)*: it replaces phonemes, which were represented in single characters in the algorithm for the simplicity reason. In the final output, those phonemes should be replaced for purpose of readability and to be consistent with the transliteration scheme used in this thesis.
- *RemoveDuplicateBoundaryMarker(string str)*: removes syllable boundary marker if duplicate syllable boundary marker exists in the final output of the syllabifier.

A complete listing of the algorithm is provided in Appendix B.

Table 4.2: Sonority scale of Amharic consonants

Stops			Affricatives			Fricatives			Nasals	Liquids	Glides
Vs	Vd	G	Vs	Vd	G	Vs	Vd	G			
1	2	3	4	5	6	7	8	9	10	11	12

Symbols: Vs= Voiceless, Vd= Voiced, G= Glottalized

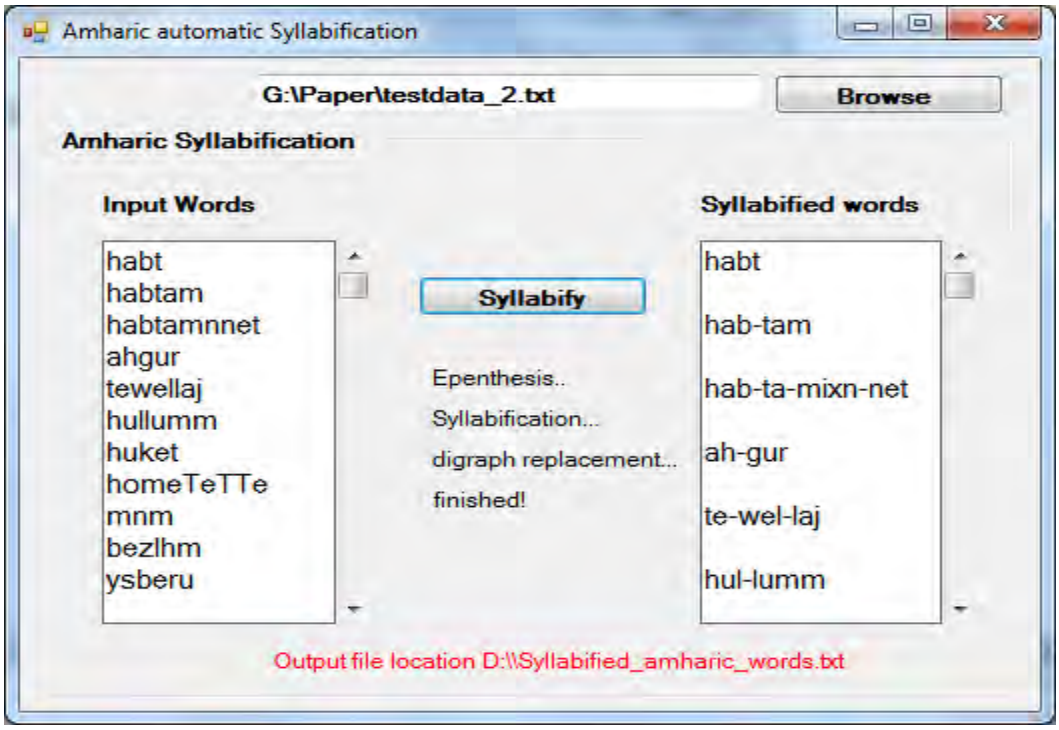

Figure 4.5: GUI for Amharic automatic syllabification

CHAPTER FIVE

EXPERIMENTAL RESULTS AND EVALUATION

To evaluate the performance of the algorithm, we have carried out testing using 1000 collected Amharic words as it is stated in data collection methodology (see Section 1.4.1). The output of the syllabification algorithm is given to expert for evaluation. In this chapter, the detail description of corpus preparation, epenthesis handling and syllabification performance of the algorithm based on expert evaluation is presented. Moreover, the statistics of syllable pattern occurrence and epenthetic vowel insertion in the test set are presented.

5.1 Test corpus description

The test corpus prepared for the testing and evaluation contains a total 1000 Amharic words. Each word contains three to four syllables in average. This test corpus contains a total of 3,025 syllables. Few words from our test corpus are shown in Appendix C. When the corpus is prepared we have tried to include all Amharic phonemes. The corpus also contains words with gemination and consonant clusters. Words that contain geminated consonants at medial and final position and with more than two geminated consonants are included. In the same way, for consonant clusters we have included words with initial consonant cluster, medial consonant cluster, final position consonant cluster and words with more than two consonant clusters.

Each word in the test corpus is recorded by the researcher for empirical experiment. The test corpus is transcribed directly without considering epenthesis or gemination. Then, gemination is considered and experts are consulted for some of words. For majority of words, the gemination information is taken from Amharic dictionary (Amsalu, 2004) and from different literatures on Amharic language. Generally, we have tried to include all features of Amharic language in the test corps that needs to be considered in syllabifying Amharic words. Table 5.1 shows the distribution of words with consonant clusters and geminated consonants.

Table 5.1: Distribution of consonant clusters and geminated consonants

Cluster type	Type	Phonemes Position in a word	Total Number	Percentage (%)
consonant	#CC	initial	256	45.47
consonant	CC#	final	78	13.85
consonant	CCC	medial or final	55	9.77
gemination	C:C	medial or final	65	11.54
gemination	CC:	medial or final	61	10.83
gemination	C:C:	medial or final	48	8.53
Total			563	100

5.2 Epenthesis performance

The total corpus contains 563 consonant clusters (including geminated consonants), that needs splitting of the impermissible consonant cluster using the high central vowel /ix/.Our epenthesis algorithm is tested by observing the waveform of the recorded speech for some of the words. In addition, the expert checked while evaluating the syllabification (Appendix C shows the complete corpus given to the expert). The expert compares the input phoneme sequence and the output of the algorithm, and gives his remark on each grapheme-to-phoneme conversion; if there is epenthesis error he provided the correct one. As a result, the expert agreed on 96.62% of the total 563 epenthetic vowel insertions.

Table 5.2, shows the total number of correct epenthetic vowel insertions, wrong epenthetic vowel insertions, correctly neglected (not inserted epenthetic vowel between consonant clusters), and wrongly neglected insertions (missed epenthetic vowel insertions). Here, accuracy is defined in percentage as:

$$\text{Accuracy} = \frac{\#\text{ Correct insertions} + \text{correctly neglected insertions}}{\text{Total clusters in each position}} \quad X \quad 100\%$$

There are 6 missed epenthetic vowel insertions between two consonant clusters (CC sequence) at word medial position, which are pointed out by the expert. Therefore, there are a total of 19 errors which are also considered as juncture error too. Moreover, most of the errors occurred at word final position.

Table 5.2: Statistics about consonant clusters and geminated consonants in epenthesis

Cluster type	Phonemes Position	Type	Total	Total # of correct insertions	Total # of wrong insertions	Total # of correctly neglected insertions	Total # of wrongly neglected insertions	Accuracy
consonant	initial	#CC	256	255	0	0	1	99.61
consonant	final	CC#	78	45	2	22	9	85.90
consonant	medial or final	CCC	55	54	0	0	1	98.20
gemination	medial or final	C:C	65	65	0	0	0	100
gemination	medial or final	CC:	61	61	0	0	0	100
gemination	medial or final	C:C:	48	48	0	0	0	100
Total			563	528	2	22	11	97.69%

5.3 Syllabification performance

The test was carried out in order to assess the performance of the proposed algorithm for automatic syllabification of Amharic words. Errors are defined as incorrect insertion of syllable boundary markers, which means if the syllabification according to the language expert is not similar with the result of the word syllabified by the algorithm. The other error type is juncture error. For instance, if a word should have five junctures and if the output by the system puts more or less junctures from the expected junctures, this leads to syllabification error. This type of error occurred due to the missed or the wrong insertion of epenthetic vowel.

The corpus is prepared as it is mentioned in Section 5.1, excluding abbreviations and acronyms. Then, words were first phonetically transcribed in order to serve as input for the rule-based epenthesis algorithm. Subsequently, syllabification is done using the output from epenthetic vowel insertion algorithm. To end with, the system gives the final result of epenthesis and syllabification. Table 5.3 presents the frequencies of syllable by their type in the test result corpus.

The evaluation by Amharic linguist expert showed an overall accuracy of the syllabifier 98.1%, which means the expert agreed on 981 of the total 1000 automatically syllabified words by the algorithm. In terms of word accuracy 98.1% and most of the syllabification errors are occurred due to wrong epenthetic vowel insertion or missed epenthetic vowel.

Table 5.3: Distribution of syllable patterns over the test set

Syllable Pattern	Frequencies			Total	Percentage (%)
	Word Initial	Word medial	Word final		
V	36	0	2	38	1.27
CV	352	690	579	1621	53.59
VC	64	25	5	94	3.10
VCC	74	0	0	74	2.44
CVC	450	310	342	1102	36.43
CVCC	24	0	72	96	3.17
Total	1000	1025	1000	3025	100%

Table 5.4 shows the summary of problems occurred during syllabification by the algorithm, as pointed out by the linguist expert while evaluating the test corpus. The analysis as shown in the Table 5.4, tells us the syllabification errors are occurred due to epenthesis error (with wrong insertion or with missed epenthetic vowel).

Table 5.4: Syllabification error types

Problem #.	Problem Descriptions	Total # in the test result corpus
1	Words which have wrong epenthetic insertions	2
2	Words with neglected epenthetic vowel insertions	11
3	Words with syllabification problem from problem 1 and 2	13
4	Syllabification problem from neglected epenthesis in CC sequence at word medial position	6
Total syllabification error		19

CHAPTER SIX
CONCLUSION AND RECOMMENDATION

6.1 Conclusion

Automatic syllabification is a task of segmenting a sequence of phonemes into syllables. It is a research area in the field of natural language processing for different languages in relation with TTS and ASR. The problem to syllabify words in Amharic language is that, the graphemes represent consonant-vowel assimilation, therefore while reading a text all the CV syllables are not uttered as expected. Moreover, the frequently occurring epenthetic vowel and gemination is not shown in the text. This problem can be solved using different techniques among which the rule-based approach is assumed to perform better in many languages. The other alternative approach is data-driven approach.

A rule-based approach is based on linguistic rules; the rules are generated based on the well known syllabification principles namely *maximum onset principles* and *sonority hierarchy principles*. If we have complete linguistic rules, better syllabification accuracy will be achieved.

A major effort was made in identifying and defining a formal set of linguistic rules for automatic syllabification and epenthesis of Amharic words. We have made a test using recorded Amharic words for each rules before designing the syllabification model and convinced with the result the design of the algorithm was made and implemented using C# programming language. Finally, the effectiveness of the proposed algorithm was demonstrated using a set of words extracted from Amharic words.

A total of 1000 Amharic words are collected from different linguistic research works and from Amharic dictionary. The corpus contains words with geminated consonants and cluster of consonants. Moreover, the corpus is recorded to look into the acoustic evidence. After the transliteration is made, words are given to the syllabifier. Then, the output of the system was given to expert and remark was given for each word in the result set. Empirical observation was also made on the recorded speech in accordance with the result of the algorithm.

As a result, the algorithm achieves an overall word accuracy of 98.1% or agreement rate with the expert. Therefore, it is possible to conclude that automatic syllabification of Amharic words can

be rule-based and linguistic rules and linguistic syllabification principles are important in implementing automatic syllabification and epenthesis in this language.

6.2 Recommendation

There are different research areas in natural language processing and syllabification is one of the issues. Syllabification is an important component of many speech and language processing systems, and the algorithm designed in this work is expected to be a significant contribution to the field, and mainly to researchers working on various aspects of the Amharic language, such as in Amharic TTS and ASR. Therefore, the researchers in the area can used the algorithm or the implemented system for processing syllabification as component in their research, mainly on speech applications or any other related natural language applications.

