Challenges of Diacritical Marker or Hudhaa Character in Tokenization of Oromo Text

Abraham Tesso Nedjo
School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering,
Dalian University of Technology, Dalian, CHINA
Email: abratesso@gmail.com

Degen Huang and Xiaoxia Liu
School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering,
Dalian University of Technology, Dalian, CHINA
Email: huangdg@dlut.edu.cn, liuxxfm@gmail.com

***Abstract*— The problem of tokenization in natural language processing is to find a way to get every token in a text. For languages like Oromo, for which, much effort has not been done yet regarding language processing, the task of tokenization by no means cannot be overlooked. This paper reports on Oromo tokenizer that we designed and tested by accommodating the challenge of diacritical marker-Hudhaa which is a sign to represent in-word glottal sound in Oromo language. In this work, we have studied the effect of using acute accent for diacritical mark rather than using other confusing marks like right-quote to write Hudhaa. Accuracy is a prime factor in evaluating any Natural Language Processing (NLP) system. So we measured the accuracy of our system on 1.2MB (9686 sentences having 164932 words) of Oromo text data and an accuracy of 99.94% was achieved by this algorithm.**

***Index Terms*—Diacritical Marker; Glottal; Hudhaa; Oromo; Tokenization**

I. Introduction

OROMO language is classified as a Cushitic language of east African nations, spoken by more than 40 million people living in Ethiopia, Somalia, Kenya, and Djibouti and is the 3rd largest language in Africa. According to Omniglot: the online encyclopedia of writing systems and languages [1], Oromo people are the largest ethnic group in Ethiopia and account for more than 40% of the population. But according to Tilahun Gamta's article, present on [2], Oromo comprise 50%-60% of the population of the Ethiopian Empire State. The people are found all over Ethiopia and particularly in Wollega, Borana, Hararghe, Shoa, Illubabour, Wollo, Jimma, Bale, Arsi, and even in the southwestern zone of Gojjam.

The writing of Oromo language was with either the Ge'ez script or the Latin alphabet until the 1970s. Then some organizations began using the Latin alphabet as the official alphabet to write Oromo during the early 1970s, Oromo Liberation Front (OLF) was the one who chose first [1][2]. Soon later the language faced other political and social challenges. Because, under the Mengistu regime (between 1974 and 1991) the writing of Oromo language in any script had been banned officially, though limited usage of the Ge'ez script was allowed. After decades of struggling for formalizing the writing system of the language, , OLF convened a meeting of over 1,000 Oromo intellectuals to decide which alphabet to use to write Oromo, in the Parliament Building at Arat Kilo, Finfinnee, on 3rd of November 1991. After many hours of debate, they decided unanimously to adopt the Latin alphabet.

In Oromo language, there are glottal sound called Hudhaa with a controversial diacritical marker. For example, like in the words ba'anii, buáa, qeéetti, and deebi'anii. In different texts, one may find usage of different characters to represent Hudhaa, as in the example mentioned. The use of such inconsistent diacritical sign in computer technology made the Natural Language Processing (NLP) activities of Oromo text a bit troublesome. Especially the use of right-quote as diacritical marker has made the problem to be more severe, since right-quote is also used as a quote mark in Oromo.

In this paper, we present an effective Oromo tokenizer that enables people to get the basic unit of document – word. Our tokenizer perfectly considers how the glottal (Hudhaa character) that is hosted within the sequence of some Oromo words should be determined. Finally after having successfully designed the tokenizer, we concluded that there need to be standard use of diacritical marker-Hudhaa for writing Oromo language. Mainly, in order to avoid the use of right-quote or left-quote as glottal marker, we recommend acute accenting of vowels at glottal sound.

The result of this research is believed to have invaluable contribution in software development, particularly in the areas of natural language processing; like in machine translation (including tough works of translating unknown words [3] and in extracting

Manuscript received September 25, 2013; revised November 17, 2013; accepted November 22, 2013.
Corresponding author: abratesso@gmail.com

doi:10.4304/jsw.9.7.1818-1826

translation phrases [4]); parsing, and information retrieval, and any text related web applications including bilingual plagiarism detector [5], since tokenization is the basic step that software development of all these applications can not bypass. For the disadvantaged languages like Oromo and many others for which less effort or no effort has been done so far, regarding language applications, this result lays a good foundation, even for other languages to benchmark this good practice. For us, we are including the result of this paper in Oromo POS tagger and Oromo-English machine translation system (undergoing researches) as a module. And we will consider it for solving other NLP problem in the future and, of course, will work on its optimization as well.

The remainder of this paper is structured as follows. Section II briefly reviews the theoretical underpinnings of tokenization to give readers a picture about its importance and how it figures prominently in text processing. In Section III, we describe related works and framework that can leverage the task of tokenization. In Section IV we present an elucidation of the Oromo alphabet, writing system, Diacritical Marker (*Hudhaa*) and Oromo word formation referred to Orthographical and grammatical rules of the language. In Section V, we present our approach to the problem. And the experimental settings are presented in Sections VI. Finally, in Section VII and Section VIII, we discuss and conclude our paper and suggest avenues for future work.

