THOMAS ADEWUMI UNIVERSITY, OKO, KWARA STATE
Science | Technology | Medicine

Thomas Adewumi University Journal of Innovation, Science and Technology (TAU-JIST)

ISSN: 3043-503X

RESEARCH ARTICLE

DEVELOPMENT OF A TOKENIZER FOR AN ORTHOGRAPHIC YORÙBÁ-BASED PROGRAMMING LANGUAGE TO ENHANCE INDIGENOUS LANGUAGE COMPUTING

Ezekiel K. Olatunji[1], John. B. Oladosu[2], Stephen O. Olabiyisi[2] , & Odetunji A. Odejobi[3]*

[1]Department of Mathematical and Computing Sciences, Thomas Adewumi University, Oko, Kwara state, Nigeria
[2]Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
[3]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-ife, Nigeria

Corresponding Author Email: ezekiel.olatunji@tau.edu.ng

ARTICLE DETAILS

Article History:
Received 02 July 2024
Accepted 05 October 2024
Available online 10 December 2024

ABSTRACT

The Yorùbá language, with its rich orthographic system, is spoken by over 100 million people worldwide. Despite its widespread use and rich linguistic heritage, the computational resources available for the Yorùbá language are limited. Current programming languages and tools are predominantly designed for English or other widely spoken languages, creating a barrier for those who are only literate in Yorùbá but interested in programming and software development. This research aims to bridge this gap by developing a robust token recognizer for a Yorùbá-based programming language that recognizes and supports the language's modern orthography. The lexical structure of the Programming Language was specified by defining an appropriate regular grammar with the BNF notation and standard Yorùbá words used as reserved words. The Programming Language's character set consists of the basic Yorùbá alphabet. The tokenizer system was implemented on Visual Studio IDE using C# programming language. The test data was created using an online text editor that enables the typing of Yorùbá characters with tonal marks. The developed Tokenizer can determine the different lexemes that form different tokens of the source program in the source language. Besides, it recognizes whitespace characters and detects and reports invalid characters of the source language as well as programmer-directed statements in the source program. The output produced by the system on some given test data showed its functional correctness. This initiative is crucial for advancing indigenous language computing, empowering Yorùbá speakers, and preserving the Yorùbá language through technological innovation. Furthermore, the study has the potential to contribute to driving the fourth Sustainable Development Goal of the United Nations.

KEYWORDS

BNF, Lexeme, Orthography, Token, Yorùbá Language

INTRODUCTION

The global dominance of major languages in computing has created a gap for underrepresented indigenous languages, limiting the technological participation of speakers from diverse linguistic backgrounds (Bird, 2020); especially those that are only literate in their native languages. In response to this, there has been a growing interest in developing programming languages that accommodate indigenous languages, particularly in Africa. One such language is Yorùbá, a tonal and orthographically complex language spoken predominantly in

Quick Response Code	Access this article online	
	Website: https://journals.tau.edu.ng/index.php/tau-jist	**DOI:** https://doi.org/10.5281/zenodo.16632051

Cite the Article: Ezekiel K. Olatunji, John. B. Oladosu, Stephen O. Olabiyisi & Odetunji A. Odejobi. (2024). Development of a Tokenizer for an Orthographic Yorùbá-Based Programming Language to Enhance Indigenous Language Computing

southwestern Nigeria and parts of West Africa (Adewumi, et al. 2020). The development of a programming language that reflects the orthographic and syntactic structure of Yorùbá has the potential to promote indigenous language computing, fostering cultural preservation and technological inclusivity.

Programming Languages (PLs) are more commonly implemented as compilers or interpreters (Olatunji, 2014). The first major component or subsystem of most programming language translators is the tokenizer (also called a lexical analyzer or scanner). A tokenizer determines the sequence of characters that form a lexeme in the PL and thereafter classifies them into their token categories. (Aho et al., 2007; Olatunji 2014). Examples of token categories are numbers, identifiers, reserved or keywords and so on. A lexeme is the actual sequence of characters in a source program that can be grouped together and treated as a single, meaningful unit. For a Yorùbá-based programming language, a token recognizer must accommodate the specific linguistic characteristics of the language, including its diacritics, tonal marks, and phonological structure.

A token of a PL is the smallest unit in a source program. They are terminal symbols of a type-2 language in the parlance of formal grammar from whose theory this study derives its philosophy (Olatunji, et al., 2019). The output of a Tokenizer is a stream of tokens that is sent to the parser for further processing during compilation. A parser is another component of a compiler that carries out syntax analysis on the input sent to it by the Tokenizer.