As a future work we would like suggest the following points:

- Gemination is one of the striking features of Amharic language and it occurs frequently in the language. Therefore, incorporating automatic gemination handling algorithm with the epenthesis and syllabification algorithm will be very beneficial.
- In our test, most of the errors arise from the missed or wrong epenthetic vowel in final consonant clusters; therefore study on final consonant cluster will improve the performance of the syllabifier.
- Stress and syllabification has their own relationship. Therefore, by studying their relationship thoroughly, we can have stress assignment algorithm.
- Performance investigation on both Amharic TTS and ASR.
- A comparison study using data-driven approaches.

REFERENCES

Adsett, C. R. and Marchand, Y. (2007). A Comparison of Data-Driven Automatic Syllabification Methods. In of Proceedings String Processing and Information Retrieval 16th International Symposium, Saariselkä, Finland.

Alemayehu Hailu. (1987). Lexical Stress in Amharic. In Journal of Ethiopian Studies: Vol XX, Addis Ababa University. Addis Ababa, Ethiopia.

Amsalu Aklilu. (2004). Amharic-English Dictionary, Second edition. Mega Publishing Enterprise, Addis Ababa, Ethiopia.

Aster Taddese. (1981). The syllable structure of Amharic and syllabification of Medial Consonant Clusters and Geminates. B.A thesis in Linguistics, Addis Ababa University, Addis Ababa, Ethiopia.

Bartlett, S., Kondrak, G., and Cherry, C. (2008). Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In Proceedings of Human Language Technologies: The 2008 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Columbus, Ohio, USA.

Bartlett, S., Kondrak, G., and Cherry, C. (2009). On the syllabification of phonemes. In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Colorado, USA.

Baye Ymam. (2010). “አጭርና ቀላል የአማርኛ ሰዋስው”: (“Short and simple Amharic Grammar”). Addis Ababa, Ethiopia.

Bigi, B., Meunier, C., Nesterenko, I., and Bertrand, R. (2009). Automatic detection of syllable boundaries in spontaneous speech. In proceedings of the International Conference on Language Resources. Valletta, Malta.

Bruck, M., Treiman, R., and Caravolas, M. (1995). Role of the syllable in the processing of spoken English: Evidence from a non word comparison task. Journal of Experimental Psychology: Human Perception and Performance.

Burileanu, D., and Negrescu, C. (2006). Prosody Modeling For AN Embedded TTS System Implementation. In Proceedings of 14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy.

Conrad, M., and. Jacobs, A. M. (2004). Replicating syllable frequency effects in Spanish in German: One more challenge to computational models of visual word recognition. Language and Cognitive Processes. *Language & Cognitive Processes*. Retrieved from http://journalsonline.tandf.co.uk/Index/10.1080/01690960344000224.

Cuayáhuitl, H. (2004). A Syllabification Algorithm for Spanish. Computational Linguistics and Intelligent Text Processing: 5th International Conference. Seoul, Korea.

Daelemans, W., and van den Bosch, A. (1992). Generalization performance of backpropagation learning on a syllabification task. Proceedings of TWLT3: Connectionism and Natural Language Processing, University of Twente, The Netherlands.

Duanmu, S. (2008). Syllable Structure the limit of variation. Oxford University Press.UK.

Ethnologue. (2004): *Languages of the World*. http://www.ethnologue.com/, last accessed date June 1, 2011.

Fujisaki, Y. (1995). Sonority and Its Role for Syllabification. Department of Humanities, Natural Language. Kochi University, Japan.

Getahun Amare. (2010).“ዘመናዊ የአማርኛ ሰዋስው በቀላል አቀራረብ”: (“Modern Amharic Grammar in a simple approach”), Addis Ababa, Ethiopia.

Hudson, G. (1996). Phonology of Ethiopian Languages: *The Handbook of Phonological Theory*. Glodsmith, John A. Blackwell Publishing.

Jany, C., Gordon, M., Nash, C. M., and Takara, N., (2007). How Universal Is The Sonority Hierarchy?:A Cross-Linguistic Acoustic Study. In proceedings the 16th International Congress of Phonetic Sciences. Saarbrücken, Germany.

Jeppe, B., Daniela, B., João, N., Miguel Sales, D., and Luis, C. (2009). Automatic syllabification for Danish Text-to-speech Systems. In proceedings of ISCA. Brighton, UK.

Jurafsky, D., and Martin, J. H. (2006). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

Kishore, S. P., and Black, A. W. (2003). Unit size in unit selection speech synthesis. In Proceedings of Eurospeech. Geneva, Switzerland.

Libossek, M. and Schiel, F. (2000). Syllable-based text-to-phoneme conversion for German. In Proceedings of the Sixth International Conference on Spoken Language Processing (ICSLP-2000). Beijing, China.

MacKinney, R. and Goddard, J. (2006). Syllabification using decision trees, early results on three languages. Metropolitan Autonomous University, Mexico.

Marchand, Y., Adsett, C. R., and Damper, R. I. (2009). Automatic Syllabification in English: A Comparison of Different Algorithms. SAGE journals online: Language and Speech.

Müller, K., Möbious, B., & Pescher, D. (2000). Inducing probabilistic syllable classes using multivariate clustering. In Proceedings of 38th annual meeting of the association for computational linguistics Morristown, NJ: Association for Computational Linguistics.

Mulugeta Seyoum (2001). The syllable Structure and Syllablification in Amharic. Masters of philosophy in general linguistic thesis. Department of Linguistics, Trondheim, Norway.

Oliveira, C., Moutinho, L. C., and Teixeira, A. (2005). On European Portuguese automatic syllabification. INTERSPEECH 2005 - Eurospeech, 9th European Conference on Speech Communication and Technology. Lisbon, Portugal.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*,Morgan Kaufmann Publishers Inc., San Mateo, CA.

Rose, S. (1997). Theoretical issues in comparative Ethio-Semitic Phonology and Morphology, PHD thesis, Department of Linguistics. McGill University, Montreal.

Sebsbie H/Mariam, Kishore, S. P., Black, A. W., Kumar, R., and Sangal, R. (2004). Unit Selection, IPA Equivalence Voice for Amharic Using Festvox", 5th Speech Synthesis Workshop. Pittsburgh.

Silvia, R., Daniela, B., João, N., Luis, C., and Miguel, S. D. (2008). Automatic Word Stress Marking and syllabification for Catalan TTS. In proceedings of ISCA. Brisbane Australia.

Solomon Teferra and Menzel, W. (2007). Syllable-Based Speech Recognition for Amharic. Proceedings of the 5thWorkshopon Important Unresolved Matters. Prague, Czech Republic.

Tadesse Anberbir and Gasser, M. (2011). Grapheme-to-Phoneme Conversion for Amharic Text-to-speech System. Conference on Human Language Technology for Development. Alexandria, Egypt.

Tian, J. (2004). Data-Driven Approaches for Automatic Detection of Syllable Boundaries. Nokia Research Center. Tampere, Finland.

Weerasinghe, R., Wasala, A., and Gamage, K. (2005). A Rule Based Syllabification Algorithm for Sinhala". Proceedings of 2nd International Joint Conference on Natural Language Processing (IJCNLP-05). Jeju Island, Korea.

Yule, G. (2006).The study of language: an introduction (3rd edition). Cambridge: Cambridge University Press.

APPENDICES

Appendix A: Amharic Phonetic List

IPA	Transcription	Amharic equivalence
Consonants		
[p]	[p]	ፕ
[t]	[t]	ት
[k]	[k]	ክ
[ʔ]	[ax][A]	ዕ
[b]	[b]	ብ
[d]	[d]	ድ
[g]	[g]	ግ
[p‘]	[px][P]	ጵ
[t‘]	[tx][T]	ጥ
[c‘]	[cx][C]	ጭ
[q]	[q]	ቅ
[f]	[f]	ፍ
[s]	[s]	ስ
[ʃ]	[sx][S]	ሽ
[h]	[h]	ህ
[s‘]	[xx] [X]	ጽ
[tʃ]	[c]	ች
[g‘]	[j]	ጅ
[m]	[m]	ም
[n]	[n]	ን
[n‘]	[nx] [N]	ኝ
[l]	[l]	ል
[r]	[r]	ር
[j]	[y]	ይ
[w]	[w]	ው
[v]	[v]	ቭ
[z]	[z]	ዝ
[z‘]	[zx] [Z]	ዥ
Vowels		
[E]	[e]	ኧ
[U]	[u]	ኡ
[I]	[ii] [I]	ኢ
[A]	[a]	ኣ
[e]	[ie][E]	ኤ
[ˆ]	[ix][i]	እ
[o]	[o]	ኦ

Appendix B: Syllabification algorithm

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;
namespace Syllabification
{
    public partial class Syllabification: Form
    {
        public Vowel_Insertion()
        {
            InitializeComponent();
        }
          string strFileName = null;
          string readcontent;

        private void Syllabification_button(object sender, EventArgs e)
        {


            string[] lines = txtInPut.Text.Split();
            try
            {
                foreach (string line in lines)
                {
                    string OutPut;
                    OutPut = InsertEpentheticVowel(line);
                    OutPut = Syllabify(OutPut);
                    OutPut = RemoveDuplicateBoundaryMarker(OutPut);
                    OutPut = Replace_digraphs(OutPut);
                    txtOutPut.AppendText(OutPut + Environment.NewLine);


                }

    File.WriteAllText("C:\\Syllabified_amharic_words.txt",   txtOutPut.Text);
                            }
            catch(Exception ex)
            {

                lblmsg.Text = ex.Message.ToString();

            }


        }

        public static string InsertEpentheticVowel(string inputStr)
        {
            //validation, if no value is provided then return an empty string
            if (string.IsNullOrEmpty(inputStr))
```

```
                return string.Empty;


            List<char> chars = new List<char>(inputStr.ToCharArray());


            //loop Left-Right through the input string.
            //RULE #1 consonant cluster at word initial is not allowed


            if (chars.Count >= 2)
            {
           if (isVowel(chars[0]) != true && isVowel(chars[1]) != true)
                {

                    chars.Insert(1, 'i');
                }
            }

        //RULE #3 CCC at word-medial is not allow in the language: CCC --CCiC
            for (int i = 1; i < chars.Count - 2; i++)
            {


if (isVowel(chars[i]) != true && isConsonant(chars[i + 1]) != false &&
chars[i] != chars[i + 1] && chars[i + 1] != chars[i + 2])
                    {
                        if (isVowel(chars[i + 2]) != true &&
SonorityOfConsonant(chars[i + 2]) > SonorityOfConsonant(chars[i+1]))
                            chars.Insert(i + 2, 'i');
                        else

                            chars.Insert(i + 1, 'i');
                    }        //RULE #4 C:C at word-medial -- C:iC
            for (int i = 1; i < chars.Count - 2; i++)
            {


 if (isVowel(chars[i]) != true && isConsonant(chars[i + 1]) != false &&
chars[i] == chars[i + 1])
                {
                    if (isVowel(chars[i + 2]) != true)
                        chars.Insert(i + 2, 'i');
                }

            }
            //RULE #5 CC: at word-medial -- CiC:
            for (int i = 1; i < chars.Count - 2; i++)
            {


                if (isVowel(chars[i]) != true && isConsonant(chars[i + 1]) !=
false && chars[i + 1] == chars[i + 2])
                {
                    if (isVowel(chars[i + 2]) != true)
                        chars.Insert(i + 1, 'i');
                }
```

```
            }
            //RULE #6 C:C: at word-medial -- C:iC:
            if (chars.Count > 5)
            {
                for (int i = 1; i < chars.Count - 2; i++)
                {


                    if (isVowel(chars[i]) != true && isConsonant(chars[i +
1]) != false && isVowel(chars[i + 2]) != true && isConsonant(chars[i + 3]) !=
false)
                    {
                        if (chars[i] == chars[i + 1] && chars[i + 2] ==
chars[i + 3])
                            chars.Insert(i + 1, 'i');
                    }

