Design Issues in Automatic Grapheme-to-Phoneme Conversion for Standard *Yorùbá*

Abímbọ́lá R. Ìyàndá and Ọdẹ́túnjí A. Ọdéjọbí

Obafemi Awolowo University
Computer Science and Engineering Department
Ile-Ife, Nigeria
abiyanda@oauife.edu.ng, oodejobi@oauife.edu.ng

Abstract. Grapheme-to-Phoneme (G2P) conversion is an important problem in Human Language Processing development, particularly Text-to-Speech (TTS). Its primary goal is to accurately compute the pronunciation of words in the input texts. This work examines design issues with respect to components of the automatic G2P for standard *Yorùbá* (SY). The automatic process includes: (i) Tokenisation of Input, (ii) Identification of nasal characters, (iii) Syllabification, and (iv) Conversion of Graphemes to Phonemes. The structure of the Yoruba text is described and a text corpus design for standard *Yorùbá* TTS is presented. The analysis of the data was done using Zipf's law. The outcome of this work provided adequate requirement for the design.

1 Introduction

Grapheme-to-phoneme conversion (G2P) refers to the task of finding the pronunciation of a word given its written form. It has important applications in human language technologies

Alphabetic writing systems are based on the idea that the orthographic form is a conventional representation of a word's pronunciation [1]. In a perfectly phonological alphabet such as existed in *Yorùbá* language, there is a one-to-one correspondence between letters (graphemes) and phonemes.

The Grapheme-to-Phoneme (G2P) conversion subsystem is a crucial component of a TTS particularly for a tone language. TTS is an important human language technology which requires a good G2P conversion process. It is impossible to have a good-working text-to-speech system without having a tool that generates correct pronunciation when presented with the word orthography.

2 Standard Yoruba language

Yorùbá language is one of the three major indigenous languages, along with Hausa and Igbo in Nigeria and it is spoken by over 37 million people [2]. *Yorùbá* is a native language of the *Yorùbá* people, an ethnic group primarily located in

south-western Nigeria (Lagos, *Ọ̀yọ́, Ògùn, Onàdó, Èkìtì, Ọ̀ṣun* and parts of *Kwara* and *Kogí* states). SY is used in language education, the mass media and everyday communication [3]. *Yorùbá* is a tone language in which a pitch of an utterance is used to express differences in meaning; or when a particular pitch or change of pitch constitute an element in the intonation of a phrase or sentence, such as high, mid or low. The tone structure in *Yorùbá* is made possible by diacritics - tone marking on top of vowels and syllabic nasals to give meaning to the contexts.

2.1 Description of *Yorùbá* Text

Yorùbá alphabet comprises of 18 consonants (represented graphemically by *b, d, f, g, gb, h, j, k, l, m, n, p, r, s, ṣ, t, w, y*) and 7 vowels represented graphemically by *a, e, ẹ, i, o, ọ, u* [4]. It should be noted that the *gb* is a diagraph i.e. a consonant written with two letters. Five nasal vowels exist in the language by adding *n* with the oral vowels (*a, e, ẹ, i, o, ọ, u*) which is represented graphemically as *an, ẹn, in, ọn, un.* Also, one syllabic nasals represented graphemically as *n* exists. It should also be noted that *an* and *ọn* are phonemically the same. The consonant and vowel systems as well as a more detailed description of SY are presented in [5].

In SY language, syllables are considered as the basic unit of sound because it has been established as a perceptually and acoustically coherent unit [4]. In addition, syllable can be considered as the basic unit of G2P in tone language as agreed with the work of [6], [7] and [8] established that syllables produce reasonably natural quality speech. The letters are combined to form syllable based on the syllable structures and syllables combined to form words. Words are combined to form statements. There exist five syllables structures in the language and these are: V (oral vowels), Vn (nasal vowels), N (syllabic nasal), CV (combination of consonants and oral vowels) as well as CVn (combination of consonants and nasal vowels) [3]

The syllables are the tone bearing elements of the SY language. For example, in the statement: *Abímbọ́lá ti lọ sí oko* (*Abímbọ́lá* has gone to the farm), there are ten syllables. This also implies that there exist ten tones (MHMHH-M-M-H-MM).