The Success of this study will facilitate the development of a full-scale standard Yorùbá-based PL. Besides enhancing the comprehension of computer-based problem-solving processes by indigenous learners and instructors which aims at contributing to the fourth Sustainable Development Goals (SDG) of the United Nations (UN) (United Nations, 2022), a Yorùbá-based PL will contribute to ensuring the continued relevance of the Yorùbá language in this age of ICT with its increasing globalization. In addition, the adoption of native language-based PLs like Yorùbá will deter their natural extinction as feared in many circles (Nwafor and Andy, 2022; Olatunji, 2019).

2. REVIEW OF LITERATURE

2.1 The Yorùbá Language

The Yorùbá language is one of the indigenous African languages, and according to Ager (2020), is a member of the Volta-Niger branch of the Niger-Congo family of languages. It is spoken by over 100 million people globally and the first language of over 30 million people in the South-Western part of Nigeria (Eludiora et al, 2015). Yorùbá is a tonal language (Kasali et al, 2021), having three different tones: high, low, and mid. Tones distinguish the meaning of words and are indicated by the use of the acute accent for high tone (e.g. ⟨á⟩, ⟨ń⟩), the grave accent for low tone (e.g. ⟨à⟩, ⟨ǹ⟩); the mid tone is unmarked.

There are twenty-five (25) letters in the Yorùbá alphabet. Figure 1 shows the upper and lowercase of the letters.

Aa Bb Dd Ee Ẹẹ Ff Gg GBgb

Hh Ii Jj Kk Ll Mm Nn Oo

Ọọ Pp Rr Ss Ṣṣ Tt Uu Ww Yy

Figure 1: Yorùbá Alphabet Letters in Upper and Lower Cases.

Three of the Yorùbá alphabets have an under-dot sign. These are Ẹ, Ọ and Ṣ for the upper case letters and ẹ, ọ and ṣ for the lower case letters. These letters are also pronounced differently from their equivalent without the under-dot sign. The tonal signs are usually placed on the vowels in the alphabet, with the exception of the letter i (Eludiora and Ajibade, 2021). The vowels in the Yorùbá lower case letters, in addition to i, are: a, e, ẹ, o, ọ and u.

2.2 Related Works

The need to simplify learning and teaching of computer programming by indigenous people has led to the development of PLs based on the lexicons of many non-English languages. The development of a prototype native language-based programming language (NLPL) that used the lexicons of the Yorùbá language was carried out by Olatunji et al (2021). A major gap in their work is that their design did not take cognizance of the modern orthography of the standard Yorùbá language; particularly the tonal nature of the language with the necessary tone markings that indicate the correct ways of spelling and pronouncing words in the language. This present study seeks to address this gap.

An Arabic-based programming language, called Alf..Eih, was developed by Abdulrasaq et al. (2019) for teaching computer programming to pupils and school children in Arabic countries. The PL was implemented as a source-to-source compiler whose object code is C++ . The PL was also developed for pedagogical reasons as, according to the authors of the PL, students will be assisted in comprehending the idea of programming in their native language rather than when foreign languages are used.

Annamalai (2013) developed a Tamil-based PL called Ezhil. Tamil is an Indian language spoken by over 60 million people (Subramanian, (2015). Ezhil is an interpreted PL which is targeted towards the K-12 (Junior high school) level Tamil-speaking students as an early introduction to thinking like a computer scientist. The syntax of Ezhil is broadly similar to that of conventional BASIC.

The development of the Hindawi programming system (HPS) that allows users to program in Indic languages (Hindi, Bangla, Gujarati, Assamese and some other indic languages) was described in Olatunji, (2019). The HPS, developed by Chaudbary, A and Chaudbary, S., is a free, open-source, programming platform that allows non-English medium literates of India to learn and write computer programs. The system removes the English language barrier and enables non-English literate Indians to take up computer science and

participate in information and communications technology (ICT) at all levels of technology in their mother tongue. It is noteworthy that most of these PLs were developed for pedagogical reasons which, in part, contributes to the fourth SDG of the United Nations (UN). The fourth SDG aims to ensure equitable quality education and promote lifelong learning opportunities for all (UN, 2022)

METHODOLOGY

Appropriate regular (type-3) grammars were designed, using the Backus Norm Form (BNF) notations, to specify the lexical structure of a subset of the Yorùbá-based PL. Some standard Yorùbá words that will not lead to ambiguity were used for the design of the lexical items of the PL. The character set of the PL was formed from the basic Yorùbá alphabets which are 25 in number. The eighth alphabet ('gb') is a special double-character which does not have a Unicode character equivalent. It is a combination of two Latin characters. Each alphabet has both the upper and the lowercase representations. Digits zero to nine and some special characters including simple commonly used arithmetic and relational operator symbols are part of the character set of the PL. Figure 2 is a subset of the design of the PL that is relevant to this study. In Figure 2, words enclosed in corner brackets are called non-terminals symbols, while those not enclosed in corner brackets are terminal symbols in the grammar of the lexical items of the PL. Furthermore, <Nọmba> is a non-terminal symbol from which integer numbers can be generated, while <Feriebu> is the non-terminal symbol from which identifiers in the programming language can be derived.