                }
            }
  //RULE #2 Consonant cluster at word final is allowed if and only if the two
  //consonant cluster sonority scale is equal or if the first consonant
//sonority scale is greater than that of the final
            for (int i = 0; i < chars.Count; i++)
            {
                if (i == chars.Count - 2)
                {

                    if (isVowel(chars[chars.Count - 1]) != true &&
isVowel(chars[chars.Count - 2]) != true)
                    {

                        if (chars[chars.Count - 1] != chars[chars.Count - 2]
&& SonorityOfConsonant(chars[chars.Count - 1]) >
SonorityOfConsonant(chars[chars.Count - 2]))
                            chars.Insert(chars.Count - 1, 'i');

                    }

                }

            }

            return new string(chars.ToArray());

        }
        public static Boolean isVowel(char Phoneme)
        {
            string[] vowels = new string[] { "A", "E", "I", "O", "U" };
            bool V = false;



            //loop through our vowel array
            for (int j = 0; j < vowels.Length; j++)
            {
```

```
                //if this current index of our word equals one
                //of the characters in our vowel array
                if (Phoneme.ToString().ToLower() == vowels[j].ToLower())
                    //V will be true  since the phoneme is a vowel
                    V = true;
                // chars.RemoveAt(i);
            }
            if (V != false && Phoneme!='A')
                return true;
            else
                return false;


        }
        public static Boolean isConsonant(char Phoneme)
        {
            string[] Consonants = new string[] { "P", "T", "K", "B", "D",
"G","T", "C", "Q", "F", "S", "H", "X","J", "M","N","L","R","Y","W","V","Z",
"X" };

            bool C = false;
            //validation, if no value is provided then return an empty string
            //if (string.IsNullOrEmpty(inputStr))
            //    return string.Empty;


            //loop backwards through the input string, this will help prevent
            //any problems with removing any characters from the string

            //now loop through our vowel array
            for (int j = 0; j < Consonants.Length; j++)
            {
                //if this current index of our word equals one
                //of the characters in our consonant array
                if (Phoneme.ToString().ToLower() == Consonants[j].ToLower())
               //if the phoneme is consonant then C will hold the value true
                    C = true;

            }
            if (C != false)
                return true;
            else
                return false;

        }
        //word_medial
        //word_final
        //this function checks for sonority and return the value

        public static int SonorityOfConsonant(char Phoneme)
        {

            if (Phoneme.Equals('p'))
            {
                return 1;
            }
            else if (Phoneme.Equals('t'))
```

```
{
    return 1;
}
else if (Phoneme.Equals('k') || Phoneme.Equals('K'))
{
    return 1;
}
else if (Phoneme.Equals('A'))
{
    return 1;
}
else if(Phoneme.Equals('b'))
{
    return 2;
}
else if (Phoneme.Equals('d'))
{
    return 2;
}
else if (Phoneme.Equals('g'))
{
    return 2;
}
else if (Phoneme.Equals('G'))
{
    return 2;
}
else if (Phoneme.Equals('P'))
{
    return 3;
}
else if (Phoneme.Equals('T'))
{
    return 3;
}
else if (Phoneme.Equals('q'))
{
    return 3;
}
else if (Phoneme.Equals('Q'))
{
    return 3;
}
else if (Phoneme.Equals('c'))
{
    return 4;
}
else if (Phoneme.Equals('j'))
{
    return 5;
}
else if (Phoneme.Equals('C'))
{
    return 6;
}
else if (Phoneme.Equals('f'))
{
```

```
        return 7;
    }
    else if (Phoneme.Equals('s'))
    {
        return 7;
    }
    else if (Phoneme.Equals('S'))
    {
        return 7;
    }
    else if (Phoneme.Equals('h'))
    {
        return 7;
    }
    else if (Phoneme.Equals('v'))
    {
        return 8;
    }
    else if (Phoneme.Equals('z'))
    {
        return 8;
    }
    else if (Phoneme.Equals('Z'))
    {
        return 8;
    }
    else if (Phoneme.Equals('X'))
    {
        return 9;
    }
    else if (Phoneme.Equals('H'))
    {
        return 9;
    }
    else if (Phoneme.Equals('n'))
    {
        return 10;
    }
    else if (Phoneme.Equals('m'))
    {
        return 10;
    }
    else if (Phoneme.Equals('N'))
    {
        return 10;
    }
    else if (Phoneme.Equals('l'))
    {
        return 11;
    }
    else if (Phoneme.Equals('r'))
    {
        return 11;
    }
    else if (Phoneme.Equals('w'))
    {
        return 12;
```

```
            }
            else if (Phoneme.Equals('y'))
            {
                return 13;
            }

            else
                return 0;
        }
        public static string Replace_digraphs(string inputStr)
        {

                    if (string.IsNullOrEmpty(inputStr))
                         return string.Empty;


                      List<char> chars = new
List<char>(inputStr.ToCharArray());
                      for (int i = 0; i <= chars.Count - 1; i++)
                      {
                          //consonant digraph replacement

                          if (chars[i] == 'T')
                          {
                              chars.Remove('T');
                              chars.Insert(i, 't');
                              chars.Insert(i + 1, 'x');

                          }
                          if (chars[i] == 'N')
                          {
                              chars.Remove('N');
                              chars.Insert(i, 'n');
                              chars.Insert(i + 1, 'x');

                          }
                          if (chars[i] == 'P')
                          {
                              chars.Remove('P');
                              chars.Insert(i, 'p');
                              chars.Insert(i + 1, 'x');

                          }
                          if (chars[i] == 'S')
                          {
                              chars.Remove('S');
                              chars.Insert(i, 's');
                              chars.Insert(i + 1, 'x');

                          }
                          if (chars[i] == 'C')
                          {
                              chars.Remove('C');
                              chars.Insert(i, 'c');
                              chars.Insert(i + 1, 'x');

                          }
```

```
if (chars[i] == 'A')
{
    chars.Remove('A');
    chars.Insert(i, 'a');
    chars.Insert(i + 1, 'x');

}
if (chars[i] == 'X')
{
    chars.Remove('X');
    chars.Insert(i, 'x');
    chars.Insert(i + 1, 'x');

}
if (chars[i] == 'Z')
{
    chars.Remove('Z');
    chars.Insert(i, 'z');
    chars.Insert(i + 1, 'x');

}
if (chars[i] == 'G')
{
    chars.Remove('G');
    chars.Insert(i, 'g');
    chars.Insert(i + 1, 'w');

}
if (chars[i] == 'Q')
{
    chars.Remove('Q');
    chars.Insert(i, 'q');
    chars.Insert(i + 1, 'w');

}
if (chars[i] == 'H')
{
    chars.Remove('H');
    chars.Insert(i, 'h');
    chars.Insert(i + 1, 'w');

}
if (chars[i] == 'F')
{
    chars.Remove('F');
    chars.Insert(i, 'f');
    chars.Insert(i + 1, 'w');

}
if (chars[i] == 'M')
{
    chars.Remove('M');
    chars.Insert(i, 'm');
    chars.Insert(i + 1, 'w');

}
if (chars[i] == 'R')
```

```
                {
                    chars.Remove('R');
                    chars.Insert(i, 'r');
                    chars.Insert(i + 1, 'w');

                }
                if (chars[i] == 'B')
                {
                    chars.Remove('B');
                    chars.Insert(i, 'b');
                    chars.Insert(i + 1, 'w');

                }

            }
             //vowel digraph replacement
            for (int i = 0; i <= chars.Count - 1; i++)
            {
                if (chars[i] == 'i')
                {
                    //chars.Remove('i');
                    //chars.Insert(i, 'i');
                    chars.Insert(i + 1, 'x');

                }

            }
            for (int i = 0; i <= chars.Count - 1; i++)
            {
                if (chars[i] == 'I')
                {
                    chars.Remove('I');
                    chars.Insert(i, 'i');
                    chars.Insert(i + 1, 'i');

                }

            }
            for (int i = 0; i <= chars.Count - 1; i++)
            {
                if (chars[i] == 'E')
                {
                    chars.Remove('E');
                    chars.Insert(i, 'i');
                    chars.Insert(i + 1, 'e');

                }

            }
            return new string(chars.ToArray());

        }
//the main syllabification algorithm

        public static string Syllabify(string inputStr)
        {
```

```
            if (string.IsNullOrEmpty(inputStr))
                return string.Empty;


            List<char> chars = new List<char>(inputStr.ToCharArray());


            //loop Left-Right through the input string.
            //six syllable templates in Amharic --> V, VC, VCC, CV, CVCC

            for (int i = 0; i < chars.Count; i++)
            {

                    //VV
                    if (i + 1 < chars.Count)
                    {
              if (isVowel(chars[i]) == true && isVowel(chars[i + 1]) == true)
                        {

                            chars.Insert(i + 1, '-');

                        }
                    }
                    //VCV
                    if (i + 2 < chars.Count)
                    {
                if (isVowel(chars[i]) == true && isVowel(chars[i + 1]) != true
&& isVowel(chars[i + 2]) == true)
                        {

                            chars.Insert(i + 1, '-');

                        }
                    }
                    //VCVC
                    if ((i + 3) < chars.Count)
                    {
               if (isVowel(chars[i]) == true && isVowel(chars[i + 1]) != true &&
isVowel(chars[i + 2]) == true && isVowel(chars[i + 3]) != true)

                            chars.Insert(i + 1, '-');


                    }
                    //VCCV
                    if ((i + 3) < chars.Count)
                    {
           if (isVowel(chars[i]) == true && isVowel(chars[i + 1]) != true &&
isVowel(chars[i + 2]) != true && isVowel(chars[i + 3]) == true)
                            chars.Insert(i + 2, '-');

                    }
                    //CVV
                    if ((i + 2) < chars.Count)
                    {
            if (isVowel(chars[i]) != true && isVowel(chars[i + 1]) == true &&
isVowel(chars[i + 2]) == true)
```

```
                            chars.Insert(i + 2, '-');

                        }
                        //CVC && CVCC
                        if ((i + 3) < chars.Count)
                        {
            if (isVowel(chars[i]) != true && isVowel(chars[i + 1]) == true &&
isVowel(chars[i + 2]) != true && isVowel(chars[i + 3]) != true && (i + 3) <
chars.Count - 1)
                            chars.Insert(i + 3, '-');

                        }
                        //CVCV
                        if (i + 3 < chars.Count)
                        {
             if (isVowel(chars[i]) != true && isVowel(chars[i + 1]) == true
&& isVowel(chars[i + 2]) != true && isVowel(chars[i + 3]) == true)
                            {
                                chars.Insert(i + 2, '-');
                            }
                        }


                }


            return new string(chars.ToArray());
        }

        public static string RemoveDuplicateBoundaryMarker(string str)
        {

           // Create the Regex.
            Regex r = new Regex(@"\--+");

            // Strip multiple boundary marker (-).
          str = r.Replace(str, @"-");

            return str.ToString();

        }

        private void button2_Click(object sender, EventArgs e)
        {

                label1.Visible = false;
                label2.Visible = false;
                label3.Visible = false;
                label7.Visible = false;
                label6.Visible = false;

            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                fileName.Text = "";
                txtInPut.Text = "";
                strFileName = openFileDialog1.FileName;
                fileName.Text = strFileName;
```

```
                if (strFileName == String.Empty)

               return; //user didn't select a file to opena
            }
            try
            {
                FileInfo file1 = new FileInfo(strFileName);

