Pushdown Automata Model for Syntax Error Detection in Yorùbá Simple Declarative Sentences

Supported by tetfund TERTIARY EDUCATION TRUST FUND

Abisola Ayomide Olayiwola[1,*], Dare Samseedeen Olayiwola[2], Ajibola Oyedeji[1,3], Martins Ositola Osifeko[1], Ayodeji Akinsoji Okubanjo[4]

[1]Department of Computer Engineering, Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria.
[2]Department of Computer Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
[3]School of Electrical and Electronics Engineering, University of Johannesburg, Johannesburg 2006, South Africa.
[4]Department of Electrical/Electronics Engineering, Olabisi Onabanjo University Ago-Iwoye, Ogun State, Nigeria
Corresponding Author: olayiwola.abisola@oouagoiwoye.edu.ng

Received: February 14, 2025, Accepted: April 28, 2025

Abstract: This study designed a model that can detect syntax errors in Yorùbá simple declarative sentences using pushdown automata (PDA). Information on the structure of Yorùbá simple declarative sentences was gathered and analysed. The structure of these sentences was then modelled using PDA. The PDA model was simulated using Java Formal Languages and Automata Package (JFLAP). The behaviour of the model was explored by testing several case scenarios. The model accepted all the cases with the correct syntax and rejected all cases with the incorrect syntax. The PDA model, if implemented, can serve as the logic behind error detection in a Yorùbá grammar checking.

Keywords - Modelling, Pushdown automata, Yorùbá, grammar checking, Yorùbá, syntax

Introduction

Yorùbá is spoken by millions of people, primarily in southwestern Nigeria and parts of neighbouring West African countries. *Yorùbá* is gradually becoming a recognized language globally. Recently, the Government of China started teaching *Yorùbá* language in its schools. As more and more individuals and organizations conduct written communication in *Yorùbá*, the need for a *Yorùbá* grammar checker becomes more important. *Yorùbá* grammar checker aims to provide users with a reliable tool to identify and fix grammatical errors, punctuation errors, and other language-related issues specific to *Yorùbá*.

Grammar checking involves both syntactic and semantic analysis. Syntax is a branch of language which is concerned with how the rules of grammar are applied in the ordering of words to form phrases, clauses and sentences. Grammar errors may be present in sentences that have syntactic errors. Syntactic analysis may serve as an intermediate stage for semantic analysis representation. Context-free grammar (CFG) is the backbone of many formal models of the syntax of human language and of computer languages (Jurafsky & Martin, 2000). One way to implement context free grammar is using PDA.

A lot of research has been done on developing both online and offline grammar checkers for high resource languages. However, there is a dearth of automatic grammar checkers for under-resourced languages such as *Yorùbá*. Developing a system that can detect errors in *Yorùbá* syntax will assist *Yorùbá* language users in improving the quality and clarity of their written communication. The aim of this study is to model and simulate a computational model that can detect syntax errors in *Yorùbá* text.

Automata has gained much attention in morphology. Several authors have employed Finite automata in the analysis of morphology of several languages. An example of such work is a model that can analyse the morphology of Oriya's negative verbal form was created. This model was developed using finite automata. The model accepts correct morpheme sequences in a negative verbal form and rejects incorrect sequences (Sahoo, 2002). Similarly, Mahyoob, (2018) designed a model that can analyze the morphology of different verb forms in Arabic using deterministic finite state automata (DFSA) (Mahyoob, 2018). A morphological parser was built with the help of a lexicon. An accuracy of 96.00% was obtained. The same author designed a model that can fully analyse the morphology of all Arabic pronoun forms using deterministic finite state automata (Mahyoob, 2020). A computational lexicon that contains all the possible inflections of Arabic pronouns was designed. The input string, morphological rules and lexicon were represented using the deterministic finite state automat (DFSA). F-score of 80.43% was obtained. A group of authors designed a model for the morphological analysis of Urdu nouns using DFSA (Alblwi et al., 2023). A lexicon was built to help with the morphological analysis. The DFSA consists of thirteen (13) states and 21 transitions. An accuracy of 92.70% was achieved. A research designed a morphological analyser for Malayalam using finite automata (Liji, 2021). The outputs of the morphological analyser for nouns are the information about the number, gender, and case, while for the verbs, the outputs are modularity and tense. Similar work was conducted by (Girija & Anuradha, 2017). They also built a morphological analyser for Malayalam text using finite automata. For morphological analysis of Malayalam text, a lexicon consisting of noun and verb stems was built, lists of suffixes and list of morphophonemic rules were also gathered. A model that describes the morphological structure of nouns in the Uzbek language using finite automata was built (Bakaev & Shafiev, 2020). The model considers all word-forming suffixes, affixes, and word forms that fall under the part of speech, nouns.

