



VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS  
INFORMATICS STUDY PROGRAMME

**Algorithms for transliteration of foreign languages to  
Lithuanian**

Master's Theses

Author: David Olurebi.

VU email: david.olurebi@stud.mif.vu.lt

Supervisor: Assoc. Prof. Dr. Pijus Kasparaitis.

Reviewer: Prof. Dr. Linas Laibinis

Vilnius - 2023

# Contents

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
<b>3</b>	<b>Background</b>	<b>10</b>
3.1	Terminologies . . . . .	10
3.2	Phonetic typography . . . . .	11
<b>4</b>	<b>Techniques and related works</b>	<b>13</b>
4.1	Rule-based Machine Transliteration . . . . .	13
4.1.1	Context-sensitive and Context-free . . . . .	14
4.1.2	Syllabification . . . . .	14
4.2	Statistical Machine Transliteration . . . . .	15
4.2.1	Noisy Channel Model . . . . .	16
4.2.2	Finite State Transducers (FST) . . . . .	16
4.2.3	Hidden Markov Model (HMM) . . . . .	16
4.3	RBMT vs SMT . . . . .	17
<b>5</b>	<b>Transliteration Routes and Data</b>	<b>18</b>
5.1	Routes . . . . .	18
5.1.1	Direct . . . . .	18
5.1.2	Intermediate . . . . .	18
5.2	Data . . . . .	20
<b>6</b>	<b>Rule-based Transliteration</b>	<b>22</b>
6.1	Syllabification . . . . .	22
6.2	Alignment . . . . .	25
6.2.1	Alignment for Languages . . . . .	25
6.2.2	Alignment for IPA . . . . .	26
<b>7</b>	<b>Machine Learning</b>	<b>28</b>
7.1	Decision Tree (CART) . . . . .	28
7.1.1	Alignment . . . . .	30
7.1.2	Pruning . . . . .	31
7.1.3	Hyper-parameter tuning . . . . .	33
<b>8</b>	<b>Experiment Appraisal and Conclusion</b>	<b>34</b>
8.1	Evaluation of RBMT approach . . . . .	35
8.1.1	Direct transliteration . . . . .	35
8.1.2	Transliteration using English and Georgian as intermediate languages . . . . .	35
8.1.3	Transliteration using IPA as an intermediate language . . . . .	37
8.2	Evaluation of machine learning approach . . . . .	38
8.2.1	Direct transliteration using ML . . . . .	38
8.2.2	Intermediate transliteration using ML . . . . .	39
8.3	Conclusion . . . . .	40

<b>References</b>	<b>41</b>
<b>9 Appendix. Transliteration Software</b>	<b>45</b>

# 1 Abstract

The continuous trend in globalization of people and world economies demands effective and efficient worldwide information access across language barriers. According to [17], around 5,400 new words are created every year; it is only the 1,000 or so deemed to be in sufficiently widespread use that make it into print. Though automatic translation of words from one language to another has helped bridge that barrier, the adaptation of out-of-vocabulary words such as proper names in such a way that preserves the grammatical or phonetic structure of the target language has proven a daunting task.

This work explores the transliteration of Yoruba proper nouns into the Lithuanian language by the means of two routes: direct and intermediate. The latter is the adaptation of out-of-vocabulary words from source to target whilst utilizing another language resource, such as the International Phonetic Alphabet (IPA) or a language, at the hub of the transliteration procedure, whereas the former is the customary transliteration process. As part of the intermediate route solution, when using the IPA, we developed a syllabification algorithm with an accuracy of 99.7% to facilitate the correct transcription of Yoruba phonemes to their phonetic alphabets before mapping the source IPA to the target IPA (Lithuanian IPA). On both routes, we experimented the performance of the classification and regression tree (CART) learning against a rule-based (context-free and context-sensitive) approach with the aim of establishing (i) which of the two routes adapt Yoruba names better to Lithuanian (ii) how the rule-based approach compares with machine learning in respect to both routes (iii) how the language resources employed in the intermediate route compare with each other and which does what better.

**KEYWORDS:** Natural Language Processing, machine translation, machine transliteration, context-sensitive rules, context-free rules, Lithuanian, Yoruba, Georgian, IPA, Decision Trees.

## 2 Introduction

Machine translation has been in existence since the 1940s and has flourished in recent times due to the continued advancements in language technologies. "machine translation was the first computer-based application in natural language processing (NLP) and its history is old" - [16]. The field is said to have served as the forcing function for computer science itself when the search for automatic means of translation between English and Russian assumed importance in the 60s due to the cold war. War and commerce have been the two drivers of translation technology. Machine transliteration (MT) emerged as part of the machine translation process a few decades ago and has since evolved into an independent branch of NLP.