                StreamReader sr = new StreamReader(strFileName);

                readcontent = sr.ReadToEnd();
                // Reading content from the file and storing to a string
                sr.Close();
                txtInPut.Text = readcontent;
                txtInPut.Multiline = true;
            }
            catch (Exception ex)
            {
                label8.Text = ex.Message + "   File is not selected.";
            }
        }


    }
}
```

Appendix C: Complete test corpus given to expert for evaluation

Evaluated by: Dr. Mulugeta Seyoum
Addis Ababa University

SNO.	Transliterated Input word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	habt	habt	✓ (Correct)	✓ (Correct)
2	habtam	hab-tam	✓	✓
3	habtamnnet	hab-ta-mixn-net	✓	✓
4	ahgur	ah-gur	✓	✓
5	tewellaj	te-wel-laj	✓	✓
6	hullumm	hul-lumm	✓	✓
7	huket	hu-ket	✓	✓
8	hometxetxtxe	ho-me-txetx-txe	✓	✓
9	mnm	mixnm	✓	✓
10	beziihm	be-zii-hixm	✓	✓
11	ysberu	yixs-be-ru	✓	✓
12	dmmet	dixm-met	✓	✓
13	dmmetocc	dixm-me-tocc	✓	✓
14	berr	berr	✓	✓
15	lbb	lixbb	✓	✓
16	brd	bixrd	✓	✓
17	tnnx	tixnnx	Incorrect (missed epenthesis)	Missed epenthesis (tix-nixnx)
18	fyyel	fixy-yel	✓	✓
19	mengst	men-gixst	✓	✓
20	tmhrt	tixm-hixrt	✓	✓
21	mlkkt	mix-lixk-kixt	✓	✓
22	bet	bet	✓	✓
23	betocc	be-tocc	✓	✓
24	krr	kixrr	✓	✓
25	sew	sew	✓	✓
26	temariiwocc	te-ma-rii-wocc	✓	✓
27	wsxsxawocc	wixsx-sxa-wocc	✓	✓
28	berie	be-rie	✓	✓
29	tjja	tixj-ja	✓	✓
30	danxnxoc	danx-nxoc	✓	✓
31	tebiib	te-biib	✓	✓
32	tebiiban	te-bii-ban	✓	✓
33	hxxanat	hix-xxa-nat	✓	✓
34	mesfn	mes-fixn	✓	✓
35	assebe	as-se-be	✓	✓
36	blhat	bixl-hat	✓	✓
37	alleme	al-le-me	✓	✓
38	bruh	bix-ruh	✓	✓
39	hayahand	ha-ya-hand	✓	✓
40	axsrahand	axes-ra-hand	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	kft	kixft	✓	✓
2	kfftt	kixf-fixtt	✓	✓
3	kokeb	ko-keb	✓	✓
4	kewakbt	ke-wak-bixt	✓	✓
5	melak	me-lak	✓	✓
6	melaaxkt	me-la-axkt	✓	✓
7	axeknaf	axe-knaf	✓	✓
8	albas	al-bas	✓	✓
9	adbar	ad-bar	✓	✓
10	debr	de-bixr	Incorrect (wrong epenthesis)	debr
11	tekl	te-kixl	Incorrect (wrong epenthesis)	tekl
12	axetaklt	axe-tak-lixt	✓	✓
13	axenbesa	axen-be-sa	✓	✓
14	axenabst	axe-nab-sixt	✓	✓
15	axemlak	axem-lak	✓	✓
16	weyzazrt	wey-za-zrixt	✓	✓
17	weyzero	wey-ze-ro	✓	✓
18	gobez	go-bez	✓	✓
19	gobezazt	go-be-zazt	✓	✓
20	tlk	tixlk	✓	✓
21	tnnsx	tixn-nixsx	✓	✓
22	qeyy	qeyy	✓	✓
23	qeyayy	qe-yayy	✓	✓
24	tlalq	tix-lalq	Incorrect (missed epenthesis)	tix-la-lixq
25	tnannsx	tix-nan-nixsx	✓	✓
26	keffete	kef-fe-te	✓	✓
27	kefaffete	ke-faf-fe-te	✓	✓
28	sebabbere	se-bab-be-re		
29	txrutxruwum	txix-rutx-ru-wum	Incorrect (missed epenthesis)	txix-ru-txix-ru-wum
30	kifukfuwun	kix-fuk-fu-wun	Incorrect (missed epenthesis)	kix-fu-kix-fu-wun
31	berra	ber-ra	✓	✓
32	gebba	geb-ba	✓	✓
33	tesebbere	te-seb-be-re	✓	✓
34	dubbe	dub-be	✓	✓
35	gm	gixm	✓	✓
36	genna	gen-na	✓	✓
37	slbabot	sixl-ba-bot	✓	✓
38	ykkedal	yixk-ke-dal	✓	✓
39	ymmotal	yixm-mo-tal	✓	✓
40	axeqwaqwam	axe-qwa-qwam	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	axexxxxaxxxaf	axexx-xxixax-xxaf	✓	✓
2	fellege	fel-le-ge	✓	✓
3	fellgo	fel-lix-go	✓	✓
4	axemmarreze	axem-mar-re-ze	✓	✓
5	merreze	mer-re-ze	✓	✓
6	ywweqqeral	yixw-weq-qe-ral	✓	✓
7	awweqaqere	aw-we-qa-qe-re	✓	✓
8	yttekkelal	yixt-tek-ke-lal	✓	✓
9	assebaber	as-se-ba-ber	✓	✓
10	yffellegal	yixf-fel-le-gal	✓	✓
11	felagii	fe-la-gii	✓	✓
12	affelaleg	af-fe-la-leg	✓	✓
13	txeretxere	txe-re-txe-re	✓	✓
14	ytxtxeretxtxeralu	yixtx-txe-retx-txe-ra-lu	✓	✓
15	atxtxeratxer	atx-txe-ra-txer	✓	✓
16	mesekkere	me-sek-ke-re	✓	✓
17	ymesekral	yix-me-sek-ral	✓	✓
18	ammesekaker	am-me-se-ka-ker	✓	✓
19	kesekkese	ke-sek-ke-se	✓	✓
20	ykeseksal	yix-ke-sek-sal	Incorrect (missed epenthesis)	yix-ke-se-kix-sal
21	akkesakes	ak-ke-sa-kes	✓	✓
22	txrtxr	txixr-txixr	✓	✓
23	ayye	ay-ye	✓	✓
24	amma	am-ma	✓	✓
25	acxe	a-cxe	✓	✓
26	yacxal	ya-cxal	✓	✓
27	yttacxal	yixt-ta-cxal	✓	✓
28	astecxacxet	as-te-cxa-cxet	✓	✓
29	yayal	ya-yal	✓	✓
30	yttayyal	yixt-tay-yal	✓	✓
31	asteyayet	as-te-ya-yet	✓	✓
32	yamal	ya-mal	✓	✓
33	astemamat	as-te-ma-mat	✓	✓
34	acxcxede	acx-cxe-de	✓	✓
35	axcxwonxa	axcx-wo-nxa	✓	✓
36	yttacxcxedal	yixt-tacx-cxe-dal	✓	✓
37	raaxy	ra-axy	✓	✓
38	azzene	az-ze-ne	✓	✓
39	awweqe	aw-we-qe	✓	✓
40	axwqet	axw-qet	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	yttazzenal	yixt-taz-ze-nal	✓	✓
2	yttawwekal	yixt-taw-we-kal	✓	✓
3	azzenecc	az-ze-necc	✓	✓
4	awweqe	aw-we-qe	✓	✓
5	tesebbere	te-seb-be-re	✓	✓
6	abeddere	a-bed-de-re	✓	✓
7	ammelletxe	am-mel-le-txe	✓	✓
8	ammerete	am-me-re-te	✓	✓
9	ymmerretal	yixm-mer-re-tal	✓	✓
10	axemelletxe	axe-mel-le-txe	✓	✓
11	ameseggene	a-me-seg-ge-ne	✓	✓
12	ammenezxzxege	am-me-nezx-zxe-ge	✓	✓
13	yammenezxzxgal	yam-me-nezx-zxix-gal	✓	✓
14	amelekkete	a-me-lek-ke-te	✓	✓
15	axesera	axe-se-ra	✓	✓
16	axelleggam	axel-leg-gam	✓	✓
17	labbelab	lab-be-lab	✓	✓
18	axetsemam	axet-se-mam	✓	✓
19	axtrotxm	axixt-ro-txixm	✓	✓
20	axetkeftm	axet-kef-tixm	✓	✓
21	axessxetxe	axes-sxe-txe	✓	✓
22	axethedm	axet-he-dixm	✓	✓
23	kremt	kix-remt	✓	✓
24	blatta	bix-lat-ta	✓	✓
25	flat	fix-lat	✓	✓
26	trf	tixrf	✓	✓
27	axtxr	axix-txixr	✓	✓
28	rotxrotx	rotx-rotx	✓	✓
29	laslas	las-las	✓	✓
30	mesrat	mes-rat	✓	✓
31	mezgeb	mez-geb	✓	✓
32	mebrat	meb-rat	✓	✓
33	mefred	mef-red	✓	✓
34	dubie	du-bie	✓	✓
35	melkam	mel-kam	✓	✓
36	kubet	ku-bet	✓	✓
37	fellgo	fel-lix-go	✓	✓
38	fellgewal	fel-lix-gix-wal	✓	✓
39	qumet	qu-met	✓	✓
40	kenafr	ke-na-fixr	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	ab	ab	✓	✓
2	axeba	axe-ba	✓	✓
3	abal	a-bal	✓	✓
4	abalat	a-ba-lat	✓	✓
5	abererac	a-be-re-rac	✓	✓
6	abietuta	a-bie-tu-ta	✓	✓
7	adebabay	a-de-ba-bay	✓	✓
8	adelem	a-de-lem	✓	✓
9	adegenxa	a-de-ge-nxa	✓	✓
10	adderoc	ad-de-roc	✓	✓
11	addiis	ad-diis	✓	✓
12	addiisu	ad-dii-su	✓	✓
13	adragiiw	ad-ra-giiw	✓	✓
14	affriika	af-fix-rii-ka	✓	✓
15	znboc	zixn-boc	✓	✓
16	zienawii	zie-na-wii	✓	✓
17	zewdacew	zew-da-cew	✓	✓
18	zetenenxa	ze-te-ne-nxa	✓	✓
19	lemelleme	le-mel-le-me	✓	✓
20	lemlamie	lem-la-mie	✓	✓
21	lemmanxnet	lem-ma-nxix-net	✓	✓
22	lemezzege	le-mez-ze-ge	✓	✓
23	leslassannet	les-las-san-net	✓	✓
24	teleqalleqe	te-le-qal-le-qe	✓	✓
25	allamede	al-la-me-de	✓	✓
26	lqllaqie	lix-qixl-la-qie	✓	✓
27	mehal	me-hal	✓	✓
28	mehakkelenxnxa	me-hak-ke-lenx-nxa	✓	✓
29	mehayyem	me-hay-yem	✓	✓