3 Data

The domain for the speech synthesis for this research is in language education, religious and mass media from various sources such as Internet, digitized printed material and existing digital materials from non-Internet sources. Two SY newspaper (Aláròyé and *Yorùbá* Gbòde) and two SY textbooks ([9], [10]) were selected. The two newspapers were not toned mark, and *Yorùbá* Gbòde is without under dots. The texts were edited using Tákàdá text editor (www.sourceforge.net/projects/takada) and were corrected for graphemic items (tone marks and under dots) using Àkọtọ́ *Yorùbá* [11]. Sample data is as shown in Table 1. To increase

the quality and coverage of the acquired materials, additional *Yorùbá* text data were gathered from existing printed materials by scanning through a process known as optical character recognition. The volume of textual data generated using these techniques was about 291,392 words.

Table 1. Sample Data

Yoruba Statement	Meaning
Ìbùkún ni fún ọkùnrin náà tí kò rìn ní ìmọ̀ àwọn ènìyàn búburú, tí kò dúró ní ọ̀nà àwọn ẹlẹ́ṣẹ̀, àti tí kò sì jókòó ní ibùjókòó àwọn ẹlẹ́gàn. Ṣùgbọ́n dídùn inúu rẹ̀ wà nínú òfin Olúwa, àti nínú òfin náà ni ó ńṣe àsàrò ní ọ̀sán àti ní òru. Yóò sì dàbí igi tí a gbìn sí etí ipa odò, tí ó ń so èso rẹ̀ jáde ní àkókò rẹ̀, ewée rẹ̀ kì yóò sì rẹ̀, àti ohunkóhun tí ó bá ńṣe ni yóò máa ṣe déédé. Nítorí Olúwa mọ ọ̀nà àwọn olódodo, ṣùgbọ́n ọ̀nà àwọn ènìyàn búburú ni yóò ṣègbé.	Blessed is the man that walks not in the counsel of the ungodly, nor stands in the way of sinners, nor sits in the seat of the scornful, but his delight is in the law of the LORD; and in his law he does meditates day and night. He shall be like a tree planted by the rivers of water, that brings forth its fruit in its season; its leaf also shall not wither; and whatsoever it does shall prosper. For the LORD knows the way of the righteous: but the way of the ungodly shall perish.

Data analysis was done using Zipf's law, which observed a phenomenon in human languages (and some other social structures) that indicated that there is a pattern in the distribution of tokens and that a few very frequent words make up a very large portion of any text or collection of texts, while the large majority of words occur relatively rarely. Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc [12].

The analysis of the research data for frequency distribution was performed using a freeware concordance; Simple Text Analysis Tool (TextSTAT) version 2.9.0.0 [13]. This tool was used to generate the frequency count for the word tokens in the text data. It was also used to detect words that have incorrect forms such as wrong diacritics (tone marks and under dots). In the analysis of the words in the text corpus, 6857 words appeared only once in the corpus and the word with the highest appearance in the data set *(tí)* occur 8257 times, followed by *ni* with frequency of 6358 and followed by *àwon* with frequency of 5753. This pattern is expected of natural language [12]. Table 2 shows the distribution of some selected words in conformity with Zipf's law for linguistic data.

Zipf's law states that there is an inverse proportional relationship between the rank and frequency of words in a text. Rank is the position of a word in

Table 2. Ranks and frequencies of the text data

Word type	Rank (r)	Frequency (f)	Proportion (%)
n	10	4365	1.4978
ti	20	2673	0.9173
fún (give)	30	2215	0.7601
gẹ (cut)	40	1560	0.5354
èdè (language)	50	1153	0.3957
o (you)	60	947	0.3250
ìṣe (doing)	70	786	0.2697
ìtàn (story)	80	694	0.2382
pọ	90	599	0.2056
ìmọ̀ (knowledge)	100	524	0.1798
ìpínlẹ̀ (state)	200	189	0.0649
àtijọ́ (former)	300	110	0.0377
yí (turn)	400	77	0.0264
ọjà (market)	500	57	0.0196
Adéyẹmí (*Yorùbá* name)	600	44	0.0151
àǎyè (opportunity)	700	37	0.0127
àgbàlagbà (adult)	800	30	0.0103
gbeyẹ̀wò (consider)	900	25	0.0086
déésì (translated name)	1000	21	0.0072
kórìíra (to hate)	10000	1	0.0003

a table of words ordered by frequency of occurrence (rank one being the most frequent) [12].