In attestation to the growth of NLP as an interdisciplinary subfield of linguistics, computer science, and artificial intelligence, works on MT alone can be categorized on several basis. The direction of a transliteration process i.e., forward or backward is one way of categorizing MT where forward transliteration, loosely speaking, transliteration, is the systematic conversion of a given name pair  $(o, t)$ , from  $o$  (original) to  $t$  (transliterated version of  $o$  in another language); backward transliteration (or back-transliteration) is retrieving the correct  $o$  given  $t$ . For example, the Lithuanization of the capital city of Oyo state, Nigeria, "Ibadan"  $\rightarrow$  "Ibadonas" is typical forward transliteration and the corresponding back-transliteration is "Ibadonas"  $\rightarrow$  "Ibadan". Another means to classify MT is by the scripts involved in the procedure. For instance, an adaptation procedure between Russian (Cyrillic writing system) and Lithuanian (Latin writing system) can be called Latin-Cyrillic transliteration or Latin-Han when the languages involved in the process are any Latin-writing languages and perhaps, Chinese. Whilst the classification list can become exhaustive when the many factors that play roles in a transliteration mechanism are considered, there is one categorization that applies to all others: transliteration unit.

Regardless of the direction or writing script, MT can be categorized in terms of the level of units adapted from source to target at a time. These units are grapheme and phoneme and are sometimes referred to as the direct and pivot methods, respectively. Grapheme-based transliteration, conceptually, involves a direct orthographical mapping from source graphemes to target graphemes ignoring the phoneme-level processes. A grapheme-based transliteration rule may be represented as  $S_p \rightarrow T$  where  $S$  is a grapheme of the source language;  $T$  is a grapheme of the target language; and  $p$ , the possibility of transliterating  $S$  to  $T$ .

Table 1: Grapheme-based transliteration

J	A	M	I	U
DŽ	A	M	I	U

Let us look at the grapheme-based example demonstrated in the Table 1 above and we will notice that the transcription of the Yoruba name, "Jamiu", involved direct mapping of characters (grapheme) from source to target with no consideration for the vowel glide in the source word as in the Table 2.

Table 2: Phoneme-based transliteration

J	A	M	I	U
DŽ	A	M	I	JU

While Grapheme-based models work by directly transforming source language graphemes into target language graphemes without explicitly utilizing phonology in the bilingual mapping [21], in pivot models, name transliteration occurs on the basis of pronunciation i.e., the written word of source language is mapped to written word of target language via the spoken form associated with the word.

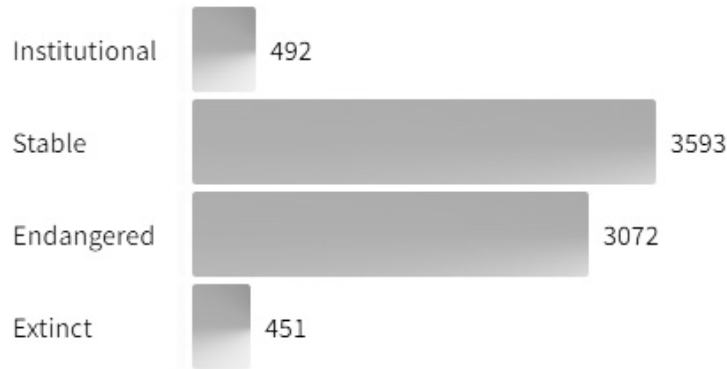


Figure 1: Language vitality count sourced from ethnologue

Though machine transliteration is broad by categorization and there are several techniques in existence to approach it, the devising of an intermediate hub approach is not very common. According to data provided by [10] there are 7000 languages in the world all of which can be grouped into 4 as in the Figure 1. [10] continued with this statement regarding the figure above: "A language is said to be institutional if it is developed to the point that it is used and sustained by institutions beyond the home and community; stable, if it is not being sustained by formal institutions, but it is still the norm in the home and community; endangered, if it is no longer the norm it is learnt and used; extinct, if such language is no longer used and no one retains a sense of ethnic identity associated with the language." Notwithstanding that both the Yoruba and Lithuanian languages fall among the 6% of the world's institutional languages, where Yoruba is spoken by a population of up to 50 million people compared to Lithuanian's 3 million, the former is a low-resourced language and the latter a high-resourced language when compared together. From an NLP standpoint, the copious availability and access to a language's data is the determinant to whether such language is high or low in resources. Lithuanian satisfies the requirements expected of a language to aid NLP such as written language corpora, spoken language corpora, corpus exploration and exploitation tools, modules (e.g. taggers, morphological analysers, parsers, speech recognizers, text-to-speech) etc., as specified in [25]'s Basic Language Resource Kit (BLARK) - a finding validated by the fact that NLP kits such as spaCy have active and optimized supports for the Lithuanian language but not for Yoruba. Therefore, with an intermediate route option, we have the possibility to utilize a language such as English (the language with most speakers and plethora of NLP data and resources) at the hub of the transliteration process to select a tool, language data or resource such as established transliteration rules capable of aiding transcription from Yoruba to Lithuanian or any other high-resource language.