30	mrreqa	mixr-re-qa	✓	✓
31	mrraqie	mixr-ra-qie	✓	✓
32	asmeremmere	as-me-rem-me-re	✓	✓
33	mrcxa	mixr-cxa	✓	✓
34	mietxmietxtxa	mietx-mietx-txa	✓	✓
35	temocxacxcxere	te-mo-cxacx-cxe-re	✓	✓
36	muqecxcxa	mu-qecx-cxa	✓	✓
37	maaxbel	maax-bel	✓	✓
38	maaxkel	maax-kel	✓	✓
39	maaxqeb	maax-qeb	✓	✓
40	maaxzen	maax-zen	✓	✓

	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	mnnxnxa	mix-nixnx-nxa	✓	✓
2	mncet	mixn-cet	✓	✓
3	mngllat	mixn-gixl-lat	✓	✓
4	mencxr	men-cxixr	✓	✓
5	mhwar	mixh-war	✓	✓
6	weyzeriit	wey-ze-riit	✓	✓
7	musxriit	musx-riit	✓	✓
8	mosxelleqe	mo-sxel-le-qe	✓	✓
9	temoggese	te-mog-ge-se	✓	✓
10	mocxellefe	mo-cxel-le-fe	✓	✓
11	mocxlafii	mocx-la-fii	✓	✓
12	selletene	sel-le-te-ne	✓	✓
13	sellatie	sel-la-tie	✓	✓
14	sltxun	sixl-txun	✓	✓
15	mesewwiiya	me-sew-wii-ya	✓	✓
16	seyyeme	sey-ye-me	✓	✓
17	seaxeliinnet	se-axe-liin-net	✓	✓
18	termetxemmetxe	ter-me-txem-me-txe	✓	✓
19	radde	rad-de	✓	✓
20	tereggetxe	te-reg-ge-txe	✓	✓
21	rebbadda	reb-bad-da	✓	✓
22	teraggetxe	te-rag-ge-txe	✓	✓
23	merragecxa	mer-ra-ge-cxa	✓	✓
24	arreggefe	ar-reg-ge-fe	✓	✓
25	sxellele	sxel-le-le	✓	✓
26	sxemmac	sxem-mac	✓	✓
27	sxebellelele	sxe-bel-le-le-le	✓	✓
28	sxkrkriit	sxixk-rixk-riit	✓	✓
29	sxeftx	sxeftx	✓	✓
30	sxaggete	sxag-ge-te	✓	✓
31	sxasx	sxasx	✓	✓
32	qellal	qel-lal	✓	✓
33	qllietam	qixl-lie-tam	✓	✓
34	qltxfftxf	qixl-txixf-fix-txixf	✓	✓
35	qlqql	qix-lixq-qixl	✓	✓
36	qelatxie	qe-la-txie	✓	✓
37	yqrta	yixq-rix-ta	✓	✓
38	qerrere	qer-re-re	✓	✓
39	meqerqeriiya	me-qer-qe-rii-ya	✓	✓
40	qbaqddus	qix-ba-qixd-dus	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	qntxtxabii	qix-nixtx-txa-bii	✓	✓
2	qentxbie	qen-txix-bie	✓	✓
3	qntxot	qixn-txot	✓	✓
4	qenecxcxebe	qe-necx-cxe-be	✓	✓
5	yeteqeddese	ye-te-qed-de-se	✓	✓
6	asqeddese	as-qed-de-se	✓	✓
7	asqeddasx	as-qed-dasx	✓	✓
8	qddst	qixd-dix-sixt	✓	✓
9	qddusnnet	qixd-du-sixn-net	✓	✓
10	tesxqedaddeme	tesx-qe-dad-de-me	✓	✓
11	qddasie	qixd-da-sie	✓	✓
12	qetxtxele	qetx-txe-le	✓	✓
13	qetxtxay	qetx-txay	✓	✓
14	qumartenxnxa	qu-mar-tenx-nxa	✓	✓
15	qxxbet	qixxx-bet	✓	✓
16	qullff	qul-lixff	✓	✓
17	bllzz	bixl-lixzz	✓	✓
18	quwanja	qu-wan-ja	✓	✓
19	beleqqete	be-leq-qe-te	✓	✓
20	bllcxa	bixl-lix-cxa	✓	✓
21	berera	be-re-ra	✓	✓
22	bltxgna	bixl-txixg-na	Incorrect (missed epenthesis)	bixl-txix-gix-na
23	abreqraqie	ab-req-ra-qie	✓	✓
24	berkatta	ber-kat-ta	Incorrect (missed epenthesis)	be-rix-kat-ta
25	brkattie	bixr-kat-tie	✓	✓
26	berrede	ber-re-de	✓	✓
27	bercxumma	ber-cxum-ma	✓	✓
28	tebesabbese	te-be-sab-be-se	✓	✓
29	beqelenxnxa	be-qe-lenx-nxa	✓	✓
30	bklet	bixk-let	✓	✓
31	bekkete	bek-ke-te	✓	✓
32	bezebbeze	be-zeb-be-ze	✓	✓
33	tebejajje	te-be-jaj-je	✓	✓
34	tebetxtxese	te-betx-txe-se	✓	✓
35	betxbacx	betx-bacx	✓	✓
36	biirrabiirro	biir-ra-biir-ro	✓	✓
37	bocxcxacxere	bocx-cxa-cxe-re	✓	✓
38	budannet	bu-dan-net	✓	✓
39	bahl	ba-hixl	✓	✓
40	balemoya	ba-le-mo-ya	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	bahtawii	bah-ta-wii	✓	✓
2	baleaxjj	ba-le-axixjj	✓	✓
3	balager	ba-la-ger	✓	✓
4	bhiereseb	bix-hie-re-seb	✓	✓
5	behieretenxnxa	be-hie-re-tenx-nxa	✓	✓
6	ixgziiabher	ixg-zii-ab-her	✓	✓
7	branna	bix-ran-na	✓	✓
8	blh	bixlh	✓	✓
9	tejj	tejj	✓	✓
10	mennesxa	men-ne-sxa	✓	✓
11	bandiira	ban-dii-ra	✓	✓
12	teblecxellecxe	teb-le-cxel-le-cxe	✓	✓
13	ziegnetun	zieg-ne-tun	✓	✓
14	mabreja	mab-re-ja	✓	✓
15	bsxq	bixsxq	✓	✓
16	abbesxaqqetxe	ab-be-sxaq-qe-txe	✓	✓
17	yemmaybeqa	yem-may-be-qa	✓	✓
18	bennene	ben-ne-ne	✓	✓
19	bnnanxnx	bixn-nanxnx	✓	✓
20	bekkele	bek-ke-le	✓	✓
21	tebekakkele	te-be-kak-ke-le	✓	✓
22	bkkt	bixk-kixt	✓	✓
23	abzanxnxaw	ab-zanx-nxaw	✓	✓
24	tebazxii	te-ba-zxii	✓	✓
25	tebetxatxasx	te-be-txa-txasx	✓	✓
26	begena	be-ge-na	✓	✓
27	betxebbetxe	be-txeb-be-txe	✓	✓
28	tmhrtawii	tixm-hixr-ta-wii	✓	✓
29	tenzazza	ten-zaz-za	✓	✓
30	zxergagga	zxer-gag-ga	✓	✓
31	tentxefetxtxefe	ten-txe-fetx-txe-fe	✓	✓
32	tenfwaffwa	ten-fix-waf-fix-wa	✓	✓
33	fwafwatie	fwa-fwa-tie	✓	✓
34	antxerawweze	an-txe-raw-we-ze	✓	✓
35	tentxefetxtxefe	ten-txe-fetx-txe-fe	✓	✓
36	tekettele	te-ket-te-le	✓	✓
37	tataqii	ta-ta-qii	✓	✓
38	tekliil	tek-liil	✓	✓
39	tekefaffele	te-ke-faf-fe-le	✓	✓
40	yegara	ye-ga-ra	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	akkaffele	ak-kaf-fe-le	✓	✓
2	zmta	zixm-ta	✓	✓
3	ziega	zie-ga	✓	✓
4	ziegnetun	zieg-ne-tun	✓	✓
5	zetxenenxnxa	ze-txe-nenx-nxa	✓	✓
6	zetxena	ze-txe-na	✓	✓
7	zerfoc	zer-foc	✓	✓
8	zemenawii	ze-me-na-wii	✓	✓
9	zelalem	ze-la-lem	✓	✓
10	zemed	ze-med	✓	✓
11	zelaqii	ze-la-qii	✓	✓
12	zedie	ze-die	✓	✓
13	zendro	zen-dix-ro	✓	✓
14	yxxeyefal	yix-xxe-ye-fal	✓	✓
15	yxxafal	yix-xxa-fal	✓	✓
16	yesxenal	yix-sxe-nal	✓	✓
17	yxxf	yixxxf	✓	✓
18	yzewal	yix-ze-wal	✓	✓
19	ywwessenal	yixw-wes-se-nal	✓	✓
20	yuhans	yu-hans	✓	✓
21	ytxenaqeqal	yix-txe-na-qe-qal	✓	✓
22	yqeyemwacewna	yix-qe-yem-wa-cew-na	✓	✓
23	ymesgen	yix-mes-gen	✓	✓
24	ynorewal	yix-no-re-wal	✓	✓
25	ymeretxalu	yix-me-re-txa-lu	✓	✓
26	ykefetletal	yix-ke-fet-le-tal	✓	✓
27	ykahiedal	yix-ka-hie-dal	✓	✓
28	yjemral	yix-jem-ral	✓	✓
29	ykefelacwal	yix-ke-fe-lac-wal	✓	✓
30	yjemralu	yix-jem-ra-lu	✓	✓
31	yhonalu	yix-ho-na-lu	✓	✓
32	ygetxmewal	yix-getx-me-wal	✓	✓
33	yfetal	yix-fe-tal	✓	✓
34	yeziegocn	ye-zie-go-cixn	✓	✓
35	yezetxenx	ye-ze-txenx	✓	✓
36	yezetxenx	ye-ze-txenx	✓	✓
37	yetemezegebebet	ye-te-me-ze-ge-be-bet	✓	✓
38	yetekesekesew	ye-te-ke-se-ke-sew	✓	✓
39	yetekahiedewn	ye-te-ka-hie-dewn	Incorrect (missed epenthesis)	ye-te-ka-hie-de-wixn
40	yetekenawenu	ye-te-ke-na-we-nu	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	yemiiyaqwaqumut	ye-mii-yaq-wa-qu-mut	✓	✓
2	yemiiyaqombetn	ye-mii-ya-qom-be-tixn	✓	✓
3	yemiiyadergat	ye-mii-ya-der-gat	✓	✓
4	yemiiwlu	ye-miiw-lu	✓	✓
5	yemaberetaca	ye-ma-be-re-ta-ca	✓	✓
6	yemagaletx	ye-ma-ga-letx	✓	✓
7	yellebetm	yel-le-be-tixm	✓	✓
8	yeixgziiabhiern	ye-ixg-zii-ab-hiern	✓	✓
9	yeiityopxyana	ye-iit-yopx-ya-na	✓	✓
10	yeiityopxya	ye-iit-yopx-ya	✓	✓
11	yeiihadieg	ye-ii-ha-dieg	✓	✓
12	yaleixdmieyacew	ya-le-ixd-mie-ya-cew	✓	✓
13	wedajocacew	we-da-jo-ca-cew	✓	✓
14	wedemiigenxubet	we-de-mii-ge-nxu-bet	✓	✓
15	wdnesx	wixd-nesx	✓	✓
16	wedeyageracew	we-de-ya-ge-ra-cew	✓	✓
17	wegebunm	we-ge-bunm	✓	✓
18	tetakkose	te-tak-ko-se	✓	✓
19	temelammeme	te-me-lam-me-me	✓	✓
20	zlglg	zixl-gixlg	✓	✓
21	azzennafele	az-zen-na-fe-le	✓	✓