II. TOKENIZATION

Tokenization is a very important task in text processing of any language. By tokenizing, we mean that we break up stream of texts into words, phrases, symbols, or other meaningful elements called tokens, which becomes input for further processing, such as parsing or text mining [6]. Tokenization is highly useful in many applications, particularly, both in linguistics (where it is a form of text segmentation), and in computer science, where create word level understanding and forms part of lexical analysis. Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation marks [7]. In addition to word segmentation, sentence segmentation which generally based on punctuation is a crucial first step in text processing. Every NLP task needs to do text normalization; like segmenting/tokenizing of words in running texts, normalizing word formats, or segmenting sentences in running texts [8].

Especially in Information Retrieval, as Christopher et al. [7] mentions, some ways to address the more difficult problems including development of more complex heuristics, querying a table of common special-cases, or fitting the tokens to a language model that identifies collocations in a later processing step, are paramount. More generally, for very long documents, the issue of indexing granularity arises. For a collection of books, it would usually be a bad idea to index an entire book as a document.

The major question of tokenization phase is, answering what are the correct tokens to use? Is it just a matter of chopping on whitespace and throw away punctuation characters? The answer for these questions is obviously no, since this is just a starting point and there are a number of tricky cases, even for English language, which seems relatively easy to tokenize [7]. For example, handling cases of various uses of the apostrophe for possession and contractions is always confusing.

Usually tokenization occurs at the word level. However, it is sometimes difficult to define what is meant by a "word". Often a tokenizer relies on simple heuristics, for example, like all contiguous strings of alphabetic characters and numbers are part of one token; tokens are separated by whitespace characters, such as a space or line break, or by punctuation characters; and punctuation and whitespace may or may not be included in the resulting list of tokens, and so on.

In languages that use inter-word spaces (such as most that use the Latin alphabet, like Oromo), this approach seems fairly straightforward. However, even here there are many edge cases such as contractions, hyphenated words, emoticons, and larger constructs such as URLs (which for some purposes may be counted as single tokens) [8]. Such problems are also there even in English. For example, in English "New York-based", a naive tokenizer may break at the space even though the better break is (arguably) at the hyphen. Particularly, in Oromo, where glottal character is used within words, splitting proper tokens as required is challenging. Therefore, close examination will make it clear that whitespace is not sufficient by itself for Oromo, even for English. Consider the following Oromo sentence from The Holy Bible:

Joh 14:6 Yesus deebisee, "Karichi, dhugaan, jireenyis, ana; anaan yoo ta'e malee, eenyu iyyuu gara abbaa hin dhufu." (Which means, Joh 14:6 Jesus saith unto him, I am the way, the truth, and the life: no man cometh unto the Father, but by me.)

Segmenting purely on whitespace would produce words like this: including the punctuations.

"Karichi, dhugaan, jireenyis, ana; dhufu."

These errors could be addressed by treating punctuations, quotations, and special characters, in addition to whitespace, as a word boundary. But punctuation often occurs word internally, which are used for short-hand writing like *A.I.B., A.B.O., O.F.C., 19/02/1977*, and other borrowed terms like *m.p.h., Ph.D., AT&T,* and *www.google.com*. Similarly, assuming that we want *35.5* to be a word, we will need to avoid segmenting every period, since that will wrongly segment this number into *35* and *5*. Number expression like use of comma (or space) in between every three digits is other complications to be thoroughly scrutinized and addressed in Oromo. For example, numbers like *9,999,888.55* or *9 999 888.55* or *9999888.55* (which are exactly the same) are common writing practices in many Oromo texts.

Moreover, Computer technology has introduced new types of character sequences that a tokenizer should probably tokenize as a single token, including email addresses and URLs like *(bilisummaa@mail.yahoo.com),*

and *(http://www. gadaa.com/index.html)*, numeric IP addresses *(142.32.48.231),* package tracking numbers *(1Z9999W99845399981*), and more.

As Jurafsky and Martin [8] mentions, a tokenizer can also be used to expand clitic contractions that are marked by apostrophe, like in English language, for example, converting *what're* to the two tokens *what* and *are*, *we're* to *we* and *are*, and *he's* to *he* and *is* or *he* and *has*. But this requires ambiguity resolution, since apostrophes are also used as genitive markers in English, for example, (as in *book's* or in *houses'*) or as quotative marks (as in *'what're you? Crazy?'*).

Of course, tokenization is particularly more difficult for languages with running text; such as Chinese which have no word boundaries, as mentioned by Huang et. al [9]. But Oromo language has no such problem of running text but the issue of compound words like creating names by joining two or more nouns is still challenging. Conceptually, splitting on white space can also split what should be regarded as a single token. This occurs most commonly with names *(Dirree Galaan, Abbaa Seenaa)* but also with borrowed foreign phrases *(Los Angeles)* and compounds that are sometimes written as a single word and sometimes space separated (such as *garaagara* vs. *garaa gara*). Other cases with internal spaces that we might wish to regard as a single token include phone numbers *(251) 911-658524)* and dates (*Ama 19, 1977*).

Therefore, tokenization algorithm may also tokenize multiword expressions of proper names like *Boolee Bulbulaa, Laga Danbii*, and *Gaara Mul'ataa* (name of places in Oromia), which requires multiword expression dictionary of some sort. This makes tokenization intimately tied up with natural language processing tasks of detecting names, dates, and organizations, a process called *named entity detection* [8].