When compared with other research areas in NLP, machine transliteration is relatively immature and has its drawbacks, some of which includes ambiguous standards and lack of compar-

isons. About ambiguity in standards, transliterations, typically, are handcrafted rules of thumb that are not easy to expand and are mostly subjective, i.e., they are subjected to the interpretation of individual producers. De facto standards have been established, but the rules are often inconsistently used. For example, rules for English spelling of the grapheme ⟨c⟩ to phonetics, defined by [9] are as follows:

[C]A → /k/

[C]O → /k/

[C] → /s/

This set of rules is read as: the grapheme ⟨c⟩ sounds as /k/ if it is followed by ⟨a⟩ or ⟨o⟩ and it sounds as /s/ if otherwise. Detecting phonemes of the words being processed is an important part of these systems, and directly affects the accuracy of these rules [19]. The first rule is context-sensitive and the second one is context-free and will successfully process words like CAKE, COOL, CEASE, CIGAR [20]. In contradictory cases such as CELLO, it is suggested that such words be included in a list of exceptions rather than changing existing rules. For example:

[C]E → /tʃ/ as in /ch/ → (ch)air”

As regards lack of comparisons, there is neither a widely recognized benchmark nor consistent measurement for machine transliteration, making it hard for comparative evaluation. Furthermore, the evaluation process often require human judgement and although the accuracies of existing transliteration techniques are unsatisfactory, [13] says there is no useful guidelines to improve them. On a final note on some of the drawbacks of MT, the validity of transliteration in documentation is questioned as a result of the loss of precise information. [46] examined the process from the linguist’s, cataloguer’s, and user’s points of view.

So far, we have familiarized with machine transliteration as an area of NLP, some of the categorization methods of transliteration, the importance of introducing an intermediate route, what makes a language high-resource - why the Lithuanian language is one and Yoruba is not and some of MT’s stumbling block. The remainder of this introductory section is dedicated to familiarizing with our language pair.

## Yoruba Language

Yoruba is a language primarily spoken in southwestern and central Nigeria by the ethnic Yoruba people with the number of Yoruba speakers (including outside of Nigeria) estimated 50 million. Historically (dating back to the latter part of the 17th century), the Yoruba language was written in the Arabic script (Ajami), the *standard* Yoruba orthography is, however, written in the Latin script with its origin dating back to the mid 19th century when Samuel A. Crowther, the first native African Anglican bishop, published a Yoruba grammar and started his translation of the Bible which included peculiar translation features such as *calquing* [31].

The current orthography of the Yoruba language employs the Latin alphabet which includes the use of digraph <gb> and certain diacritics, including the underdots under the letters <ẹ>, <ọ>, and <ş> [40].

Consonants:

B	D	F	G	GB	H	J	K	L	M	N	P	R	S	Ş	T	U	W	Y
b	d	f	g	gb	h	j	k	l	m	n	p	r	s	ş	t	u	w	y

Vowels:

A	E	Ẹ	I	O	Ọ	U
a	e	ẹ	i	o	ọ	u

The pronunciation of the letters without diacritics corresponds more or less to their IPA equivalents, except for /k̂p/ written <p> and /ĝb/ written <gb> (see Table 4), in which both consonants are pronounced simultaneously rather than sequentially. In addition to the underdots, three further diacritics are used on vowels and syllabic nasal consonants (see Table 3 below) to indicate the language's tones: an *acute accent* <ˊ> for the high tone, a *grave accent* <ˋ> for the low tone, where the middle tone is often left unmarked.