The general nature of Zipf's distribution as it applies to human language use whether in oral or written communication or discourse is as highlighted below [11]:

i. a few word tends to score very high on the frequency table meaning that they appear regularly in discourse;
ii. a medium number of words or linguistic tokens appears with relatively medium frequency indicating that they are regularly used;
iii. a huge number of words in a document have very low level of occurrence, indicating that they are rarely used.

With the proportion of these tokens in relation to the total token count, it means that these words occurred in many contexts and are good features to be used in grapheme to phoneme conversion system for *Yorùbá* language. The Zipf's curve for the words whose rank ranges from tens to thousands (the regularly occur to the rarely occur), to have a wide coverage of the database is shown in Figure 1. The shape of the curve justifies Zipf's law for linguistic data. This means that a few word appear regularly in discourse; a medium number of words or linguistic tokens appears with relatively medium frequency indicating that they are averagely used; a huge number of words in a document have very low level of occurrence, indicating that they are rarely used.

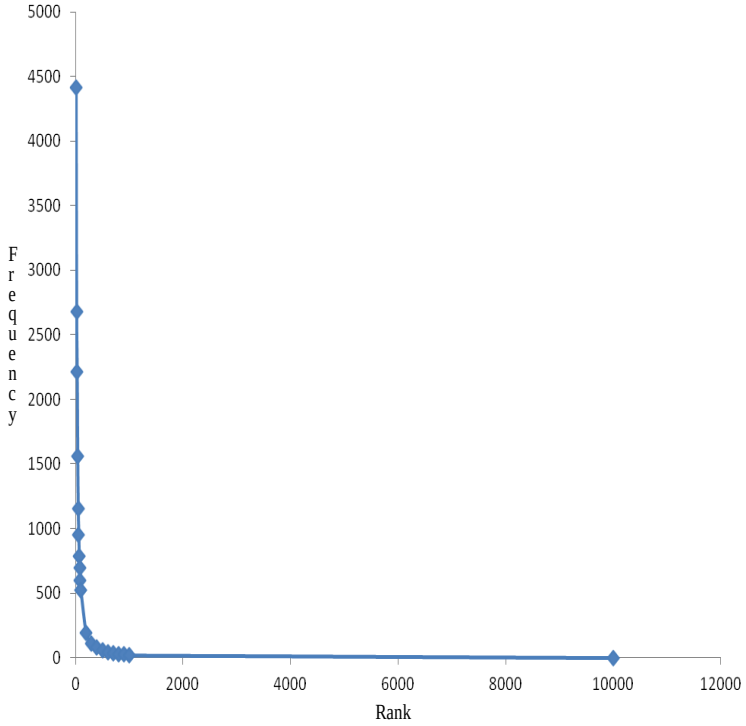

Fig. 1. Zipf frequency-rank curve for *Yorùbá* texts

4 Processes in G2P Conversion

The processes involved in the G2P conversion for SY is shown in Figure 2. These processes include: (i) Input tokenisation process which splits the input sentence into individual characters, (ii) Nasal characters identification process which detects nasalised vowels, (iii) Syllabification process which accepts input tokens and produces corresponding syllables, and (iv) Graphemes to Phonemes conversion process. To facilitate this process of converting grapheme to phoneme, the characters with underdots and the diagraph- *ẹ̀, ẹ́, ọ̀, ọ́* and *gb* were transformed to 'z', 'x', 'c', 'v', and 'q' respectively.