III. Related Works

Word tokenization has been designed for many languages of the world pertinent to their specific orthographic and grammatical rules. But no such an effort has been made for Oromo language so far. Here we want to mention the word tokenization of English, since it has some similarity with Oromo, even though English language has few in-word glottal sign or diacritical marker. English has also apostrophe which somehow seems like the *Hudhaa* problem of Oromo. But the meaning and the way apostrophe is used and has been tokenized in English is quite different from the one we are presenting in this paper for Oromo. The English tokenizer works very good for English. But this English tokenizer does not work well for Oromo text as it would completely destroy the meaning of some of the words. Especially it highly affects words with *Hudhaa* or diacritical marker. To show the inefficiency of this existing English tokenizer for Oromo language, we looked at tokenizer of *split()* method of Python for comparison, as presented under our Discussion section (Section 7).

IV. Oromo Alphabet and Writing System

To understand a language, starting at its basic data unit (Alphabet or *Qubee* in Oromo) is mandatory. Oromo language shares a lot of features with English writing system except some differences. The Oromo writing system is a modification of Latin Writing system. The writing system of the language is straightforward which is designed based on the Latin script. All the letters in English language are also in Oromo but have different names and phonetic representations. Oromo has also *"Qubee Dachaa",* which is doubling of consonants from some other alphabets to represent different alphabet (*Qubee*) in Oromo. Wakshum [10] and Morka [11] describe Oromo writing system in more details.

Learning Oromo alphabet is very important because its structure is used in every day conversation. Without it, one will not be able to say words properly even if one knows how to write those words. The better you pronounce a letter in a word, the more you will understand in speaking the Oromo language. Specially, for someone who deals with text processing or NLP in general, understanding the characters is mandatory. Oromo alphabet has both vowels and consonants just like in English. Five of the letters *('a', 'e', 'i', 'o',* and *'u')* serve as vowels; whereas the rest of the alphabets including *"Qubee Dachaa"* are all consonants.

In Oromo language, the diacritic marker (*Hudhaa)* appears in between these vowels of words mostly, as these vowels are the one that have their own sounds and also give sound to the consonants in all syllable. In the subsequent sections, we will show how these *Hudhaa* is represented in words and the difficulties that it intrudes in differentiating from other quotation marks.

A. Oromo Punctuation Marks

Analysis of Oromo texts reveals that different Oromo punctuation marks follow the same punctuation pattern used in English and other languages that follow Latin Writing System [12]. The full stop (.) in statement, the question mark (?) in interrogative and the exclamation mark (!) in command and exclamatory sentences marks at the end of a sentence, and comma (,) which separates listing in a sentence and the semi colon (;) listing of ideas, concepts, names, items, etc. The other quotation marks are also used just like in English. The double quotations (" ") and single quotation ('') are used in similar rules as in English.

We might as well use the correct dashes where we need them in writing Oromo. There are three types of dash like that of English: the *hyphen*, the *endash*, and the *emdash*. Hyphens are typed as you'd hope, just by typing a - at the point in the word where you want a hyphen. Even though our typesetter or some conventional word processors takes care of hyphenation that is required to produce pretty linebreaks, we type a hyphen when we explicitly want one to appear, as in a combination like *"gar-tokko, irra-daddarbaa, qurxummii-qabduu, dhuga-baatuu",* and so on.

An *endash* is the correct dash to use in indicating number ranges, as in Lk.1—3 meaning "No.1–3". To

specify an endash one can type two consecutive dashes on the keyboard, as in 1--3. An emdash is a punctuation dash, used at the end of a sentence—some writers tend to use them too much, including us. To specify an emdash one can type three consecutive dashes on the keyboard, as in ". . . a sentence---we tend to. . . ". Moreover, ellipsis--also called points of suspension; which consists of three periods set close together is another punctuation used in the language. Ellipsis is often used to indicate an interruption or pause.

B. Diacritical Marker or Hudha in Oromo Language

The term Hudhaa is diacritical marker in Oromo. Just like in many languages of the world, in Oromo language there are places in words' syllable where it happens to have glottal sound. The main use of *Hudhaa* as diacritical marker in Oromo is to change the sound value of letter to which they are added. These show that the vowel with the diaeresis mark is pronounced separately from the preceding vowel; the acute and grave accents, which can indicate that a final vowel is to be pronounced differently.

Therefore, the definition of *Hudhaa* is the same as that of diacritical mark defined on English dictionary [13], which is a mark placed over, under or through a letter in some languages, to show that the letter should be pronounced differently from the same letter without the mark. Diacritical mark is also called diacritical point or diacritical sign. These marks are needed to accommodate sounds in a language with combination of those limited number of alphabet of the language. *Hudhaa*, serving as diacritical marker in Oromo, is found in many words of the language.

In computer technology, non-English languages seem to be unfavored, regarding *Hudhaa* (*diacritic*) type representation. Even though computer technology was developed mostly in the English-speaking countries, so data formats, keyboard layouts, etc. were developed with a bias favoring English (a language with an alphabet without diacritical marks), efforts have been made to handle them nowdays. Depending on the keyboard layout, which differs amongst countries, it is more or less easy to enter letters with diacritics on computers and typewriters. Some have their own keys; some are created by first pressing the key with the diacritic mark followed by the letter to place it on. Such a key is sometimes referred to as a dead key, as it produces no output of its own but modifies the output of the key pressed after it.