Table 3: Yoruba nasal sound samples sourced from [47]

Sound	Phonemic Representation	Sample
Ìtǎ́n (history)	/ã̃/	<b>Bran</b>
hě́n (yes)	/ɛ̃/	<b>Hen</b> ; French <i>vin</i>
rín (to walk)	/ĩ/	<b>Mean</b> ;
ìbǒ́n (gun)	/õ̃/	<b>Pond</b>
ràkṹnmí (camel)	/ũ̃/	<b>Soon</b>



Table 4: The unique letters of the Yoruba language sourced from [1]

Letter(Grapheme)	Phonemic Representation	General notes and pronunciation
P (as in "Pe" → "call")	/k̄p/	These do not occur in any European language. Each is a single sound unit. To pronounce this letter you must close your velum as if you want to say /k/ and then release at your lips for /p/ - aim to try and pronounce /k/ and /p/ at the same time. Same method applies to the pronunciation of /ḡb/
Gb (as in "Gba" → "take")	/ḡb/	-

### Lithuanian Language

Like the Yoruba language, the Lithuanian language uses the Latin script supplemented with diacritics. However, grammatically, the language is highly inflected. In Lithuanian, there are two grammatical genders for nouns (masculine and feminine) and three genders for adjectives, pronouns, numerals and participles (masculine, feminine and neuter). Every attribute must agree with the gender and number of the noun. The neuter forms of other parts of speech are used with a subject of an undefined gender (a pronoun, an infinitive etc.).

Consonants:

B C Č D F G H J K L M N P R S Š T V Z Ž  
b c č d f g h j k l m n p r s š t v z ž

Vowels:

A Ȧ E Ė È I Į Y O U U̇ Ū  
a ȧ e ė è i į y o u u̇ ū

In addition to its standard 32-letter alphabets, the following digraphs are used, but are treated as sequences of two letters for collation purposes: ⟨Dz⟩ → /Dz/, ⟨Dž⟩ → /dʒ/. Writing in Lithuanian is largely phonemic, i.e., one letter usually corresponds to a single phoneme. There are a few exceptions: for example, the letter ⟨i⟩ represents either the vowel /i/, as in the English word "sit", or is silent and merely indicates that the preceding consonant is palatalized. Table 5 contains samples of pronunciation of each Lithuanian letter.

Table 5: Lithuanian grapheme-IPA mapping

Consonants	IPA	English Approximation
<b>b</b> utas	b	<b>b</b> oot
<b>d</b> u	d	<b>d</b> o
<b>dz</b> ūkas	dz	<b>ad</b> ze
<b>dž</b> ar	ɟʒ	<b>j</b> eans
<b>f</b> abrikas	f	<b>f</b> olly
<b>g</b> alva	g	<b>g</b> ood
<b>h</b> alas	ɣ	<b>a</b> head
<b>j</b> auna	j	<b>y</b> ellow
<b>k</b> as	k	<b>K</b> elvin
<b>l</b> abas	l	H <b>al</b> lo
<b>m</b> ama	m	<b>A</b> merica
<b>n</b> amas	n	<b>A</b> nnex
<b>r</b> anka	ŋ	<b>S</b> ing
<b>p</b> adas	p	<b>p</b> anda
<b>r</b> atas	r	trill 'R' <i>like in Spanish</i>
<b>s</b> aulė	s	<b>s</b> oup
<b>š</b> aukštas	ʃ	<b>sh</b> op
<b>t</b> aras	t	<b>t</b> ap
<b>c</b> aras	ts	<b>c</b> ats
<b>č</b> aižus	tʃ	<b>ch</b> ip
<b>v</b> anduo	v	<b>v</b> isi
<b>z</b> aunyti	z	<b>z</b> oo
<b>ž</b> odis	ʒ	<b>a</b> sia
Vowel	IPA	English
<b>r</b> atas	a	<b>f</b> ather
<b>tę</b> sti	ɛ:	<b>p</b> et (but longer)
<b>r</b> etas	æ	<b>b</b> ad
<b>k</b> as	ɐ	<b>pu</b> tt
<b>tė</b> tė	e:	<b>f</b> airy
<b>m</b> esti	ɛ:	<b>m</b> et
<b>v</b> yras	i:	<b>n</b> eed
<b>k</b> itas	ɪ	<b>s</b> it
<b>rū</b> ta	u:	<b>m</b> oon
<b>b</b> utas	u:	<b>p</b> ot
<b>p</b> onas	o:	<b>d</b> oor

We have interchangeably used the words transcription, adaptation among others in place of transliteration in this section. Future in this research paper, we make conversant terminologies and concepts, review of related works is considered, algorithms for transliteration between our language pair is presented, results are given and analyzed and a conclusion is drawn in reference to the research aim.