Definition of < Nọmba> (Number) and < Feriebu> (Variable)

< Nọmba> ⟶ <Dijiti> [{ <Dijiti> }^{4}]

< Feriebu> ⟶ <ABD> [{ <ABD> | <Dijiti>}^{6}]

<Dijiti> ⟶ 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<ABD> ⟶ A | B | D | E| Ẹ | F| G | 'GB'| H | I | J | K | L | M | N | O | Ọ| P| R | S | S | T | U | W | Y |
a | b | d | e | ẹ | f | g | 'gb'| h | i | j | k | l | m | n | o | ọ | p | r | s | ṣ |

<ABDTona> ⟶ á | à | é | è | ẹ́ | ẹ̀ | ó | ò | ọ́ | ọ̀ | ù | ú |
À | Á | É | È | Ẹ́ | Ẹ̀ | Ó | Ò | Ọ́ | Ọ̀ | Ú | Ù

Note:
The definition of <Nọmba> implies it cannot be greater than 99999, while a
< Feriebu> which is an identifier cannot be more than 7 characters long.

Figure 2: Production Rules for <Nomba> and <Feriebu>

Figure 3 is a pseudo-code developed for the implementation of the Tokenizer. The system was implemented in C-Sharp (C#) programming language on Visual Studio IDE. The test data for the system consists of Yorùbá-based source program statements, each of which consists of strings of valid characters in the PL. The test data were created through an online Yorùbá text editor called Lexilogos (Lexilogos.com, n.d). The editor has facility for typing the special alphabet with tonal marks. Any other Yorùbá text editor that has this facility can also be used. A simple fictitious test data used for the system is shown in Figure 4. A peculiar feature of the system is its ability to recognize the Yorùbá alphabet which does not have ASCII representation but only UNICODE representations (Ronacher, 2024).

1. Start
2. Declare all necessary variables
3. Initialize all variables as deemed fit
4. Initialize table of terminal symbols (reserved words, delimiters, operators)
5. Do While not EOF(End of source program file)
   - 5.1 Read a source statement
   - 5.2 Do Until End of source-statement-read
     - 5.2.1 Scan for first /next non-blank character
     - 5.2.2 Classify character scanned
     - 5.2.3 Process the classified character
   - 5.3 Enddo
6. Enddo
7. Stop

Figure 3: Pseudo-code for Implementing the Tokenizer

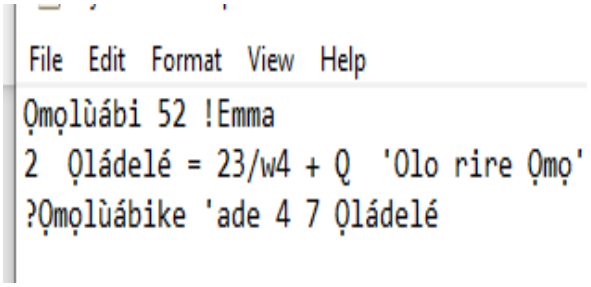

Figure 4: Sample Test Data for the System

RESULTS AND DISCUSSION

Sample outputs produced for using the source statements in Figure 4 as test data are shown in Figures 5a, 5b and 5c. Figure 5 contains the different tokens recognized in the source statements of Figure 4 along with their token categories. The column tagged 'improper token' in Figures 5a, 5b and 5c contains correct lexemes in the PL, though they are not valid tokens of the PL. In Figure 5a, the word "Ọmọlùábi" is deliberately included as a reserved word in the design of the PL when the system was being tested.