On computers, the availability of code pages determines whether one can use certain diacritics. Unicode solves this problem by assigning every known character its own code; By doing so, most modern computer systems provide a method to input it. With Unicode, it is also possible to combine diacritical marks with most characters. This Unicode representation is used in our program for accenting *Hudhaa* of Oromo language in this project.

Oromo is one of the African languages that have many words which contain diacritical marks – *Hudhaa*. Of course, the variety of the marker is not too many in Oromo, even though there are too many words. Moreover, a lot of unassimilated foreign loanwords are there in Oromo. The acute and grave accents are occasionally used.

C. Oromo Word Formation

Oromo word construction is straight forward and most convenient to represent any phoneme in the language and even to write foreign words. Words are formed from two or more sounds by combining vowels and consonants somehow like that of English. In Oromo words, all the consonants have no sound of their own unless a vowel is following them. But the vowels have their own sound as well as give sound to the consonants.

In Oromo, there are many writing grammatical rules, just like in any other language. But here we are going to mention the ones related to appearance of *'Hudhaa'* in some Oromo words. Besides the punctuations and quotations mentioned, there is another special character, usually apostrophe (') called "*Hudhaa*" for glottal sounding in some Oromo words. This *Hudhaa* in Oromo language appears between diphthongs (diphthong also known as a gliding vowel, which refers to two adjacent vowel sounds occurring within the same syllable) of Oromo word.

Mostly *Hudhaa* appears in between characters of words having consecutive vowels of different type (for example, *ta'e*, *ta'uu, bu'aa, ho'a, xaa'oo, mi'a, mi'aa, fe'uu, waa'ee*). So there is no instance in Oromo words having consecutive vowels of different type without *Hudhaa*. *Hudhaa* sometimes appears between characters of words having consecutive vowels of same type also, depending on the phoneme of the syllable. Especially when the number of the consecutive vowels is greater than two (for example, *re'ee, qe'ee, qu'uu*), though there are cases when it may rarely occur between just two of them (like in *sa'a).* Or *Hudhaa* may be inserted, between vowels and consonants, depending on the phoneme again (like in *har'a, hir'uu, mul'anne*).

Here our point is that the *Hudhaa* character selection found in many of Oromo texts and the difficulty of differentiating it from other quotations. Some people just use apostrophe (') or use of the right-quote character (') or left-quote character (') to indicate the glottal somewhere in the middle of words or as component of words. This leads to serious confusion in text processing as the single quote character represents another thing and meaning in text. Look at the confusion of the apostrophe in the following text:

> *Luk 14:17 Yeroon nyaataa yommuu ga'e, 'Amma kottaa hundinuu qophaa'eera' jedhee warra waamaman haa yaadachiisuuf hojjetaa isaa itti erge.*

As common practice, some people use symbols of vowels with hat from symbols dictionary (like *r□ee, bůaa, qůuu såa, xaåoo, etc).* One may find this kind of scriptures in Oromo literature, which is, of course, not a standardized way of writing the language. Because, these symbols (*å, □, ů, etc)* are not alphabetic characters and intrude another complication in tokenization of Oromo or in Oromo text processing activities in general.

Still in other texts written by use of LATEX or any TEX typesetter, *Hudhaa/glottal* in Oromo is represented by accenting of the vowels by control symbols. Of course,

LATEX typesetter can provide accents for just about all situations as needed. The accents in LATEX can be accessed through a variety of control symbols and single-letter control words which accept a single argument, like the letter to be accented [14]. For example, to write *ho'a,* some people use *ho\'{a}* in LATEX typesetter and when it is compiled, it becomes *ho'a.* Likewise, *bu\'{a}a* will produce *bu`aa*, *qe\'{e}e* will produce *qe`ee*, *waa\'{e}e* will produce *waa`ee*, and so on. But this is true only in the case of skilled writers of few publication domains (like academic journal articles and some books) and is not applicable for the vast majority of text processing package users. Of course, another way of using acute or grave accent in text processing applications (like in Microsoft Word) is using combination of dead key (**Ctrl + '** in our case) followed by the letter to be accented. Likewise, one can activate one's dead key on OfficeWriter by changing system keyboard layout to English US, International with dead keys. But, as observed in the corpus we used for this research purpose, these things are not much used, but few.

In some texts, the combinational use of these different writing schemes may exist inconsistently in the same document. At some point in the text, you may find the use of the right-quote character (') or left-quote (‘) or apostrophe (’), at another point you may find this glottal inserted from the symbol lists of the text processor, and so forth. And even writers who use LATEX accents for typesetting, sometimes insert the glottal (') manually and loose consistence.

So, all the writing schemes of *Hudhaa* in Oromo (use of apostrophe or right-quote within a word and use of special characters (vowels with hat), or using acute accent in LATEX) has got drawbacks of lacking uniformity. Therefore, there need be standard usage of such diacritical marker character selection in Oromo text (preferably ‘`’, which is different from apostrophe or right-quote character having a different Unicode value and easily available on any standard keyboard) in order to avoid such a confusion while word tokenization and other text processing activities. For example, writing *ta`e* instead of *ta'e* or *tåe*, *bu`aa* instead of *bu'aa*, or *buåa,* har*`a* instead of *har'a*, or *harå* etc. The issue is the question of standardization, but with computationally expensive process, it is possible to make double round replacement to reinstate the original apostrophe or right-quote character, indeed. According to our approach to the problem and solution we set, to be presented in the subsequent sections, the use of acute accent of the vowels is found to be the best way of representing *Hudhaa* in Oromo texts.