22	tezxgoreggore	tezx-go-reg-go-re	✓	✓
23	zxgurgur	zxix-gur-gur	✓	✓
24	tedeladdele	te-de-lad-de-le	✓	✓
25	tederragii	te-der-ra-gii	✓	✓
26	tedenaggere	te-de-nag-ge-re	✓	✓
27	addabbele	ad-dab-be-le	✓	✓
28	tegafetxe	te-ga-fe-txe	✓	✓
29	meggonatxtxefe	meg-go-natx-txe-fe	✓	✓
30	meggonatxefiiya	meg-go-na-txe-fii-ya	✓	✓
31	tetxmmeme	te-txixm-me-me	✓	✓
32	tetxaffa	te-txaf-fa	✓	✓
33	tmlml	tixm-lix-mixl	✓	✓
34	tefetelleke	te-fe-tel-le-ke	✓	✓
35	tbiitenxnxa	tix-bii-tenx-nxa	✓	✓
36	tzzbt	tixz-zix-bixt	✓	✓
37	taggese	tag-ge-se	✓	✓
38	taxgst	tixax-gixst	✓	✓
39	tllqnnet	tixl-lix-qixn-net	✓	✓
40	thtnna	tixh-tixn-na	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	tkesxsxa	tix-kesx-sxa	✓	✓
2	tzzta	tixz-zix-ta	✓	✓
3	tdar	tix-dar	✓	✓
4	tgray	tixg-ray	✓	✓
5	cellta	cel-lix-ta	✓	✓
6	tecellese	te-cel-le-se	✓	✓
7	cebeccebe	ce-bec-ce-be	✓	✓
8	cenkar	cen-kar	✓	✓
9	cekkway	cek-kix-way	✓	✓
10	ceggere	ceg-ge-re	✓	✓
11	cenakkere	ce-nak-ke-re	✓	✓
12	asceggarii	as-ceg-ga-rii	✓	✓
13	cxeffecxefe	cxef-fe-cxe-fe	✓	✓
14	mecefcefiya	me-cef-ce-fix-ya	✓	✓
15	hdar	hix-dar	✓	✓
16	hayyal	hay-yal	✓	✓
17	hatxiiyatenxnxa	ha-txii-ya-tenx-nxa	✓	✓
18	hddag	hixd-dag	✓	✓
19	neh	neh	✓	✓
20	nehasie	ne-ha-sie	✓	✓
21	nessa	nes-sa	✓	✓
22	axennesassa	axen-ne-sas-sa	✓	✓
23	nerrere	ner-re-re	✓	✓
24	nesennese	ne-sen-ne-se	✓	✓
25	nesxsxetxe	nesx-sxe-txe	✓	✓
26	neqaqqele	ne-qaq-qe-le	✓	✓
27	menqeya	men-qe-ya	✓	✓
28	saran	sa-ran	✓	✓
29	samii	sa-mii	✓	✓
30	same	sa-me	✓	✓
31	rucxcxa	rucx-cxa	✓	✓
32	robna	rob-na	✓	✓
33	saybela	say-be-la	✓	✓
34	riesa	rie-sa	✓	✓
35	rguz	rix-guz	✓	✓
36	reta	re-ta	✓	✓
37	rbrb	rixb-rixb	✓	✓
38	raq	raq	✓	✓
39	qusl	qu-sixl	✓	✓
40	qtxr	qix-txixr	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	nat	nat	✓	✓
2	nacew	na-cew	✓	✓
3	neger	ne-ger	✓	✓
4	necc	necc	✓	✓
5	nefsie	nef-sie	✓	✓
6	nenx	nenx	✓	✓
7	nen	nen	✓	✓
8	nebrmma	neb-rixm-ma	✓	✓
9	nb	nixb	✓	✓
10	necxcx	necxcx	✓	✓
11	nuro	nu-ro	✓	✓
12	tenafaqii	te-na-fa-qii	✓	✓
13	tenefafaqi	te-ne-fa-fa-qix	✓	✓
14	nssha	nixs-six-ha	✓	✓
15	manoriiya	ma-no-rii-ya	✓	✓
16	gena	ge-na	✓	✓
17	genna	gen-na	✓	✓
18	mescxa	mes-cxa	✓	✓
19	mesasat	me-sa-sat	✓	✓
20	merietu	me-rie-tu	✓	✓
21	merii	me-rii	✓	✓
22	lahunu	la-hu-nu	✓	✓
23	ammeme	am-me-me	✓	✓
24	amelkac	a-mel-kac	✓	✓
25	amlak	am-lak	✓	✓
26	tammeme	tam-me-me	✓	✓
27	ammasx	am-masx	✓	✓
28	mehaym	me-haym	Incorrect (missed epenthesis)	me-ha-yixm
29	ammetxtxetxe	am-metx-txe-txe	✓	✓
30	amba	am-ba	✓	✓
31	ambesxsxa	am-besx-sxa	✓	✓
32	mamonxnxa	ma-monx-nxa	✓	✓
33	assese	as-se-se	✓	✓
34	arrami	ar-ra-mix	✓	✓
35	asar	a-sar	✓	✓
36	ixssrat	ixs-six-rat	✓	✓
37	ixsrenxnxa	ixs-renx-nxa	✓	✓
38	ixssr	ixs-sixr	✓	✓
39	assere	as-se-re	✓	✓
40	ixssratie	ixs-six-ra-tie	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	txebiib	txe-biib	✓	✓
2	txebiiban	txe-bii-ban	✓	✓
3	txebn	txe-bixn	✓	✓
4	txefta	txef-ta	✓	✓
5	txeqlala	txeq-la-la	✓	✓
6	txeyaqiinet	txe-ya-qii-net	✓	✓
7	txfatenxoc	txix-fa-te-nxoc	✓	✓
8	txnat	txix-nat	✓	✓
9	txiennet	txien-net	✓	✓
10	txomacewn	txo-ma-cewn	✓	✓
11	wcx	wixcx	✓	✓
12	wedaj	we-daj	✓	✓
13	was	was	✓	✓
14	txqmoc	txixq-moc	✓	✓
15	txqmt	txixq-mixt	✓	✓
16	txrat	txix-rat	✓	✓
17	txret	txix-ret	✓	✓
18	txrie	txix-rie	✓	✓
19	txrrii	txixr-rii	✓	✓
20	txru	txix-ru	✓	✓
21	txrunesx	txix-ru-nesx	✓	✓
22	txur	txur	✓	✓
23	txutocwan	tu-toc-wan	✓	✓
24	txutocwa	txu-toc-wa	✓	✓
25	txuwafun	txu-wa-fun	✓	✓
26	tekl	te-kixl	✓	✓
27	telixko	te-lix-ko	✓	✓
28	temedebe	te-me-de-be	✓	✓
29	tekenawnwal	te-ke-naw-nix-wal	✓	✓
30	tekuwasx	te-ku-wasx	✓	✓
31	tekeffetu	te-kef-fe-tu	✓	✓
32	tekebbere	te-keb-be-re	✓	✓
33	tejemmere	te-jem-me-re	✓	✓
34	tegsaxxx	teg-saxx	✓	✓
35	tegojiiwoc	te-go-jii-woc	✓	✓
36	tegenebu	te-ge-ne-bu	✓	✓
37	tegelxxo	te-gel-xxo	✓	✓
38	tegebiiwn	te-ge-biiwn	Incorrect (missed epenthesis)	te-ge-bii-wixn
39	tegbaroc	teg-ba-roc	✓	✓
40	tegbarawii	teg-ba-ra-wii	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	amelekkete	a-me-lek-ke-te	✓	✓
2	ygelexxna	yix-ge-lexx-na	✓	✓
3	yfexxembacewal	yix-fe-xxem-ba-ce-wal	✓	✓
4	ygaletxalu	yix-ga-le-txa-lu	✓	✓
5	yhiedal	yix-hie-dal	✓	✓
6	liig	liig	✓	✓
7	liikahied	lii-ka-hied	✓	✓
8	liilewetx	lii-le-wetx	✓	✓
9	liiqwaqwam	lii-qwa-qwam	✓	✓
10	liiyakahied	lii-ya-ka-hied	✓	✓
11	liiyastelalf	lii-yas-te-lalf	Incorrect (missed epenthesis)	lii-yas-te-la-lixf
12	siiyayu	sii-ya-te-yu	✓	✓
13	liiyayu	lii-ya-yu	✓	✓
14	liisetx	lii-setx	✓	✓
15	liiyakahiid	lii-ya-ka-hiid	✓	✓
16	siiqoretx	Sii-qo-retx	✓	✓
17	liiqoretx	lii-qo-retx	✓	✓
18	lieliitna	lie-liit-na	✓	✓
19	kewer	ke-wer	✓	✓
20	kezemedocwa	ke-ze-me-doc-wa	✓	✓
21	kezmut	kez-mut	✓	✓
22	kfatn	kix-fa-tixn	✓	✓
23	kffl	kixf-fixl	✓	✓
24	keteketelut	ke-te-ke-te-lut	✓	✓
25	kesost	ke-sost	✓	✓
26	yexxadqan	ye-xxad-qan	✓	✓
27	yemukera	ye-mu-ke-ra	✓	✓
28	yemusna	ye-mus-na	✓	✓
29	yewenbedienet	ye-wen-be-die-net	✓	✓
30	tassere	tas-se-re	✓	✓
31	mettaseriiya	met-ta-se-rii-ya	✓	✓
32	yesxggr	ye-sxixg-gixr	✓	✓
33	arramii	ar-ra-mii	✓	✓
34	asetxtxa	a-setx-txa	✓	✓
35	asefesefe	a-se-fe-se-fe	✓	✓
36	astewale	as-te-wa-le	✓	✓
37	asrekkabii	as-rek-ka-bii	✓	✓
38	asderrege	as-der-re-ge	✓	✓
39	asderragii	as-der-ra-gii	✓	✓
40	mastawesxa	mas-ta-we-sxa	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	arrekakkebe	ar-re-kak-ke-be	✓	✓
2	astemarii	as-te-ma-rii	✓	✓
3	asxoleqe	a-sxo-le-qe	✓	✓
4	asxofe	a-sxo-fe	✓	✓
5	asxwafii	asx-wa-fii	✓	✓
6	asxokesxsxke	a-sxo-kesx-sxix-ke	✓	✓
7	abay	a-bay	✓	✓
8	aqqoranxnxe	aq-qo-ranx-nxe	✓	✓
9	teqoranxnxe	te-qo-ranx-nxe	✓	✓
10	aqqwaqwame	aq-qix-waq-wa-me	✓	✓
11	bhierawii	bix-hie-ra-wii	✓	✓
12	gra	gix-ra	✓	✓
13	gobez	go-bez	✓	✓
14	gubnxt	gub-nxixt	✓	✓
15	glxxnet	gixl-xxix-net	✓	✓
16	gubaxie	gu-ba-axie	✓	✓
17	gmasx	gix-masx	✓	✓
18	fetawrarii	fe-taw-ra-rii	✓	✓
19	gabcacew	gab-ca-cew	✓	✓
20	fndata	fixn-da-ta	✓	✓
21	fiitien	fii-tien	✓	✓
22	atallale	a-tal-la-le	✓	✓
23	atosetose	a-to-se-to-se	✓	✓
24	anekkete	a-nek-ke-te	✓	✓
25	qetxqatxa	qetx-qa-txa	✓	✓
26	angeraggere	an-ge-rag-ge-re	✓	✓
27	angeragarii	an-ge-ra-ga-rii	✓	✓
28	angewalelay	an-ge-wa-le-lay	✓	✓
29	tengefeggefe	ten-ge-feg-ge-fe	✓	✓
30	tengedgaj	ten-ged-gaj	✓	✓
31	anjjet	a-nixj-jet	✓	✓
32	andmm	an-dixmm	✓	✓
33	angagga	an-gag-ga	✓	✓
34	akkenannebe	ak-ke-nan-ne-be	✓	✓
35	takkeme	tak-ke-me	✓	✓
36	awegga	a-weg-ga	✓	✓