## 3 Background

### 3.1 Terminologies

The word transliteration is an umbrella term for the adaptation of a foreign word to a specific language, however, there are several other terms that can be substituted for it for specificity. Romanization is the conversion of non-Latin scripts into Roman (Latin) Alphabet. When transliterating into a specific language, the transliteration process can be named based on the target language, e.g., Anglicization or Lithuanization. The latter, according to [20], often include adaptation to a more drastic degree than that implied, for example, in Romanization. In respect of the Lithuanian language, definition of terms used in Lithuanian when discussing the writing of personal and place names in other languages are stated in [36] and approved by The State Commission of the Lithuanian Language:

**Transliteration** – rewriting personal names (place names) written in non-Latin alphabets into letters (or as accurately as possible by rendering other written signs) in Lithuanian characters.

**Transcription** – recording a personal name (place name) of another language according to the approximate pronunciation in Lithuanian characters.

**Grammaticalization** – adaptation of a personal name (place name) of another language to the morphological system of the Lithuanian language by adding inflectional endings of the Lithuanian language.

Transliteration should not be confused with translation which allows words in one language to be understood by those who speak another language by representing its meaning in the target language and neither should it be mixed up with transcription - the systematic representation of spoken language in written form where the source can either be utterances (speech or sign language) or preexisting text in another writing system [7]. Transliteration does not tell the meaning of a word but how to pronounce it by converting the letters from one alphabet or language into the corresponding, similar-sounding characters of another alphabet.

Speaking of sounds and alphabets, i.e., phonetics and phonology (where phonetics is the actual production and perception of speech sounds by humans, and phonology, the systems of sounds in use in a particular language or context), the following linguistic features are notable in the course of this work:

**Phoneme** – In linguistics, a phoneme is the smallest unit of speech distinguishing one word (or word element) from another, such as the grapheme ⟨p⟩ in “tap” which separates that word from “tab” “tag” and “tan”. A phoneme may have more than one variant, called an allophone, which functions as a single sound; e.g., the p’s of “pat” “spat” and “tap” differ slightly phonetically, but that difference, determined by context, has no significance in English. In some languages where the variant sounds of /p/ can change meaning, they are classified as separate phonemes – e.g., in Thai the aspirated /p/ (pronounced with an accompanying puff of air) and unaspirated /p/ are distinguished one from the other.

**Grapheme** – A grapheme is the smallest functional unit of a writing system. It is impossible to speak about graphemes without mentioning phonemes as the two are so inextricably

linked. Some graphemes can carry the sound of a variety of different phonemes and the same is true vice versa. Some written scripts are simple in which 1 letter usually represents 1 sound. These letters and their corresponding sounds are consistent and transparent. For this reason, languages like Italian or Spanish are easy to read.

Table 6: The influence of pitch in the pronunciation and meanings of Yoruba words

Word	Translation
Ogún	Inheritance
Ogun	War
Ògùn	Drug

**Diacritics** – often loosely called “accents”, are the various little dots and squiggles which, in many languages, are written above, below or on top of certain letters of the alphabet to indicate something about their pronunciation [43]. Thus, Lithuanian has words like “Ačiū” - Thank you; “Šeštadienis” - Saturday; while Yoruba has “Adé” - Crown; “Lékelèké” - Cattle Egret. Accents in Yoruba words, however, do not only discern the pronunciation of words but also meanings. This feature is called tone. Tone is the variation in the pitch of the voice while speaking which serves to help distinguish words and grammatical categories – i.e., to differentiate one word from another word that is otherwise identical in its sequence of consonants and vowels [26]. For example, the word *inheritance*, in Yoruba, may have up to 3 different meanings depending on its pitch. See Table 6 above.

**Declension** – Declension is the changing of the form of a word, generally to express its syntactic function in the sentence, by way of some inflection and it is an important aspect of adapting a foreign word to suit the phonological requirements of the Lithuanian language.

### 3.2 Phonetic typography

As we might have observed so far, NLP is greatly influenced by linguistics, hence, the substantial use of phonetic typography in this work. The following phonetic notations are common across this paper:

**Slashes / ... /** – According to the International Phonetic Association, in [35], slashes are used for abstract phonemic notation, which note only features that are distinctive in the language, without any extraneous detail. Therefore, we utilized slashes to represent phonemes. For instance, coming across such notation as /p/ means that we are making reference to the actual sound produced by the phoneme written between the slashes and not a mere letter.