V. Our Approach

The problem of tokenization can be formally stated as follows. Given a streams of text *t* (including all its punctuations and quotes), we want to finds the words or tokens $t_1, t_2,\ldots,t_n$, discretely. Punctuation marks and other accompanying symbols and quotations are to be treated as separate tokens as well. In this Oromo language specific tokenization, our problem is dealing with the in-word glottal character *(Hudhaa)* handling, which wrongly separates and reports a single word into two or more tokens after tokenization. For this purpose, we proposed the algorithm shown in Fig. 1 and Fig. 2.

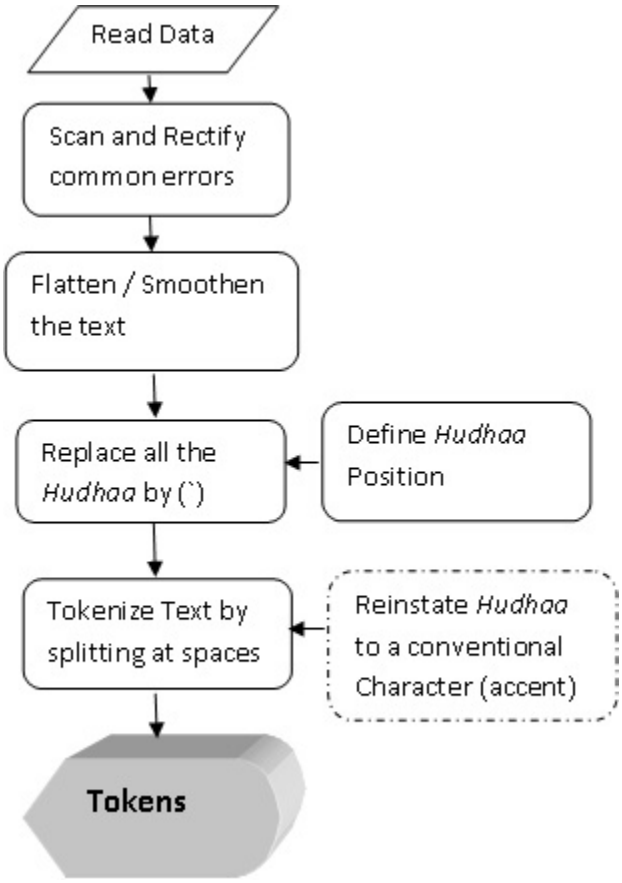


Figure 1: Algorithm of the Tokenizer with accenting the diacritical marker-Hudhaa

The algorithm begins by reading text data form file. Scanning through the text and modifying as required, in order to remove common typographical errors like missing of spaces after punctuation marks. (Eg *Haqa,Mirga,fi Diimokiraasii* should be written as *Haqa, Mirga, fi Diimokiraasii*). And then flattening or smoothen the text by removing unnecessarily included redundant whitespaces as well as replace fake whitespace characters with normal single whitespace.

The main challenge of this work lays at this point. Since the character for *Hudhaa*, as we discussed earlier may be: apostrophe, right-quote, acute accent, defining *Hudhaa* positions in the text, based on Oromo language orthographic rules is the main task of this endeavor. So scrutinizing which one of these are for *Hudhaa* representation and which of them are for their normal quotation purpose was challenging. For this purpose we depend on the behavior of *Hudhaa* in Oromo language that it is always surrounded by vowels, at least on one side, and that *Hudhaa* appears within a word (not at word boundary). After determining which ones of these characters are meant for *Hudhaa* purpose, we replaced them by *grave* (`), a character which is normally not used in writing Oromo language but easily available on

standard keyboards. This replacement has been needed as place holder until the subsequent tokenization activities like separation of other quotations from the text. And finally will be used for acute accenting the vowel following them.

```
function TOKENIZE-OROMO(file,punc,quotes, dub,
    hudhaa, apost, acute, grave) return token
 INITIALIZE (filename)
    f ← open(filename)
    while not EOF
      s ← readline()
 proc REPLACE-HUDHAA(s)
    for each(i,0→length(s))
      if s[i]? (apost, right-quote)
        CHECK ?(s[i-1], s[i+1], dub, Hudhaa_rule)
              S[i] ← grave
    return s
 proc CLEAR_EXTRA_SPACES(REPLACE-HUDHAA(), space)
    for each(i,0→length(s))
      if s[i]? AND s[i-1]? space
              REMOVE(s[i])
    return s
 proc TOKENIZE(EXTRA_SPACES(), punc, quotes, symbols)
    for each(i,0→length(s))
      if s[i]?( quotes,symbols)
              REMOVE s[i]
      else
              if s[i]?( punc)
              s ← ORTHO-RULE(s[i])
    return s
 proc MORE_SMOOTH(tokenize())
    for each(e,0→length(s))
      if s[e]?(space) AND s[e-1]? (space)
              REMOVE s[e]
      else:
              s ← ADD-TO-S (s[e])
    return s
 proc ACCENTIZE(MORE_SMOOTH(),dub, grave)
    for each(e,0→length(s))
      if s[e]?(grave) AND s[e+1]? (dub)
              REMOVE s[e]
              s[e+1]←ACUTE-ACCENT(s[e+1])
      elif s[e]?(grave) AND NOT s[e+1]? (dub)
              REMOVE s[e]
              s[e+1]←GRAVE-ACCENT(s[e-1])
      else:
              s ← ADD-TO-S (s[e])
    return s
 proc WORD_LIST(accentize())
    for each(e,0→length(s))
      if s[e]? (space)
        s[e]←NEWLINE
   token ←s
 return (token)
```