4.1 Input Tokenisation

The input to this process is a text file. Tokenisation takes the input text and breaks it into sentences marked by a new-line, and each sentence is further breakdown into chunks called tokens. Tokenisation was done using white spaces as delimiters. This process was achieved using the *split()* function of the String class. This function takes a string and separates them using white space. The

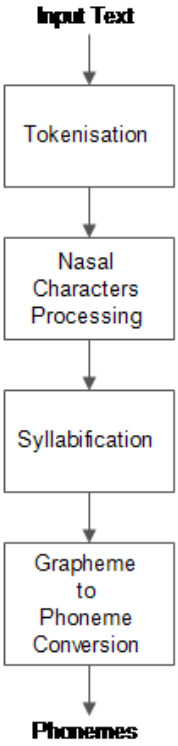

Fig. 2. G2P Processes

output of the *split()* function is a List object containing all the tokens. Each token is separated by a front slash (/).

Figure 3 shows the output for Software processing of tokenisation.

4.2 Process Nasal Characters

This stage processes nasal characters. For example: to convert *'a'* and *'n'* to *'an'*; *'ẹ'* and *'n'* to *'ẹn'*; *'i'* and *'n'* to *'in'*; *'ọ'* and *'n'* to *'ọn'*; and *'u'* and *'n'* to *'un'* where appropriate.

It is should be noted that any of the consonant can precede nasal vowel except l,m and n. E.g.*imún*, *inán* as well as *lan, lẹn, lọn, lun* does not exist orthographically in SY words. The process is achieved by the following lines of code.

```
def processNasals(self, sym=[]):
        ind = [i for i, x in enumerate(sym) if x == 'n']
        for i in ind:
            if i != 0:
```

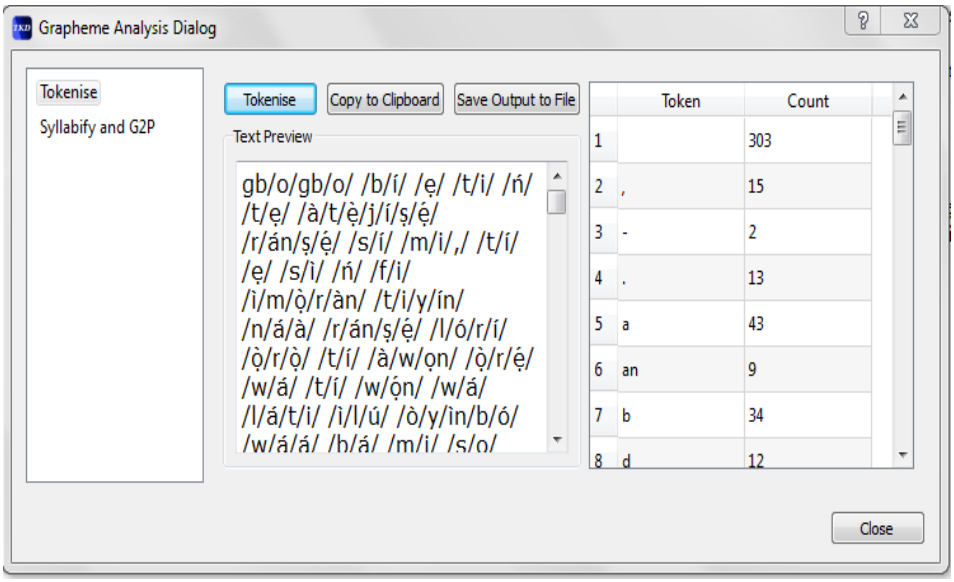

Fig. 3. Screen shot for sentence tokenised

```
            vn = sym[i - 1] + 'n'
            if vn in self.Vn:
                if i + 1 < len(sym):
                    if sym[i + 1] not in self.V:
                        sym[i - 1] = vn
                        sym[i] = u'@'
                else:
                    sym[i - 1] = vn
                    sym[i] = u'@'

    # remove all x's
    for i, x in enumerate(sym):
        if x == '@':
            sym.remove('@')
    return sym
```