Several authors used PDA to detect if a sentence is syntactically correct or not. For instance, PDA was used in the design of English language recognizer (Pasha & Khan, 2014). English language context-free grammar (CFG) rules were constructed. These rules were then converted into PDA. The PDA model was tested by using it to parse different English sentences. The model was able to parse valid English sentences. Similarly, a PDA that can parse Bangla sentences was designed (Rahman et al., 2017). Bangla language syntax was represented using CFG rules. The CFG rules served as input to the PDA. The PDA was able to parse syntactically correct Bangla sentences and unable to parse syntactically incorrect Bangla sentences.

Little has been done around *Yorùbá* grammar checking. The only available work evident in the literature is the development of a *Yorùbá* grammar checker that could detect syntactic errors in simple, compound and complex sentences

(Olayiwola et al., 2024). The Government and Binding theoretical framework were employed to create parse trees for sentences. A sentence that cannot be parsed is considered grammatically incorrect while a sentence that can be parsed is considered grammatically correct. In evaluating the accuracy of the system, three hundred (300) sentences were used and an accuracy of 88.89%, 88.89% and 86.67% were gotten for compound, complex and simple sentences respectively. This system will only detect that a sentence is not grammatically correct. It will not be able to point to the error(s) in the sentence.

Materials and Methods

The methodology involves several steps such as gathering information on *Yorùbá* syntax, system design assumption, model design, model simulation and testing.

Information gathering

Information on correct word formation of *Yorùbá* sentences, and various parts of speech in *Yorùbá* were gathered from textbooks and journals. This information was augmented with the knowledge of the authors as *Yorùbá* native speakers. This information was analysed and used in the design of the error detection model for *Yorùbá* syntax.

System Design Assumption

Some assumptions were made to reduce the problem to that which is amenable to computing treatment. Therefore, for the successful simulation of the system, the following assumptions were made. The initial state is state q_0. This is the state in which the stack is empty. Each other states, q_1 to q_{15} represent when the model expects a particular part of speech. q_1 is the state in which the machine is expecting a noun or pronominal. Nouns and pronominals are categories of words that can be used to start a *Yorùbá* simple declarative sentence. If the model sees any word whose part of speech does not belong to any of these three (3) categories, the word will be underlined, the word will be underlined, and the model will reject such a sentence. State q_2 is the state in which the system has already seen a noun or pronominal and expecting a word in any of the following categories: conjunction (CONJ), auxiliary verb (AUX), main verb (V), qualifier (Q), relative clause (REL). If the system sees any word not belonging to these categories, such word is underlined. State q_3 is the state in which the system has already seen a noun or pronominal and expecting a qualifier. Also, the model remains at this state if it continues to see a qualifier. If the system sees any word that is not a qualifier, the word will be underlined, and the model will reject such a sentence. State q_4 is the state in which the system has already seen a relative clause marker and expecting a noun, verb, pronoun object, adverb or preposition. If the system sees any word that is not a verb, the model will reject such a sentence. State q_5 is the state in which the system has already seen a preposition and expecting a noun. If the system sees any word that is not a noun, the model will reject such a sentence. Likewise, at state q_7, the system has already seen a pronoun and is expecting an auxiliary verb or a main verb. If the system sees any word not belonging to these two categories, such sentence will be rejected. State q_8 is the state in which the system has already seen an auxiliary verb and expecting a verb. If the system sees any word that is not a verb, such word will be underlined. State q_{14} is the state in which the system has seen a verb and expecting a noun or pronoun object. If the system sees any word that is not a verb, the word will be highlighted. The final states are the state at which the model is expecting an adverb (Adv) (q_9), verb (V) and noun (q_6) and (q_{12}), qualifier (q_{10}), pronoun object (Prnobj) (q_{11}). Words belonging to these categories can be used to end a *Yorùbá* sentence.