Figure 2: Pseudocode for the Algorithm of the Tokenizer

After that, we tokenized the text by considering different grammatical rules to remove or separate punctuation marks, quotes and symbols from words. Thereafter, performing one more smoothening before reinstating the *Hudhaa* (which is currently held by *grave*) is needed to clear out unnecessarily intruded extra whitespace. The final step of our tokenizer is *acute* accenting the diacritical marks of *Hudhaa* instead of the *grave* or instead of reinstating to one of those controversial marks like apostrophe (') character, right-quote (’) character or any other symbol.

VI. Experimental Results

The tokenizer prototype is tested and passed on the sample test data prepared for this purpose. We conducted the research on Oromo text of 1.2MB having 9686 sentences collected by making text mining. The corpus was collected from different public Oromo sites of newspapers and bulletins in order to make the sample corpus balanced and representative. The original data was full of terms, punctuation marks, quotes, and symbols. We fed the data to the algorithm designed and observed the result. After flattening the corpus, we have got 164932 words, among which, about 5.23% (8628 words have in-word glottal--*Hudhaa*, comprising 1674 unique words). Most of them are represented by right-quote character, some with English apostrophe ('), which is quite similar with the character used to close single quote. This is one of the difficulties encountered while striping off quotation marks from such accented words exclusively.

The tokenization we did without specially handling the *Hudhaa* character resulted in producing two or more wrong words by splitting at the syllable or diphthongs of words with *Hudhaa*. For that matter, in this research we made investigation on the difference between Oromo tokenizer in both scenarios for comparison.

Without taking out the *Hudhaa* before applying tokenization algorithm, we entirely loose the glottal representation of those words while casting out the quotes. But by carefully scrutinizing and locating the *Hudhaa* (based on Oromo Orthographic rules) we managed to retain the words with diphthongs by representing them with *grave* mark ‘`’ (some people call it accentor) before tokenizing the text. And this approach works inclusively very well with our tokenizer. That means the accuracy is more than 99% to handle the diacritical markers in Oromo, except some typographic errors found in the corpus. Table 1 and Table 2 show the accuracy of the prototype, both without and with accenting the *Hudhaa* respectively. We measured the accuracy by the ratio of the number of error of the tokenizer to the total number of words. The tokenizer prototype, in general, performs very well. We also repeatedly checked the performance of the algorithm with other small size texts and attested that it works even more effectively.

TABLE 1:
ACCURACY OF THE TOKENIZER WITHOUT ACCENTING HUDHAA

Total No of Words	Words with Hudhaa	Error of the Tokenizer without accenting Hudhaa	Effective Tokens without accenting Hudhaa	Accuracy
164932	8628	8716	156216	94.72 %

In addition to its effectiveness, we evaluated the performance of our tokenizer in terms of its efficiency based on space economy and response time. Table 3 shows our observation. Response time of the tokenizer by accenting the *Hudhaa* is a bit longer than that of without accenting (122 seconds and 92 seconds respectively for our 1.2MB data). But when we consider the space economy, accenting reduces the space requirement by as much as the number of words having *Hudhaa* in the text. In our case tokenization with accenting the *Hudhaa* is efficient by 8628 bytes.

TABLE 2:
ACCURACY OF THE TOKENIZER WITH ACCENTING HUDHAA

Total No of Words	Words with Hudhaa	Error of the Tokenizer without accenting Hudhaa	Effective Tokens without accenting Hudhaa	Accuracy
164932	8628	88	164844	99.94 %

TABLE 3:
EFFICIENCY OF THE TOKENIZER

	No of Characters	Response Time in Second
Tokenization without accenting *Hudhaa*	1002059	92
Tokenization with accenting *Hudhaa*	993431	122
Difference	8628	30

Moreover, we consulted linguistics in Oromo language to get scientific judgment for the work. The output of our tokenizer was seen by experts of the language and the result was judged to be in- line with the orthographic rules of Oromo and the tokens reported are correct.

VII. DISCUSSIONS

Here we begin our discussion by showing the inefficiency of the existing English tokenizer for Afan Oromo even if both languages show some similarities, in order to admire this endeavour and the result we obtained. For example, look at the following Oromo sentence *(s)* tokenized by English tokenizer of *split()* method of Python.

```
>>> s = "Manaa ba'anii buáa malee qeéetti
    deebi’anii hin galan!"
>>> ss= s.split()
>>> ss
```

['Manaa', "ba'anii", 'bu\xc3\xa1a', 'malee','qe\xc3\xa9etti' 'deebi\xe2\x80\x99anii','hin' 'galan!']