**Angle brackets < ... >** – The angle brackets and the alphabet contained therein are used to identify individual graphemes. Unlike slashes, they are used to indicate the IPA letters themselves rather than the sound values that they carry. Therefore, the notation ⟨a⟩ refers to a letter and not the possible sound such alphabet may produce. In the case of distinguishing between original orthography from transliteration, double angle brackets are used i.e., ⟨⟨...⟩⟩. Let’s say

we had just transcribed the Yoruba proper name "Pápá" to Lithuanian, "Kpakpa", the grapheme ⟨p⟩ maps to the original form of the word while ⟨⟨p⟩⟩ references the transliterated version i.e., ⟨kp⟩.

**Square brackets [ ... ]** – In this paper, square brackets are used in scenarios where the context, loosely speaking, position, of a letter is being examined amongst adjacent letters. In recent example of the transcription of the word "Pápá", writing such notation as [A] would mean that we are checking the position of ⟨a⟩ in the word in order to determine the correct sound it produces in the word.

## 4 Techniques and related works

Various automatic machine translation techniques have been adopted with each having its own advantages and disadvantages. Three paradigms have dominated machine translation. In temporal order, they are rule-based machine translation (RBMT), Example-based machine translation (EBMT), and statistical machine translation (SMT). They differ in the way they handle the three fundamental processes in machine translation: analysis, transfer, and generation. In its pure form, RBMT uses rules, while SMT uses data, i.e., examples of parallel translations. EBMT tries a combination: data supplies translation parts that rules recombine to produce translation. In their work, [14] states that the paradigms mentioned above are the major approaches to Machine Translation where SMT takes a larger number of chapters as it is said to be the leading paradigm.

### 4.1 Rule-based Machine Transliteration

A rule-based machine transliteration system consists of collection of rules called grammar rules, lexicon and software programs to process the rules. Rule based approach is the first strategy ever developed in the field of machine transliteration - [21]. RBMT relies on countless built-in linguistic rules for each language pair. The algorithm parses text and creates a transitional representation from which the text in the target language is generated. This process requires extensive lexicons with morphological, syntactic, and semantic information, and substantial sets of rules. The algorithm uses these complex rule-sets and then transfers the grammatical structure of the source language into the target language. Also known as handcrafted/knowledge-based machine transliteration method, RBMT requires considerable human linguistic knowledge to construct a fairly efficient set of bilingual rules that will generate transliteration from source to target. As with many other transliteration techniques, there is no one-direction to come about a RBMT algorithm, making it more unopinionated - especially in this case where it is human-knowledge-intensive.

[20], in their work, highlighted some of the challenges that face the handcrafted rule method where it is relatively daunting to generate transliteration for English proper nouns made up of two or more words e.g., the proper noun LONG ISLAND. To work around this concern, the words formulating the noun are concatenated and subsequently adapted to their target word - where Lithuanization occurs at the rear of the noun i.e., LONG ISLAND → LONGAILANDAS. In their algorithm, two analogous pair of sets of English-Lithuanian words containing several examples to illustrate possible transliteration for each word pair were excerpted from [29] and [28], respectively. Making use of these pair of data sets, a new format of data records were generated with the following parameters:

- The current letter of the English word, the left context of the current letter (6 letters to the left of the current letter; if the context was narrower than 6 letters, underscores were added)
- The right context of the current letter (6 letters to the right of the current letter, if the context was narrower than 6 letters, underscores were added).
- The string of Lithuanian letters corresponding to the current letter.

The algorithm has an accuracy of up to 59 percentage of word accuracy.

### 4.1.1 Context-sensitive and Context-free

Context-sensitive and context-free rules are two types of rules used in rule-based machine transliteration. Context-free rules are rules that apply uniformly to a particular character or sequence of characters regardless of their position in the word. In relation to the Lithuanian language, one example of a context-free rule could be to transliterate the Yoruba letter ⟨e⟩ as ⟨⟨e⟩⟩ and the letter ⟨o⟩ as ⟨⟨o⟩⟩ in all positions. Another example of a context-free rule could be to transliterate the Yoruba letter ⟨y⟩ as ⟨⟨j⟩⟩ in all positions, as this letter has a distinct pronunciation in Yoruba. In a study by [18], a context-free grammar-based approach was used for Persian-to-English transliteration. The approach used a set of phonetic and orthographic rules, including context-free grammar rules, to generate accurate transliterations.