Designing the model

A Pushdown automaton (PDA) consists of a finite automaton and a stack for storing an unlimited number of symbols. The syntax of *Yorùbá* simple sentence was modelled using a PDA. A pushdown automaton is made of several components. These components are defined in Definition 1.

Definition 1 A pushdown automaton M is an abstract model formally defined as a 7-tuple, that is,

$$DFA = (Q, \Sigma, \delta, \Gamma, Q_{Start}, Z, F>)$$

Where,

- Q is a finite set of states that are not empty.
- Σ represents a finite set of input symbols that are not empty.
- δ denotes the transition function.
- Γ is a finite set of stack symbols that are not empty.
- Q_{Start} represents the start state and $Q_{Start} \in Q$.
- The start symbol for the stack is represented by Z.
- F denotes the set containing the end states and $F \subseteq Q$

The various components of the PDA are described below:

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$

Γ= {N (noun), V (verb), Q (qualifier), Prn (pronoun), Adv (adverb), P (preposition), CONJ (conjunction), Z (empty stack)}

Σ= {λ, N, V, Q, Prn, Adv, P, CONJ, REL}

$Q_{Start} = q_0$ is the start state and $Q_0 \in Q$

$F = \{q_{13}, q_4, q_6, q_9, q_{10}, q_{11}, q_{12}\}$ $F \subseteq Q$

Simulating the model

The model was simulated using Java Formal Languages and Automata Package (JFLAP) as shown in Figure 1.

Results and Discussion

The system was tested with ten (10) correct and five (5) incorrect cases as shown in Figure 2 and interpreted in Table 1. These cases are explained in Table 2. In Table 2, cases 1-10 were accepted by the model while 11-15 were rejected by the model. This is because cases 1-10 followed the principles guiding *Yorùbá* syntax while 11-15 violated the rules guiding *Yorùbá* syntax.

For instance, given a *Yorùbá* sentence with the sentence structure NQVPN, that is, noun followed by a qualifier, and then a verb and then a preposition and lastly, a noun. Example of such sentence structure is *Bísọ́lá pupa lọ sí oko*, were,

1. Noun = *Bísọ́lá*, oko
2. Qualifier = pupa
3. Verb = lọ
4. Preposition = sí

The transition table is shown in Table 2 while the transition diagram is shown in Figure 10. At state q_1, the noun (N) *Bísọ́lá* is read while the stack is empty. This causes the noun to be pushed on the stack, and the machine moves to state q_2. At state q_2, the adjective (Adj) *pupa* was seen, while *Bísọ́lá* is on top of the stack. This takes the machine to state q_3. At state q_3, the verb (V) *lọ* is seen while the adjective pupa is seen on top of the stack. This takes the machine to state q_4. At state q_4, the preposition (P) *sí* is seen while the verb is on top of the stack. Finally, at state q_5, the noun *oko* is read. This causes the machine to move to state q_6. At the current state, all the tokens in the expression have been processed, and no input symbol is read. Since the sentence causes the machine to halt at q_6 which is a valid final state, the sentence is recognized as a valid *Yorùbá* sentence.

Table 1: Testing the PDA model

S/N	Sequence	Example	Result
1	NQVPN	*Délé pupa lọ sí oko*	Accept
2	NAuxV	*Délé ń jó*	Accept
3	NQAuxV	*Délé pupa ń jó*	Accept
4	NQCONJNQV	*Délé pupa àti Tití dúdú ń jó*	Accept
5	PrnV	*Mo jẹun*	Accept
6	NV	*Délé jẹun*	Accept
7	NVN	*Délé se oúnjẹ*	Accept
8	NAuxVPN	*Délé ń jó sí ìlù*	Accept
9	NVAdv	*Ó jó gidigidi*	Accept
10	PROCONJNV	*Èmi àti Délé jẹun*	Accept
11	PrnObjV	*Mi jẹun*	Reject
12	PrnQV	*Mo pupa jẹun*	Reject
13	NVAux	*Délé jó ń*	Reject
14	NVQ	*Délé lọ pupa*	Reject
15	QNV	*pupa Délé lọ*	Reject