The tokens we get contain terms like 'bu\xc3\xa1a', ‘qe\xc3\xa9etti’ and 'deebi\xe2\x80\x99anii', depending on the diacritical marker we use for *Hudhaa*. These are actually the distorted form of Oromo words buáa, qeéetti, and deebi’anii according to the input string, though different diacritical marker is used inconsistently. And also the last word 'galan!' is mistakenly accompanied by exclamation mark ('!').

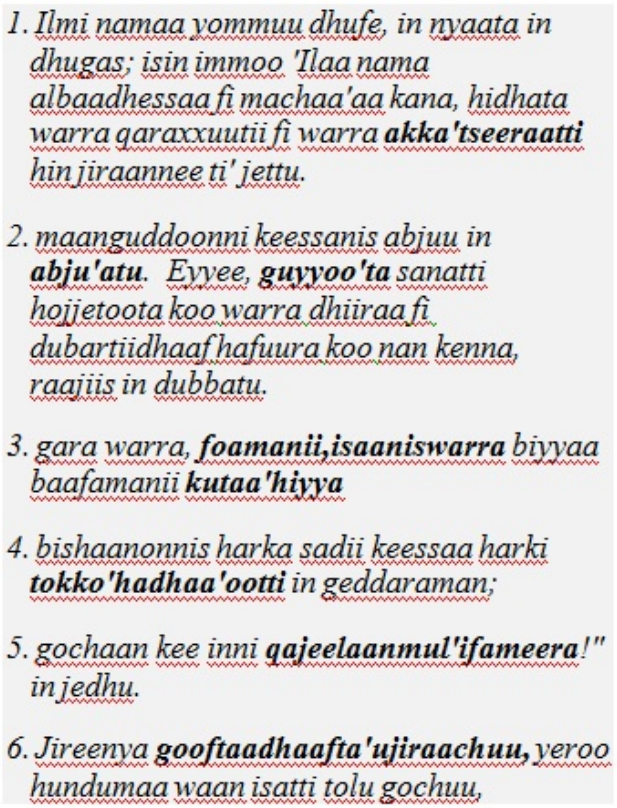
Furthermore, in order to compare the result and judge that one cannot depend on English tokenizer for Oromo text, as may be asked by some people, since both languages show some similarities, we tokenized the same corpus we used for our research purpose and obtained the result as in Table 4.

TABLE 4:
PERFORMANCE OF ENGLISH TOKENIZER FOR OROMO TEXT

Total No of Words	Error of the Tokenizer (in number of words)	Effective Tokens with English Tokenizer	Accuracy
164932	34055	130877	79.35%

So, one can generalize that this performance (less than 80% accuracy) is not credible enough to be used for Oromo text tokenization, at all.

In this project, all scenarios of Oromo grammatical and orthographical rules, are exhaustively investigated and deployed for Oromo tokenization. But the named entity detection and multiword expression dictionary was not considered except compound words linked with hyphen. And finally we reached at successful tokenizer for Oromo which can be adopted and used in any language processing activities of Oromo language.


1. *Ilmi namaa yommuu dhufe, in nyaata in dhugas; isin immoo 'Ilaa nama albaadhessaa fi machaa'aa kana, hidhata warra qaraxxuutii fi warra* ***akka'tseeraatti*** *hin jiraannee ti' jettu.*

2. *maanguddoonni keessanis abjuu in* ***abju'atu****. Eyyee,* ***guyyoo'ta*** *sanatti hojjetoota koo warra dhiiraa fi dubartiidhaaf hafuura koo nan kenna, raajiis in dubbatu.*

3. *gara warra,* ***foamanii,isaaniswarra*** *biyyaa baafamanii* ***kutaa'hiyya***

4. *bishaanonnis harka sadii keessaa harki* ***tokko'hadhaa'ootti*** *in geddaraman;*

5. *gochaan kee inni* ***qajeelaanmul'ifameera****!" in jedhu.*

6. *Jireenya* ***gooftaadhaafta'ujiraachuu,*** *yeroo hundumaa waan isatti tolu gochuu,*

Figure 3: Some Error sources seen in the corpus

The most challenging and critical problems encountered in this endeavor was the typographic errors of the corpus. Particularly, the errors around the punctuations and quotation marks were troublesome. For example, look at the numbered sentences, as in Fig. 3 taken from the corpus. The term *akka'tseeraatti* in the first sentence, which is actually not a word, but is unintentionally, linked two words. It happened so, because typist of the verse forgot pressing space bar between the word *akka* and the word *'tseeraatti* with left-quote. So the apostrophe which appeared as *Hudhaa* of the preceding vowel *‘a’*, is not glottal. The word *tseeraatti* is even not a correct Oromo word, for that matter. The first letter of the word *‘t’* is an error. The intention was to write *akka seeraatti (with space, meaning as rule of law),* gives meaning together with the preceding word *akka.* In the second sentence, the word ***abju'atu*** and ***guyyoo'ta*** are totally spelling error to write ***abjootu*** and ***guyyoota***.