In order to capture more complex patterns and variations in a transliteration process, the possible contexts of each character in a given word must be evaluated as there oftentimes are cases where some letters provide information about the pronunciation of other letters in the word. [38] uses the term *markers* for such letters. Letters may mark different kinds of information as demonstrated in our context-sensitive Yoruba-Lithuanian transliteration rule for the Yoruba alphabet "A":

[A] → ⟨A⟩ (e.g. IYA → IJA) | Mother

[A]N → ⟨O⟩ (e.g. IYAN → IJON) | Pounded Yam (food)

AN[A] → ⟨O⟩ (e.g. IYANA → IJANO) | Place

Based on the context of the letter ⟨A⟩ in the examples, the correct sound is established. The *current* [A] letter (the marker) in the last example informs that the nasal ⟨â⟩ sound that should have been applied to the preceding ⟨AN⟩ must be nullified. In the penultimate example, however, the nasal sound applies given that there were no markers to manipulate it as in the third example which makes the letter in scope context-sensitive. The first example is context-free.

### 4.1.2 Syllabification

Syllabification is an important aspect of machine transliteration. In many cases, the target language may have a different syllable structure than the source language, which can pose a challenge for accurate transliteration. Several studies have explored the use of syllabification rules and techniques in machine transliteration. For example, researchers have developed algorithms that use knowledge of the syllable structures of both the source and target languages to generate candidate transliterations. These algorithms often rely on linguistic resources such as pronunciation dictionaries and syllable templates to identify the optimal syllable boundaries for a given word or name.

The principal purpose of breaking down a word into meaningful units in a transliteration process is to determine the structure of a word, which in turn affects its pronunciation, stress, and rhythm. [6], in their work on "Thai word segmenter", utilized syllabification to locate word boundaries and extract useful features which enabled them to capture language patterns in their devising of the word segmenter. Similar technique is used in [48], where they broke down English names into a sequence of syllables and generated the most probable Pinyin sequence with the mapping model of English syllables to Pinyin, and then converted the Pinyin sequence

into a Chinese character sequence with the mapping model of Pinyin to characters (PC model) to retrieve final Chinese character sequence.

Other research works have demonstrated how RBMT can be used with statistical machine translation methods. To examine RBMT from a SMT viewpoint, [14] modeled each stage of their proposed **analysis**  $\rightarrow$  **transfer**  $\rightarrow$  **generation** (ATG) process where each of these three sub-processes can be purely rule or purely data driven, or may be a combination. During **analysis**, RBMT would use rules of morphology analysis, parsing, semantics generation, and so on. However, it will have to grapple with ambiguities such as lemma ambiguity, morphological features ambiguity, named entity etc (which SMT deals with through the use of probability). During **transfer**, RBMT would perform bilingual dictionary lookup for word and phrase mappings. If the analysis stage has been successful in complete disambiguation, this stage is easy, provided the bilingual dictionary is rich enough to record all words, named entities, and multiword mappings. During **generation**, RBMT would typically arrange for morphology synthesis of lemmas that came from bilingual dictionary lookup, and perform syntax ordering, i.e., place words and phrases in positions licensed by the syntactic rules of the target language. Seemingly, generation (G-stage) is an easier proposition than the A-stage. But this is not completely true. If RBMT must maintain native speaker acceptability of translation, then generation also can be complicated.

## 4.2 Statistical Machine Transliteration

This paradigm relies on example of transliteration, the so-called parallel corpora. Statistical-based transliteration approaches tend to be computationally easier in language transliteration than trying to parse and evaluate grammatical rules and it employs various mathematical techniques. Following are the 3 major phases in an SMT system:

**Alignment** - An important component of any SMT system is the alignment of words. Word alignment is a mapping between the words of a pair of sentences that are a translation of each other. The most popular alignment methods are IBM Model 1, Model 3 [5], and the hidden markov model (HMM) [45][42]. Although word alignment and SMT in general are machine translation components, machine transliteration systems have equally benefited from this constituent, performing it at the character level as performed by [42] where an *n-gram* model was adopted to achieve a phonemic alignment of word pairs. Now, an n-gram is a type of probabilistic language model for predicting the next item in a sequence in the form of a (n - 1)-order Markov model.

**Model Training** - Training and evaluation of transliteration systems require a bilingual corpus of source words and their transliterations. A bilingual corpus B is the set  $\{(S, T)\}$  of transliteration pairs, where  $S = s_1..s_l$ ,  $T = T_k$ , and  $T_k = t_1..t_m$ ;  $s_i$  is a letter, logogram, or symbol in the source language alphabet, and  $t_j$  is a letter, logogram, or symbol in the target language alphabet - [19]. The training phase is usually one to follow the alignment stage and for training a model, a list of considerable number of proper nouns in the source language and their representation in their target language counterpart are collected to form a training set.