4.3 Syllabification Process

The syllable corpus was created from the word corpus using a process known as syllabification. The syllabification was achieved using a Push Down Transducer (PDT) [14]. The PDT employed for the syllabification process extends the functionality of PDA (generalised NFA) beyond language recognition by ensuring an output on every transition. This is necessary because we are not interested

in verifying the acceptance of the input, but rather to obtain *Yorùbá* syllables from a continuous stream of texts. As a result, the PDT developed is a 8-tuple defined as:

$$PDT =< Q, \Sigma_i, \Sigma_o, \Gamma, q_0, Z_0, F, \delta > \tag{1}$$

Where:

- Q is a finite set of states, and
  $Q = \{Q_0, Q_1, Q_2\}$
- Σ_i is a finite set of input symbols (tokens), and
  $\Sigma_i = V \cup V_n \cup C \cup N$, where
  V is a set of vowels, i.e.,
  $V = \{a,\ e,\ ẹ,\ i,\ o,\ ọ,\ u,\ à,\ á,\ è,\ é,\ ẹ̀,\ ẹ́,\ ì,\ í,\ ò,\ ó,\ ọ̀,\ ọ́,\ ù,\ ú\ \}$
  V_n is a set of nasalised vowels, i.e.,
  $V_n = \{an,\ ẹn,\ in,\ ọn,\ un,\ àn,\ án,\ ẹ̀n,\ ẹ́n,\ ìn,\ ín,\ ọ̀n,\ ọ́n,\ ùn,\ ún\ \}$
  C is a set of consonants, i.e., $C = \{b,\ d,\ f,\ g,\ gb,\ h,\ j,\ k,\ l,\ p,\ r,\ s,\ ṣ,\ t,\ w,\ y\}$
  N is a set of syllabic nasals, i.e., $N = \{m, n, ń, ǹ, m̀\}$
- Σ_o is a finite set of output syllable symbols. This represents the acceptable sequence of characters that make up a syllable. As such,
  $\Sigma_o = V, v_n, CV, CV_n, N$
- Γ is a finite set of stack symbols, i.e., $\Gamma = C \cup N \cup \{\lambda\}$
- $q_0 \in Q$ is an initial state, and $q_0 = Q_0$
- $Z_0 \in \Gamma$ is the initial stack symbol, and $Z_0 = \lambda$
- $F \subseteq Q$ is a set of final states, and $F = Q_3$
- δ is a transition function given as:

$$Q \times \Sigma_i \times (\Gamma \cup \epsilon) \rightarrow 2^{Q \times \Gamma \times \Sigma_o} \tag{2}$$

A transition of a PDT could be represented using an element $(p, a, A, q, B, b) \in \delta$ which is represented as shown in Figure 4 is interpreted as: In a state $p \in Q$, given an input symbol $a \in (\Sigma_i \cup \{\epsilon\})$, and $A \in \Gamma$ being the topmost stack symbol, move to state $q \in Q$, pop $A \in \Gamma$ and push $B \in \Gamma$ onto the stack , and produce an output symbol $b \in (\Sigma_o \cup \{\epsilon\})$.

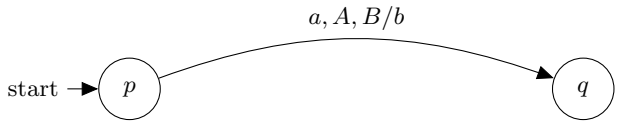

Fig. 4. A transition $(p, a, A, q, B, b) \in \delta$ of a PDT

As found in PDAs, PDTs have two modes of acceptance which are: acceptance by final state and acceptance by empty stack. We employed the two modes of acceptance in the modified PDA such that the PDA accepts the sequence of symbols w if there is a sequence of transitions from $(q_0, a_i, \lambda, q_i, B_i, b)$ and terminates at $(q_i, \#, \lambda, \epsilon, q_F \in F, \epsilon, \epsilon)$. This means that for a sequence of input