Similarly, in the third sentence, the term ***foamanii,isaaniswarra*** has three errors. The second and the third characters (*o* and *a*) of the term are vowels of two different type which cannot be written consecutively without *Hudhaa*. The second problem with this error is that there is no space after the comma, and hence our tokenizer considers the entire term as one word by just knocking out the comma. Which is of course not an Oromo word appearing together. Even the term after the comma ***isaaniswarra*** is an error. Since the two words ***(isaanis warra)*** was the real intention of the phrase, to write ***fo'amanii, isaanis warra****(meaning, chosen, they are those)*. The term ***kutaa'hiyya*** happening with such confusing *Hudhaa* is also another error in the same sentence. The true words of the Bible verse were written as ***kutaa biyya***. In the forth sentence we see the term ***tokko'hadhaa'ootti***, which is an error again. A space is needed between the word ***tokko*** and ***hadhaa'ootti***. In addition, the left-quote *(')* preceding the first character *(h)* of the term ***'hadhaa'ootti*** is intruded by mistake. The correct spelling of the term ***qajeelaanmul'ifameera*** is ***qajeelaan mul'ifameera*** with space between the two words *qajeelaan* and *mul'ifameera.*

Too many similar cases are there in punctuation marks as well. Particularly, missing of spaces after comma and period are the more critical ones. And also there are full of unnecessary inclusion of spaces after periods while writing acronyms, which our algorithm judges as end of sentence. These things have somehow affected the result of our algorithm we designed for Oromo tokenizer.

VIII. Conclusions and Future Work

In this paper, we have shown that there is high degree of inconsistent use of diacritical marker in Oromo language among writers. Though our system accommodates all, for the tokenization purpose, we pose an issue of standardizing *Hudhaa* representation in computer technology that Oromo language linguistics should further investigate. We recommend to opt for acute accent on the vowels to show diacritical mark. One can make accenting in conventional text processing applications represent *Hudhaa* by using dead key (like ***Ctrl +'***) followed by the vowel to be accented.

Finally, we conclude our paper by recommending researchers and practitioners in the area of computational linguistics as well as Oromo language studies to use the result of this work and explore more in the future. Moreover, we advise software developers working in the area of natural language processing to include this tokenizer since it is basic and very important for the other subsequent processes. For us, we are including the result of this paper in Oromo POS tagger and Oromo-English machine translation system (undergoing researches) as a module. And we will consider it for solving other NLP problem in the future and, of course, will work on its optimization as well.

Acknowledgment

This work has been supported by the National Natural Science Foundation of China. Project No (61173100, 61173101, and 61272375).

References

[1] http://www.omniglot.com/writing/oromo.htm. Omniglot: The Online Encyclopedia of Writing Systems & Languages, Available Online, May 2013.
[2] http://www.africa.upenn.edu/Hornet/Afaan_Oromo_19777.html. *Afaan Oromo*, Available Online, May 2013
[3] K. Md. Anwarus Salam, S. Yamada, and T. Nishino. (2013). *How to Translate Unknown Words for English to Bangla Machine Translation Using Transliteration*, Journal of Computers, Vol. 8, no. 5, May 2013. doi:10.4304/jcp.8.5.1167-1174
[4] Z. Chun-Xiang, M. Y. Ren, Z. M. Lu, Y. H. Liang, D. S. Sun, Y. Liu (2011). *Multiple Linear Regression for Extracting Phrase Translation Pairs,* Journal of Computers, Vol. 6, NO. 5, MAY 2011. doi:10.4304/jcp.6.5.905-912
[5] M. Shamsul Arefin, Y. Morimotoy, M. A. Sharifz. (2013) BAENPD: A Bilingual Plagiarism Detector, Journal of Computers, Vol. 8, no. 5, May 2013. doi:10.4304/jcp.8.5.1145-1156
[6] *http://en.wikipedia.org/wiki/Tokenization* Wikipedia, The Free Encyclopedia, Available Online, May 2013
[7] C. D. Manning, P. Raghavan, and H. Schütze, *An Introduction to Information Retrieval*, Cambridge University Press, (2008).
[8] D. Jurafsky, J. Martin, *Speach and Language Processign:* An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, (2010).
[9] C. Huang, P. Simon, S. Hsieh, and L. Prevot, *Rethinking Chinese Word Segmentation:* Tokenization, Character Classification, or Word break Identification, (2007)
[10] W. Mekonnen *Development of Stemming Algorithm for Oromo Texts*, Master's Thesis, Addis Ababa University, (2000).
[11] M. Mekonnen *Text to Speech System for Afaan Oromo*, Master's Thesis, Addis Ababa University, (2001).
[12] D. Megersa *Automatic Sentence Parser for Oromo Language Using Supervised Learning Technique*, Master's Thesis, Addis Ababa University, (2002).
[13] OXFORD *Advanced Learners' Dictionary*, Oxford University Press 2000
[14] G. Maltby (1992). An Introduction to TEX and friends. http://heather.cs.ucdavis.edu/~matloff/LaTeX/Malthy.pdf, Available Online, May 2013

Abraham Tesso Nedjo, is a PhD Candidate in Computer Science at School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning Province, CHINA. His current research interest focuses machine translation, natural language processing, and machine learning. He can be reached at: abratesso@gmail.com

Degen Huang (Prof), is a Professor of Computer Science at School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning Province, CHINA. His major research interests include machine translation, information retrieval and natural language processing. He can be reached at: huangdg@dlut.edu.cn

Xiaoxia Liu is a PhD Candidate in Computer Science at School of Computer Science and Technology, Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning Province, CHINA. Her current research interest focuses machine translation, natural language processing, and machine learning. She can be reached at: liuxxfm@gmail.com