37	awrariis	aw-ra-riis	✓	✓
38	azzele	az-ze-le	✓	✓
39	tzaz	tix-zaz	✓	✓
40	mettedaderiiya	met-te-da-de-rii-ya	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	ajja	aj-ja	✓	✓
2	tajjebe	taj-je-be	✓	✓
3	gsat	gix-sat	✓	✓
4	asaffarii	a-saf-fa-rii	✓	✓
5	ixbiita	ix-bii-ta	✓	✓
6	ixbamm	ix-bamm	✓	✓
7	ixmbossa	ixm-bos-sa	✓	✓
8	ixmmyye	ixm-mixy-ye	✓	✓
9	ixndihumm	ixn-dix-humm	✓	✓
10	anccii	a-nixc-cii	✓	✓
11	ixnkoy	ixn-koy	✓	✓
12	ixnkokko	ixn-kok-ko	✓	✓
13	ixnziira	ixn-zii-ra	✓	✓
14	oriitawii	o-rii-ta-wii	✓	✓
15	hgge	hixg-ge	✓	✓
16	ortodoksawii	or-to-dok-sa-wii	✓	✓
17	ixgeliit	ix-ge-liit	✓	✓
18	kellele	kel-le-le	✓	✓
19	klkl	kixl-kixl	✓	✓
20	mekkelakeya	mek-ke-la-ke-ya	✓	✓
21	tekessete	te-kes-se-te	✓	✓
22	kereddede	ke-red-de-de	✓	✓
23	kessiitta	kes-siit-ta	✓	✓
24	kebbebe	keb-be-be	✓	✓
25	tekettebe	te-ket-te-be	✓	✓
26	kettownmm	ket-tow-nixmm	✓	✓
27	keccece	kec-ce-ce	✓	✓
28	kflfay	kixf-lix-fay	✓	✓
29	kfyya	kix-fixy-ya	✓	✓
30	kedda	ked-da	✓	✓
31	kedatenxnxa	ke-da-tenx-nxa	✓	✓
32	kfunxnxa	kix-funx-nxa	✓	✓
33	kftenxnxa	kixf-tenx-nxa	✓	✓
34	kulaliit	ku-la-liit	✓	✓
35	kuffnxnx	kuf-fixnxnx	✓	✓
36	krstiiya	kixr-six-tii-ya	✓	✓
37	krstnna	kixr-six-tixn-na	✓	✓
38	kremt	kix-remt	✓	✓
39	kurreja	kur-re-ja	✓	✓
40	kbriit	kixb-riit	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	kutkuwato	kut-ku-wa-to	✓	✓
2	tekunesennese	te-ku-ne-sen-ne-se	✓	✓
3	kunsns	kun-sixns	✓	✓
4	welela	we-le-la	✓	✓
5	wehynii	weh-yix-nii	✓	✓
6	wllqat	wixl-lix-qat	✓	✓
7	wlqlq	wixl-qixlq	Incorrect (missed epenthesis)	wixl-qix-lixq
8	awelaleqe	a-we-la-le-qe	✓	✓
9	awwalede	aw-wa-le-de	✓	✓
10	awwalaj	aw-wa-laj	✓	✓
11	tewallede	te-wal-le-de	✓	✓
12	awelalleqe	a-we-lal-le-qe	✓	✓
13	wld	wixld	✓	✓
14	awwelaggede	aw-we-lag-ge-de	✓	✓
15	wemmete	wem-me-te	✓	✓
16	werrere	wer-re-re	✓	✓
17	welled	wel-led	✓	✓
18	wellad	wel-lad	✓	✓
19	wllaj	wixl-laj	✓	✓
20	awwallede	aw-wal-le-de	✓	✓
21	wraj	wix-raj	✓	✓
22	wrdet	wixr-det	✓	✓
23	wrjbbnxx	wixr-jixb-bixnx	✓	✓
24	werarrede	we-rar-re-de	✓	✓
25	wrrrd	wixr-rixrd	✓	✓
26	werrede	wer-re-de	✓	✓
27	werienxnxa	we-rienx-nxa	✓	✓
28	wesentenxnxa	we-sen-tenx-nxa	✓	✓
29	wesxsxaqqa	wesx-sxaq-qa	✓	✓
30	wesxeba	we-sxe-ba	✓	✓
31	wesxekkete	we-sxek-ke-te	✓	✓
32	weqqese	weq-qe-se	✓	✓
33	weqqesa	weq-qe-sa	✓	✓
34	webbeqa	web-be-qa	✓	✓
35	wetewette	we-te-wet-te	✓	✓
36	wenjel	wen-jel	✓	✓
37	wendnnet	wen-dixn-net	✓	✓
38	wenekkere	we-nek-ke-re	✓	✓
39	wenjay	wen-jay	✓	✓
40	wenjelenxnxa	wen-je-lenx-nxa	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	xxotan	xxo-tan	✓	
2	xxotacew	xxo-ta-cew	✓	
3	xxhfet	xxixh-fet	✓	
4	xxgie	xxix-gie	✓	
5	xxetxta	xxetx-ta	Incorrect (missed epenthesis)	xxe-txix-ta
6	xxensa	xxen-sa	✓	✓
7	xxeggaye	xxeg-ga-ye	✓	✓
8	xxegana	xxe-ga-na	✓	✓
9	xxegga	xxeg-ga	✓	✓
10	xxdat	xxix-dat	✓	✓
11	xxadqan	xxad-qan	✓	✓
12	wetxtewal	wetx-te-wal	✓	✓
13	wesdalet	wes-da-let	✓	✓
14	wenjelenxoc	wen-je-le-nxoc	✓	✓
15	wendmamac	wen-dix-ma-mac	✓	✓
16	weleteyesus	we-le-te-ye-sus	✓	✓
17	welajocu	we-la-jo-cu	✓	✓
18	wegenocacew	we-ge-no-ca-cew	✓	✓
19	wediiyawnu	we-dii-yaw-nu	✓	✓
20	wedeyageracew	we-de-ya-ge-ra-cew	✓	✓
21	wedemiigenxubet	we-de-mii-ge-nxu-bet	✓	✓
22	txemamocnv	txe-ma-moc-nixv	✓	✓
23	txabiiyawocn	txa-bii-ya-wo-cixn	✓	✓
24	fwafwatie	fwa-fwa-tie	✓	✓
25	teweledulacew	te-we-le-du-la-cew	✓	✓
26	tetxenaqqwal	te-txe-naq-qix-wal	✓	✓
27	tesxefnwal	te-sxef-nix-wal	Incorrect (wrong epenthesis)	te-sxef-nwal
28	tesetxtwacew	te-setx-tix-wa-cew	✓	✓
29	sxmagliewoc	sxix-mag-lie-woc	✓	✓
30	srawocn	six-ra-wo-cixn	✓	✓
31	smmnetocn	sixm-mix-ne-to-cixn	✓	✓
32	smmnet	sixm-mix-net	✓	✓
33	smmnetoc	sixm-mix-ne-toc	✓	✓
34	sltxanat	sixl-txa-nat	✓	✓
35	slenekubacew	six-le-ne-ku-ba-cew	✓	✓
36	sl	sixl	✓	✓
37	siiyasgeddew	sii-yas-ged-dew	✓	✓
38	siiwetxu	sii-we-txu	✓	✓
39	siiqeldubet	sii-qel-du-bet	✓	✓
40	siihiedu	sii-hie-du	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	saytenxa	say-te-nxa	✓	✓
2	sayqer	say-qer	✓	✓
3	sayasdebedbwat	sa-yas-de-bed-bix-wat	✓	✓
4	saniitiesxn	sa-nii-tie-sxixn	✓	✓
5	mkniiyat	mixk-nii-yat	✓	✓
6	mngiiziem	mixn-gii-ziem	✓	✓
7	mnxotuna	mix-nxo-tu-na	✓	✓
8	mnzarii	mixn-za-rii	✓	✓
9	mogziit	mog-ziit	✓	✓
10	mkrn	mixk-rixn	✓	✓
11	metxetx	me-txetx	✓	✓
12	memeriiyawoc	me-me-rii-ya-woc	✓	✓
13	melewawecxa	me-le-wa-we-cxa	✓	✓
14	melaktenxoc	me-lak-te-nxoc	✓	✓
15	mehonacnen	me-ho-nac-nen	✓	✓
16	meglexx	meg-lexx	✓	✓
17	megdelawiitna	meg-de-la-wiit-na	✓	✓
18	mebrathayl	meb-rat-hayl	✓	✓
19	mazawer	ma-za-wer	✓	✓
20	mastaweqiiya	mas-ta-we-qii-ya	✓	✓
21	masgebatun	mas-ge-ba-tun	✓	✓
22	maqwaqwamiiya	maq-waq-wa-mii-ya	✓	✓
23	leixntxotxo	le-ixn-txo-txo	✓	✓
24	leixgziiabhier	le-ixg-zii-ab-hier	✓	✓
25	leaxxyandandum	leax-yan-dan-dum	Incorrect (missed epenthesis)	le-axix-yan-dan-dum
26	lebalehabtoc	le-ba-le-hab-toc	✓	✓
27	laltesxefenu	lal-te-sxe-fe-nu	✓	✓
28	kwas	kwas	✓	✓
29	ktbat	kixt-bat	✓	✓
30	kesebat	ke-se-bat	✓	✓
31	krstiiyan	kixr-six-tii-yan	✓	✓
32	konfiedieriesxn	kon-fie-die-rie-sxixn	✓	✓
33	wettadderawii	wet-tad-de-ra-wii	✓	✓
34	wendmm	wen-dixmm	✓	✓
35	wendmmamacnnet	wen-dixm-ma-ma-cixn-net	✓	✓
36	yemendmm	ye-men-dixmm	✓	✓
37	wengel	wen-gel	✓	✓
38	wekkiil	wek-kiil	✓	✓
39	weyyebe	wey-ye-be	✓	✓
40	weymm	we-yixmm	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	kewelajocwa	ke-we-la-joc-wa	✓	✓
2	ketmhrtu	ket-mixh-rix-tu	✓	✓
3	ketemezegebebet	ke-te-me-ze-ge-be-bet	✓	✓
4	ketegenebu	ke-te-ge-ne-bu	✓	✓
5	kemiil	ke-miil	✓	✓
6	weddese	wed-de-se	✓	✓
7	wgenxnxa	wix-genx-nxa	✓	✓
8	wetxtxa	wetx-txa	✓	✓
9	ixjj	ixjj	✓	✓
10	awwaci	aw-wa-cix	✓	✓
11	wanza	wan-za	✓	✓
12	wqyanos	wixq-ya-nos	✓	✓
13	wsxlsxl	wixsx-lix-sxixl	✓	✓
14	wddiela	wixd-die-la	✓	✓
15	wdddr	wixd-dix-dixr	✓	✓
16	wjmbr	wixj-mix-bixr	✓	✓
17	kamst	kam-sixt	✓	✓
18	jmmatien	jixm-ma-tien	✓	✓
19	jemro	jem-ro	✓	✓
20	jemmeru	jem-me-ru	✓	✓
21	jemmere	je-mme-re	✓	✓
22	jemmer	jem-mer	✓	✓
23	jegnnetwo	je-gixn-net-wo	✓	✓
24	jbutii	jix-bu-tii	✓	✓
25	ixsrenxoc	ixs-re-nxoc	✓	✓
26	drjtocn	dixr-jix-to-cixn	✓	✓
27	dnq	dixnq	✓	✓
28	zemeca	ze-me-ca	✓	✓
29	zlflf	zixl-fixlf	✓	✓
30	qenxnx	qenxnx	✓	✓
31	zemmete	zem-me-te	✓	✓
32	mezmur	mez-mur	✓	✓
33	zerkkata	ze-rixk-ka-ta	✓	✓
34	zerffafa	ze-rixf-fa-fa	✓	✓
35	zrfiiya	zixr-fii-ya	✓	✓
36	zrtxrt	zixr-txixrt	✓	✓