Algorithm 1: Algorithm for the Nasal Vowel Identification and Syllabification Processes.

```
1  V = {a, e, ẹ, i, o, ọ, u, á, à, é, è, ẹ́, ẹ̀, í, ì, ó, ò, ọ́, ọ̀, ú, ù } ;
2  C = {b, d, f, g, gb, h, j, k, l, p, r, s, ṣ, t, w, y} ;
3  V_n = {an, ẹn, in, ọn, un, àn, án, ẹ̀n, ẹ́n, ìn, ín, ọ̀n, ọ́n, ùn, ún } ;
4  N = {m, n, ń, ǹ, m̀} ;
   Data: tokenList: A List of Input Tokens
   Result: syllables: A List of Syllables
5  procedure syllabificator (tokenList)
6      for token in enumerate(tokens) do
7          syllables = ";
8          stack = new Stack();
9          if token ∈ (V ∪ Vn) then
10             if size(stack) == 0 then
11                 syllables =+ token;
12             else
13                 top = stack.pop();
14                 if top ∈ (self.C ∪ self.N) then
15                     syllables =+ top + token;
16                 end if
17             end if
18         else if token ∈ (self.C ∪ self.N) then
19             if size(stack) == 0 then
20                 stack.push(token);
21             else
22                 top = stack.pop();
23                 if top ∈ self.N then
24                     stack.push(token);
25                     syllables =+ top;
26                 end if
27             end if
28         else
29             syllables =+ token;
30         end if
31         return syllables ;
32     end
33 end procedure
```

symbols to be accepted the PDT must be in a final state, the stack must be empty, and all the input symbols must be read. The PDT was implemented using the algorithm presented in Algorithm 1.

The PDT was simulated in JFLAP using the following input tokens:

tokens = [‘ǹ’, ‘ǹ’, ‘k’, ‘an’, ‘ ’, ‘ḿ’, ‘b’, ‘ẹ’, ‘ ’, ‘l’, ‘á’, ‘b’, ‘ẹ́”, ‘ ’, ‘ọ̀’, ‘ọ̀’, ‘r’, ‘un]

The transition table for simulation of the syllabification process using these input tokens is shown in Table 3, and the output of the simulation is *'n-n-kan m-be la-be o-o-run'*

Table 3. Syllabification Process for *'ǹǹkan ḿbẹ́ lábẹ́ ọ̀ọ̀run#'*

Token	Current State	Current Stack	Pop	Push	Resulting Stack	Output	Next State
'ǹ'	Q_0	[λ]	λ	ǹ, λ	[ǹ, λ]	ϵ	Q_2
'ǹ'	Q_2	[ǹ, λ]	ǹ	ǹ	[ǹ, λ]	ǹ	Q_2
'k'	Q_2	[ǹ, λ]	ǹ	k	[k, λ]	ǹ	Q_2
'an'	Q_2	[k, λ]	k	ϵ	[λ]	kan	Q_1
' '	Q_1	[λ]	λ	λ	[λ]	' '	Q_1
'ḿ'	Q_1	[λ]	λ	ḿ,λ	[m, λ]	ϵ	Q_2
'b'	Q_2	[m, λ]	ḿ	b	[b, λ]	ḿ	Q_2
'ẹ'	Q_2	[b, λ]	b	ϵ	[λ]	bẹ	Q_1
' '	Q_1	[λ]	λ	λ	[λ]	' '	Q_1
l	Q_1	[λ]	λ	l, λ	[l, λ]	ϵ	Q_2
á	Q_2	[l, λ]	l	ϵ	[λ]	lá	Q_1
b	Q_1	[λ]	λ	b, λ	[b, λ]	ϵ	Q_2
ẹ́	Q_2	[b, λ]	b	ϵ	[λ]	bẹ́	Q_1
' '	Q_1	[λ]	λ	λ	[λ]	' '	Q_1
ọ̀	Q_1	[λ]	λ	λ	[λ]	ọ̀	Q_1
ọ̀	Q_1	[λ]	λ	λ	[λ]	ọ̀	Q_1
r	Q_1	[λ]	λ	a, λ	[r, λ]	ϵ	Q_2
un	Q_2	[r, λ]	r	ϵ	[λ]	run	Q_1
#	Q_2	[λ]	λ	ϵ	[]	ϵ	$Q_3 \in F$