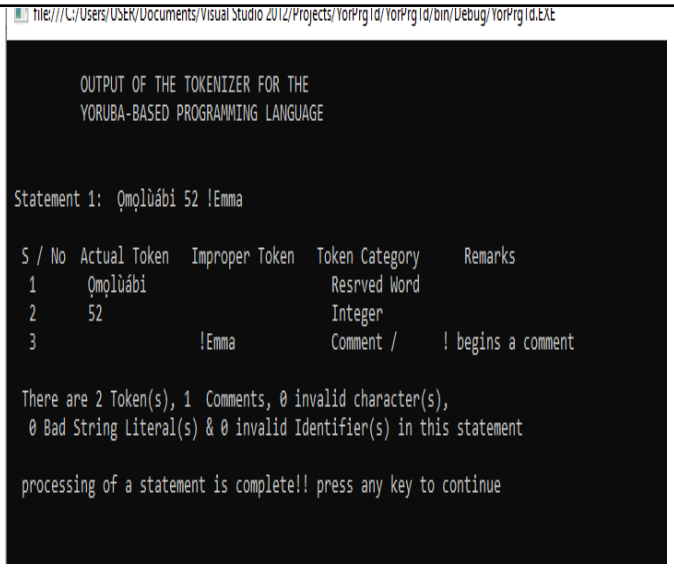

Figure 5a: Output of the Tokenizer for First Source Statement

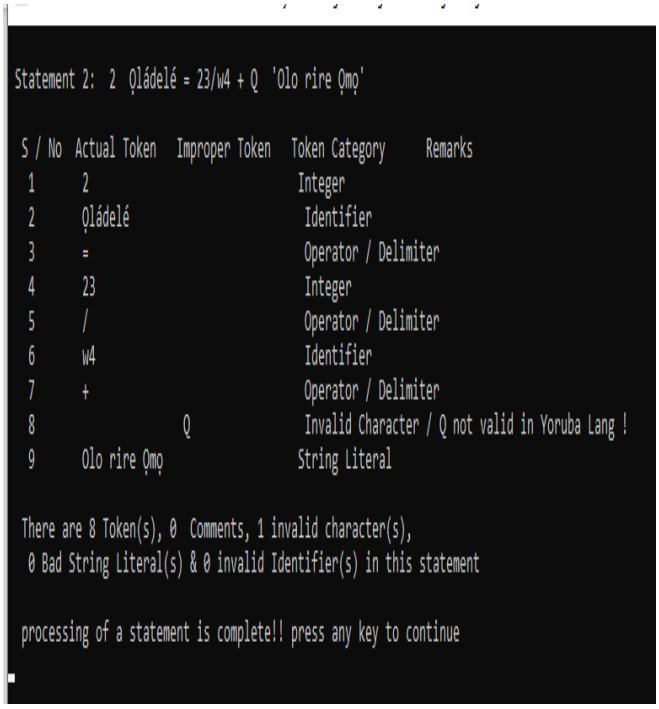

Figure 5b: Output of the Tokenizer for the Second Source Statement

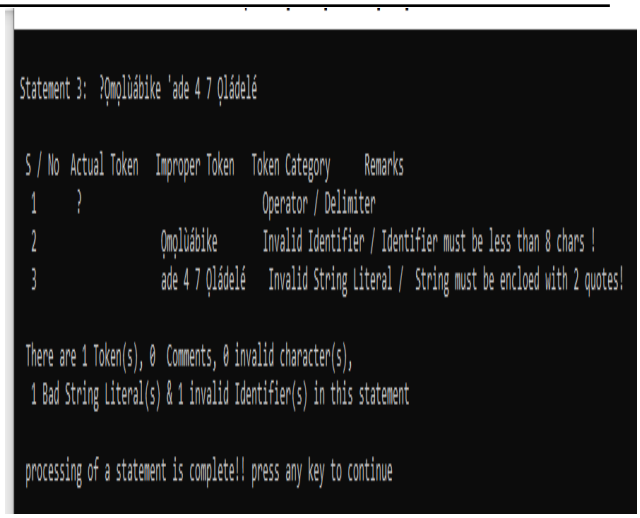

Figure 5c: Output of the Tokenizer for the Third Source Statement

Examination of Figure 5 shows that the implemented Tokenizer is working correctly. For instance, in the test data, an exclamatory mark (!) precedes a comment statement, while a string literal is enclosed in two single quotes. These are correctly recognized as shown in Figure 5a for source statement one and Figure 5b for source statement two respectively. Furthermore, Figure 5b shows that the character Q in statement 2 of Figure 4 is correctly recognized as being invalid in the alphabet of the Yorùbá language. The design of the PL specifies that the length of an identifier must not be greater than seven (7) characters as shown in Figure 2. This is correctly recognized as such in Figure 5c for the third source statement of the test data in Figure 4.