Table 2: Mapping functions for the move in PDA

Input	**Symbol**	**Current state**	**Transition**	**Next state**	**Stack**
NQVPN	-	q_0	-	q_1	Z
NQVPN	N	q_1	δ (q_1, N, -, push (N), q_2)	q_2	NZ
QVPN	Adj	q_2	δ (q_2, Q, -, push (Q), q_3)	q_3	QNZ
VPN	V	q_3	δ (q_3, V, -, push (V), q_4)	q_4	VQNZ
PN	P	q_4	δ (q_4, P, -, push (P), q_5)	q_5	PVQNZ
N	N	q_5	δ (q_5, N, -, push (N), q_6)	q_6	NPVQNZ
-	-	q_6	δ (q_6, -, pop, -, q_{12})	q_{12}	PVQNZ
-	-	q_6	δ (q_6, -, pop, -, q_{12})	q_{12}	VQNZ
-	-	q_6	δ (q_6, -, pop, -, q_{12})	q_{12}	QNZ
-	-	q_6	δ (q_6, -, pop, -, q_{12})	q_{12}	NZ
-	-	q_6	δ (q_6, -, pop, -, q_{12})	q_{12}	Z
-	-	q_6	δ (q_6, -, pop, push (Z), q_{12})	q_{12}	Z

(i)

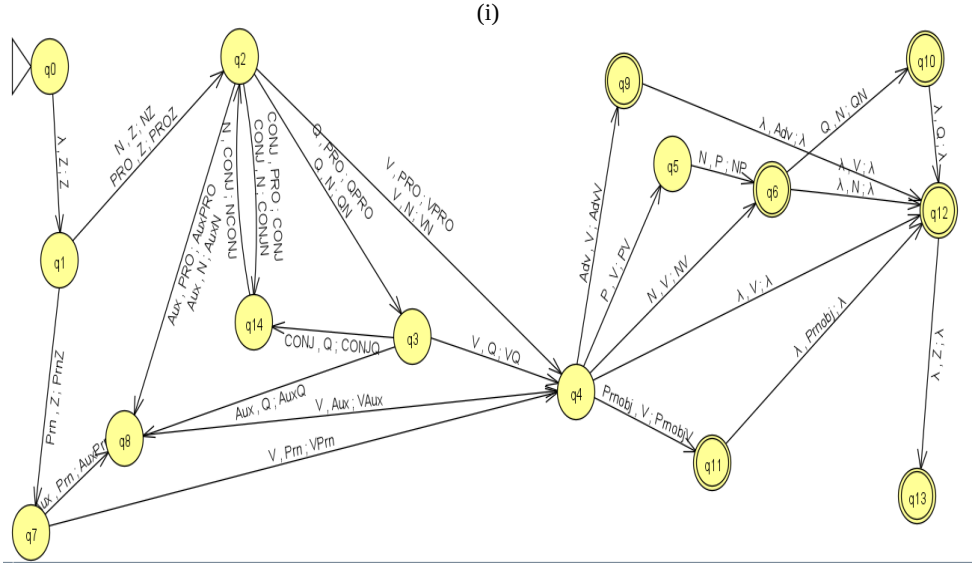
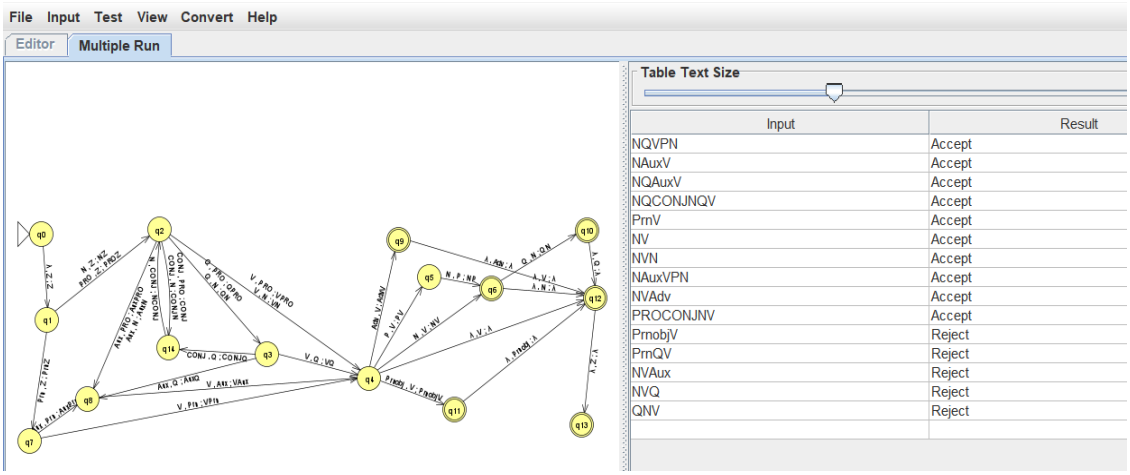