37	zrkrk	zixr-kixrk	✓	✓
38	zereffe	ze-ref-fe	✓	✓
39	zebzaba	zeb-za-ba	✓	✓
40	zebatelo	ze-ba-te-lo	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	zebebenaynnet	ze-be-be-na-yixn-net	✓	✓
2	mezzebabeca	mez-ze-ba-be-ca	✓	✓
3	zebbete	zeb-be-te	✓	✓
4	zbrqrq	zixb-rixq-rixq	✓	✓
5	zrrya	zixr-rix-ya	✓	✓
6	gommen	gom-men	✓	✓
7	zerra	zer-ra	✓	✓
8	zenneqe	zen-ne-qe	✓	✓
9	znab	zix-nab	✓	✓
10	zenkkata	ze-nixk-ka-ta	✓	✓
11	tezenegga	te-ze-neg-ga	✓	✓
12	zennacx	zen-nacx	✓	✓
13	zend	zend	✓	✓
14	mstxiir	mixs-txiir	✓	✓
15	zekke	zek-ke	✓	✓
16	zewwere	zew-we-re	✓	✓
17	tezawwere	te-zaw-we-re	✓	✓
18	zewwetere	zew-we-te-re	✓	✓
19	azewettere	a-ze-wet-te-re	✓	✓
20	zeggeme	zeg-ge-me	✓	✓
21	zegejje	ze-gej-je	✓	✓
22	zgjju	zix-gixj-ju	✓	✓
23	memmezgebiiya	mem-mez-ge-bii-ya	✓	✓
24	mexxhiet	mexx-hiet	✓	✓
25	zeffeqe	zef-fe-qe	✓	✓
26	zefetxtxete	ze-fetx-txe-te	✓	✓
27	zgba	zixg-ba	✓	✓
28	zggtenxnxa	zixg-gix-tenx-nxa	✓	✓
29	zggta	zixg-gix-ta	✓	✓
30	mazoriiya	ma-zo-rii-ya	✓	✓
31	znnenxnxa	zixn-nenx-nxa	✓	✓
32	tezuwazuware	te-zu-wa-zu-wa-re	✓	✓
33	zxlletxe	zxixl-le-txe	✓	✓
34	tezxelletxe	te-zxel-le-txe	✓	✓
35	zxeqezxzxeqe	zxe-qezx-zxe-qe	✓	✓
36	zxret	zxix-ret	✓	✓
37	zxwazxwwie	zxwa-zxixw-wie	✓	✓
38	zxrat	zxix-rat	✓	✓
39	zxww	zxixww	✓	✓
40	znnar	zixn-nar	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	yewwaknnet	yew-wa-kixn-net	✓	✓
2	yeftxtxnxnx	ye-fixtx-txixnxnx	✓	✓
3	yannmma	yan-nixm-ma	✓	✓
4	yann	yann	✓	✓
5	yanxnxaw	yanx-nxaw	✓	✓
6	yacc	yacc	✓	✓
7	yaccatnna	yac-ca-tixn-na	✓	✓
8	yaw	yaw	✓	✓
9	yawllh	ya-wixl-lixh	✓	✓
10	yawnna	ya-wixn-na	✓	✓
11	yawkko	ya-wixk-ko	✓	✓
12	meyzxa	mey-zxa	✓	✓
13	ayyayaz	ay-ya-yaz	✓	✓
14	yhmma	yix-hixm-ma	✓	✓
15	yhmm	yix-hixmm	✓	✓
16	yhnn	yix-hixnn	✓	✓
17	ylunxnxta	yix-lunx-nxix-ta	✓	✓
18	ylq	yixlq	Incorrect (missed epenthesis)	yix-lixg
19	ymam	yix-mam	✓	✓
20	ygbanxnx	yixg-banxnx	✓	✓
21	ylqunmm	yixl-qu-nixmm	✓	✓
22	dldy	dixl-dixy	✓	✓
23	tedeladele	te-de-la-de-le	✓	✓
24	dmmera	dixm-me-ra	✓	✓
25	dmmr	dixm-mixr	✓	✓
26	demmere	dem-me-re	✓	✓
27	tedemarii	te-de-ma-rii	✓	✓
28	dembocc	dem-bocc	✓	✓
29	derresellet	der-re-sel-let	✓	✓
30	derresenxnx	der-re-senxnx	✓	✓
31	drsan	dixr-san	✓	✓
32	dereggme	de-reg-gix-me	✓	✓
33	desstenxnxa	des-six-tenx-nxa	✓	✓
34	dqdq	dixq-dixq	✓	✓
35	dblqlq	dixb-lixq-lixq	✓	✓
36	dbbq	dixb-bixq	✓	✓
37	dbbnn	dixb-bixnn	✓	✓
38	debzazza	deb-zaz-za	✓	✓
39	debbal	deb-bal	✓	✓
40	deneqqole	de-neq-qo-le	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	denbenxnxa	den-benx-nxa	✓	✓
2	dendanna	den-dan-na	✓	✓
3	deneggeze	de-neg-ge-ze	✓	✓
4	denbejan	den-be-jan	✓	✓
5	denn	denn	✓	✓
6	dengello	den-gel-lo	✓	✓
7	dhnnet	dix-hixn-net	✓	✓
8	dejjaf	dej-jaf	✓	✓
9	deggese	deg-ge-se	✓	✓
10	dggs	dixg-gixs	✓	✓
11	duro	du-ro	✓	✓
12	dammetxe	dam-me-txe	✓	✓
13	dassese	das-se-se	✓	✓
14	datenxnxa	da-tenx-nxa	✓	✓
15	danxnxa	danx-nxa	✓	✓
16	daggete	dag-ge-te	✓	✓
17	dllh	dixl-lixh	✓	✓
18	dnber	dixn-ber	✓	✓
19	dnkyya	dixn-kixy-ya	✓	✓
20	dngulla	dixn-gul-la	✓	✓
21	dkkula	dixk-ku-la	✓	✓
22	dww	dixww	✓	✓
23	ddd	dixdd	✓	✓
24	dgr	dix-gixr	✓	✓
25	jmmr	jixm-mixr	✓	✓
26	gehannem	ge-han-nem	✓	✓
27	begll	be-gixll	✓	✓
28	glffafii	gix-lixf-fa-fii	✓	✓
29	ixnglt	ixn-gixlt	✓	✓
30	gerreme	ger-re-me	✓	✓
31	gbbii	gixb-bii	✓	✓
32	gennet	gen-net	✓	✓
33	legenna	le-gen-na	✓	✓
34	sbket	sixb-ket	✓	✓
35	gzfet	gixz-fet	✓	✓
36	geddele	ged-de-le	✓	✓
37	gedlenxnxa	ged-lenx-nxa	✓	✓
38	gdaj	gix-daj	✓	✓
39	gegemmtenxnxa	ge-gem-mix-tenx-nxa	✓	✓
40	gffya	gixf-fix-ya	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	txellele	txel-le-le	✓	✓
2	metxtxeleya	metx-txe-le-ya	✓	✓
3	txelaqii	txe-la-qii	✓	✓
4	txelefa	txe-le-fa	✓	✓
5	txlflf	txixl-fixlf	✓	✓
6	txelat	txe-lat	✓	✓
7	txlacca	txix-lac-ca	✓	✓
8	tlenxnxa	tix-lenx-nxa	✓	✓
9	txella	txel-la	✓	✓
10	txemmeme	txem-me-me	✓	✓
11	txmqet	txixm-qet	✓	✓
12	txemezzeze	txe-mez-ze-ze	✓	✓
13	txmtxm	txixm-txixm	✓	✓
14	txeffere	txef-fe-re	✓	✓
15	txefertenxnxa	txe-fer-tenx-nxa	✓	✓
16	txefetxfew	txef-txix-few	✓	✓
17	txftx	txixftx	✓	✓
18	cxereqa	cxe-re-qa	✓	✓
19	txrztxrie	txixr-zixtx-rie	✓	✓
20	txnzizza	txixn-ziiz-za	✓	✓
21	txorenxnxa	txo-renx-nxa	✓	✓
22	cxelleme	cxel-le-me	✓	✓
23	cxelacx	cxe-lacx	✓	✓
24	cxmmaqii	cxixm-ma-qii	✓	✓
25	cxemeddede	cxe-med-de-de	✓	✓
26	acxcxerammede	acx-cxe-ram-me-de	✓	✓
27	cxercxassa	cxer-cxas-sa	✓	✓
28	cxeffarii	cxef-fa-rii	✓	✓
29	ascxeffarii	as-cxef-fa-rii	✓	✓
30	cxeffie	cxef-fie	✓	✓
31	cxlffiit	cxix-lixf-fiit	✓	✓
32	cxrrta	cxixr-rix-ta	✓	✓
33	cxorra	cxor-ra	✓	✓
34	cxorrere	cxor-re-re	✓	✓
35	cxnqqlat	cxix-nixq-qix-lat	✓	✓
36	cxra	cxix-ra	✓	✓
37	cxlffa	cxix-lixf-fa	✓	✓
38	cxqqnna	cxixq-qixn-na	✓	✓
39	pxaqumie	pxa-qu-mie	✓	✓
40	pxapxpxas	pxapx-pxas	✓	✓

SNO.	Transliterated word	Syllabification by the system	Remark on G2P	Remark on Syllabification
1	pxepxpxese	pxepx-pxe-se	✓	✓
2	axapxepxpxese	axa-pxepx-pxe-se	✓	✓
3	xxeleye	xxe-le-ye	✓	✓
4	xxelot	xxe-lot	✓	✓
5	xxelotenxnxa	xxe-lo-tenx-nxa	✓	✓
6	texxelleye	te-xxel-le-ye	✓	✓
7	xxenna	xxen-na	✓	✓
8	xxhfet	xxixh-fet	✓	✓
9	xxexxetenxnxa	xxe-xxe-tenx-nxa	✓	✓
10	xxehafii	xxe-ha-fii	✓	✓
11	xxyonawii	xxix-yo-na-wii	✓	✓
12	xxota	xxo-ta	✓	✓
13	flsfnna	fixl-six-fixn-na	✓	✓
14	flqlq	fixl-qixlq	Incorrect (missed epenthesis)	fixl-qix-lixq
15	fllagot	fixl-la-got	✓	✓
16	flfl	fixl-fixl	✓	✓
17	fecxcxe	fecx-cxe	✓	✓
18	fth	fix-tixh	✓	✓
19	foqqeqe	foq-qe-qe	✓	✓
20	fucxcxet	fucx-cxet	✓	✓
21	posta	pos-ta	✓	✓
22	postenxnxa	pos-tenx-nxa	✓	✓
23	fana	fa-na	✓	✓
24	affafame	af-fa-fa-me	✓	✓
25	fexxxxmo	fexx-mo	✓	✓
26	fegegta	fe-geg-ta	✓	✓
27	feyyede	fey-ye-de	✓	✓
28	feggege	feg-ge-ge	✓	✓
29	fejje	fej-je	✓	✓
30	fekka	fek-ka	✓	✓
31	fntxtxaqii	fix-nixtx-txa-qii	✓	✓
32	fentxezyya	fen-txe-zixy-ya	✓	✓
33	affenatxtxetxe	af-fe-natx-txe-txe	✓	✓
34	fenedda	fe-ned-da	✓	✓
35	wesfentxr	wes-fen-txixr	✓	✓
36	ftxnet	fixtx-net	✓	✓
37	wefcxo	wef-cxo	✓	✓
38	befxxxxum	bef-xxum	✓	✓
39	fetxtxene	fetx-txe-ne	✓	✓
40	tefexxeme	te-fe-xxe-me	✓	✓

Declaration

This thesis is my original work and has not been submitted as a partial requirement for a degree in any university.

Name: Nirayo Hailu Gebreegziabher

Signature: _____

Date: _____

Confirmed by advisor:

Name: Sebsbie Hailemariam (PhD)

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, June, 2011.