CONCLUSION AND FUTURE WORK

The development of a Tokenizer for a subset of the Yorùbá-based PL with the lexicons and modern orthography of the standard Yorùbá language will enhance the full-scale development of a compiler for the Yorùbá-based PL. Without any controversy, a Yorùbá-based PL, when fully developed, will provide an indigenous platform for the introduction of computer programming to indigenous pupils of both senior primary and junior secondary schools. This research therefore contributes partly to accomplishing the UN's fourth SDG. Development of a suitable parser and semantic analyzer for the Yorùbá-based PL is the next major work in this research endeavour.

REFERENCES

Abdulrazaq, H.H., Gasera, A.S., Mohammed, M. A., ... Farhana, R.N. (2019), Designing and Implementing an Arabic Programming Language for Teaching Pupils, *Journal of SouthWest Jiaotong University*, vol. 54, No. 3, June 2019.; DOI:1035741/issn.0258 2724.54.3.11

Adewumi, T.P., Liwicki, F and Liwicki, M. (2020), The challenge of Diactirics in Yorùbá Embeddings. In ML4D Workshop at 34th Conference on Neural Information Processing Systems (NeurIPS2020), vancour, Canada

Ager, S (2020) The online encyclopedia of writing systems and languages. [Online].

Available:https://www.omniglot.com/writing/ Retrieved December 2023

Aho, A. V.; Lam, M. S.;Sethi, R. and Ullam, J.D (2007),Compilers – Principles, Techniques and Tools, New York,Pearson Education/ Addison Wesley, 2nd ed; pp1040

Annamalia, M. (2013). Invitation to Ezhil: A Tamil Programming Language for Early Computer Science Education; Retrieved from http://ezhillang.org on 17-03-2015

Bird, S. (2020). *Decolonising speech and language technology*. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 6–15.

Eludiora, S. and Ajibade, B. (2021). Design and implementation of English to Yorùbá Verb Phrase Machine Translation System. In Proceeding of the Africa NLP Workshop - International Conference European Chapter of the Association for Computational Linguistics [EACL 202], Ukraine, DOI:10.4855/arXiv.2104.04125

Eludiora, S. I.; Agbeyangi, A.O. and O.I. Fatusin (2015) ; "Development of English to Yorùbá Machine Translation System for Yorùbá verbs's Tone Changing", *International Journal of Computer Applications,* vol. 129, No. 10, 2015

Kasali, A.A., Jimoh, K.O., Adeagbo, M.A. and Bello, S. A., (2021), Web-Based Text Editing System for Nigerian Major Languages, *Nigerian Journal of Technology, Vol. 40, No. 2, 2021, pp. 292–301.* http://dx.doi.org/10.4314/njt.v40i2.15Lexilogos (n,d), Available online from https://www.lexilogos.com/english/index.htm Retrieved on 25/07/2023

Nwafor, E. and Andy, Anietie (2022), A survey of Machine TranslationTasks on Nigerian languages; in Proceedings of the 13th Conference on Language Resources and Evaluation [LREA, 2022], Marsielle

Olatunji, E. K., Oladosu, J. B. and Olabiyisi, S. O. (2022*), Towards the Development of a Token Recognizer for a Yorùbá-based Programming Language;* In Proceedings of International Conference on Information systems and Emerging Technologies, Nambia Universityof Technology (ICISET), 23-25 November, 2022

Olatunji, E. K., Oladosu, J. B., Odejobi, O. A. and Olabiyisi, S. O. (2021). Design and Implementation of an African Native Language based Programming Language.International Journal of Advances in Applied Sciences (IJAAS), 10(2); 171-177. https://doi.org/10.11591/ijaas.v2.i2.pp171-177

Olatunji, E. K. (2019). Development of a Programming Language with Yorùbá Lexicons. Unpublished PhD Thesis, Ogbomoso-Nigeria, Ladoke Akintola University of Technology.

Olatunji, E. K., Oladosu, J. B., Odejobi, O. A. and Olabiyisi, S. O. (2019). Design of an African Native Language-based Programming Language. University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR), 3(1); 72-78

Olatunji, E.K. (2014), "Programming Languages – Introductory Text on Concepts and Principles", Igbagbogbemi Publishers, Ilorin, Nigeria, ISBN 978-187-458-9, pp 8, 2014.

Ronacher, Armin (2024); Information, Characters, Unicode. Available online from https://cs.fit.edu/~ryan/cse1002/lectures/unicode.pdf Retrieved on 15-02-2024

Subramanian, K. (2015), Bridging the Digital Divide with Tamil Coding; Retrieved from www.thehindu.com/~/making computing-in-tamil-easy/ on 17-3-2015

United Nations(2022), Together 2023 – Position Paper -Sustainable Development Goals; Available online from https://sustainabledevelopment.un.org/ ; Retrieved on 21-02-2024.