Figure 1: Transition diagram of the *Yorùbá* grammar error detector using PDA

ii)



Input	Result
NQVPN	Accept
NAuxV	Accept
NQAuxV	Accept
NQCONJNQV	Accept
PrnV	Accept
NV	Accept
NVN	Accept
NAuxVPN	Accept
NVAdv	Accept
PROCONJNV	Accept
PrnobjV	Reject
PrnQV	Reject
NVAux	Reject
NVQ	Reject
QNV	Reject

Figure 2: Testing the PDA model

Conclusion

For high resource languages (such as Arabic, English, and Chinese), a number of grammar checkers have been created with various levels of accuracy. However, there has not been much progress in developing computerized grammar checkers for low-resource languages, such as Standard *Yorùbá*. Developing a syntactic analyzer is a step in the right direction. In this work, a model for *Yorùbá* syntax was created using push down automata. The model was tested using several case scenarios of a *Yorùbá* simple sentence. The sentences with the correct syntax were accepted by the PDA while sentences with the wrong sequence were rejected by the PDA. The model, if implemented, will not only detect that a sentence is syntactically incorrect, it will also be able to detect where the error lies in a sentence. Further studies should be done to implement the model designed in this work.

References

Alblwi, A., Mahyoob, M., Algaraady, J., & Mustafa, K. S. (2023). A Deterministic Finite-State Morphological Analyzer for Urdu Nominal System. *Engineering, Technology & Applied Science Research*, *13*(3), 11026–11031. https://doi.org/10.48084/etasr.5823

Bakaev, I. I., & Shafiev, T. R. (2020). Morphemic analysis of Uzbek nouns with Finite State Techniques. *Journal of Physics: Conference Series*, *1546*(1). https://doi.org/10.1088/1742-6596/1546/1/012076

Girija, V., & Anuradha, T. (2017). Application of Finite State Methods in Malayalam Text Analysis. *International Journal of Computer Applications*, *168*(12), 43–47. https://doi.org/10.5120/ijca2017914516

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. 908.

Liji, S. K. (2021). Morphological Analyser for Malayalam Language. *Journal of Emerging Technologies and Innovative Research*, *8*(7), 634–637.

Mahyoob, M. (2018). Deterministic Finite State Automaton of Arabic Verb System: A Morphological Study. *International Journal of Computational Linguistics (IJCL)*, *Volume (9)*(9), 13–25. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3806730

Mahyoob, M. (2020). Developing a Simplified Morphological Analyzer for Arabic Pronominal System. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3599719

Olayiwola, A., Olayiwola, D., & Oyedeji, A. (2024). Development of an automatic grammar checker for Yorùbá word processing using Government and Binding Theory. *Expert Systems with Applications*, *236*. https://doi.org/10.1016/j.eswa.2023.121351

Pasha, M. K., & Khan, M. S. A. (2014). To Design a English Language Recognizer by using Nondeterministic Pushdown Automata (ELR-NPDA). *International Journal of Computer Applications*, *105*(1), 36–43.

Rahman, M., Abdulla-Al-sun, Hasan, K. M. A., & Shuvo, M. I. R. (2017). Designing a bangla parser using non- Deterministic push down automata. *ECCE 2017 - International Conference on Electrical, Computer and Communication Engineering*, 571–576. https://doi.org/10.1109/ECACE.2017.791297.

Sahoo, K. (2002). A deterministic finite state automaton for the Oriya negative verbal forms. *Proceedings - Language Engineering Conference, LEC 2002*, 110–120. https://doi.org/10.1109/LEC.2002.1182298