4.4 Graphemes to Phonemes conversion

This stage converts all the graphemes resulting from the previous stage into phonemes. Different methods for building automatic alignments have been proposed, among which there are one-to-one and many-to-many alignments [1]. In this study one-to-one alignment was adopted because each letter corresponds only to one phoneme and vice versa. This is achieved by first replacing all transformed characters back to their original characters. For example, grapheme ẹ which was formally replaced with 'z' was converted back to ẹ before the phonemic transcription output was given. Syllables with high, mid and low tones are transcribed with their tones (H, M, L, respectively) placed in bracket beside them.

5 Conclusion

The Grapheme-to-Phoneme (G2P) conversion subsystem is a crucial component of a TTS particularly for a tone language. Attempts have been made to develop

other components of Standard *Yorùbá* TTS, but the G2P subsystem is yet to be addressed.

This work opens the way for synthetic speech to approach the quality of natural speech. In this paper we outlined the fundamentals of this concept and gave a detailed progress report on our ongoing research. We gave an account of the data analysis performed and the process underlying SY text to sound was examined. Some other design issues were as well discussed . This work provides adequate requirement for the design and as well provides the basis for research and development in TTS for SY language.

References

1. Bisani, M., Ney, H.: Joint-Sequence Models for Grapheme-to-Phoneme Conversion. Speech Communication **50** (2008) 434–451
2. CIA: Central intelligence agency world factbook 2014. Available online at http://www.cia.gov/cia/publications/factbook. (2014) Visited April, 2014.
3. Odéjobí, O.A.: A quantitative model of yorùbá speech intonation using stem-ml. INFOCOMP Journal of Computer Science **6** (2007) 47–55
4. Adewole, L.O.: The Categorical Status and Function of the Yoruba Auxiliary Verb with some Structural Analysis in GPSG. PhD thesis, University of Edinburgh, Edinburgh (1988)
5. Iyanda, A.R.: Design and Implementation of a Grapheme-to-Phoneme Conversion System for Yorùbá Text-to-Speech Synthesis. PhD thesis, Obafemi Awolowo University, Ile-Ife, Nigeria (2014)
6. Mohri, M.: Finite-state transducers in language and speech processing. Computational Linguistics **23** (1997) 269–311
7. Gakuru, M., Iraki, F.K., Tucker, R.C.F., Shalonova, K., Ngugi, K.: Development of a kiswahili text to speech system. In: INTERSPEECH, ISCA (2005) 1481–1484
8. Anberbir, T., Takara, T.: Development of an amharic text-to-speech system using cepstral method. Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages AfLaT, Athens, Greece ©Association for Computational Linguistics (2009) 46–52
9. Awobùlúyì, O.: Èkọ́ Ìṣẹ̀dá-Ọ̀rọ̀ *Yorùbá* . Montem Paperbacks, Akure, Ondo State (2008)
10. Owólabí, K.: Ìjìnlẹ̀ Ìtúpalẹ̀ Èdè Yorùbá (1): Fònẹ́tîkì ati Fonọ́lọjì. Universal Akada Books Nigeria Limited (2011)
11. Asahiah, F.O.: The development of a Standard *Yorùbá* Diacritics Restoration System. PhD thesis, Obafemi Awolwowo University, Ile-Ife, Nigeria (2014)
12. Sorell, C.: Zipf's law and vocabulary. (2012) In Carol A. Chapelle, editor, The Encyclopedia of Applied Linguistics. Wiley, New York, NY, USA. ISBN 1-4051-9843-5.
13. Hüning, M.: Textstat - simple text analysis tool/ concordance software. available from http://neon.niederlandistik.fu-berlin.de/en/textstat/. visited: April, 2014. (2014)
14. Choffrut, C., Culik-II, K.: Properties of finite and pushdown transducers. SIAM J. Comput. **12** (1983) 300–315