**Model Application:** According to [13], GIZA++ (extensive description of this toolkit can be found in [30]) is utilized to implement SMT alignment models of Model-1 through Model-4, where training is typically carried out unsupervised to iteratively improve alignments between



parallel phoneme sequences. Each phoneme from the source language is aligned to only one symbol of the target. In the event that a source character has no alignment with its pair it has a *fertility* of NULL. If it remains unaligned to any source phoneme, it is known as *zero-fertility*. The same approach applies when the target language serves as the source - e.g., if a Lithuanian phoneme is left unaligned, it is called *NULL-generated*;

In the remainder of this section, we survey models and techniques used in statistical machine translation while closing the segment by drawing comparisons among the reviewed models.

#### 4.2.1 Noisy Channel Model

Noisy channel model (NCM) is the basic phrase-based statistical machine translation model. The noisy-channel model is a machine translation technique which was initially developed for the translation of French sentences, by [5], tries to capture how source names can be mapped to target names. In NCM, the source language to target language letters are aligned using GIZA++ [23]. Every letter is treated as a single word for the GIZA++ input. The alignments are then used to learn the phrase transliteration probabilities which are estimated using the scoring function given in [24].

#### 4.2.2 Finite State Transducers (FST)

Finite State Transducers are models that are being used in different areas of pattern recognition and computational linguistics. In the area of machine transliteration the transducer-based approaches that are based on building models automatically from training examples are becoming more and more attractive. A transducer has the intrinsic ability of *transducing* or transliterating. Whenever the machine shifts from one state to another, it will print the output word, if any. So, as a result, not only will it accept the sentence of one language, but it will print the transliteration in another language. Alternatively, a transducer can be seen as a bilingual generator. A FST is an automaton that transforms one string into another. It can be seen as a network of states with transitions between them which are labeled with input and output symbols. Starting at some state and walking through the automaton to some end state, the FST can transform an input string by matching the input labels to an output string by printing corresponding output labels [26]. [22] modeled Japanese-to-English transliteration with weighted finite state transducers (WFSTs) by combining several parameters including romaji-to-phoneme, phoneme-to-English, English word probabilities, and so on. A similar model was developed for Arabic-to-English transliteration [41]. [Meng] proposed an English-to-Chinese transliteration method based on English grapheme-to-phoneme conversion, cross-lingual phonological rules, mapping rules between English phonemes and Chinese phonemes, and Chinese syllable-based and character-based language models.

#### 4.2.3 Hidden Markov Model (HMM)

Markov model is a probabilistic function of Markov process which was first developed by Andrei A Markov in 1913 for modeling the letter sequences in Russian literature - [11]. The HMM which is commonly used for speech recognition but has gained popularity in machine translation and transliteration. "The HMM function is one of the powerful statistical probability tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence" - [8].

There are several well-known algorithms for hidden Markov models. For example, given a sequence of observations, the Viterbi algorithm (a dynamic programming algorithm for obtaining the estimate of an unknown quantity) will compute the most-likely corresponding sequence of states [44]. Mathematical model of HMM is described in the paper of [37] for speech recognition and then extended for translation and transliteration. In their survey, [8] highlighted existing transliteration works based on the HMM Model.

### 4.3 RBMT vs SMT

While several machine transliteration techniques exist and continue to both grow in number and improve, they all share tantamount objective - generate a word in another language with utter precedence to pronunciation. It is, however, a fact that each of these models do not perform at equal levels nor generate a result with equal accuracy. For instance, the result of a grapheme-based transliterated word (GT) may differ from that of a phoneme-based (PT), where one may offer better accuracy than the other. In their analysis of the contrasting difference in word-generation between both of these unit-level transliteration methods, [32] highlighted how the transliteration result of the medical word *amylase* differed. The standard Korean transliteration of the test word is 'a-mil-la-a-je', which GT tends to produce correctly whereas PT is disposed towards producing wrong ones like 'ae-meol-le-i-seu', which is derived from /AE M AH L EY S/, the pronunciation of amylase. In contrast, PT can produce 'de-i-teo', which is the standard Korean adaptation of data and a PT transliteration, while GT tends to give a wrong one, like 'da-ta'. This issue is typical of some of the problems found in other models, hence, the cause for the utilization of a combination of models.

The rule-based machine transliteration method can be modeled around both direct and pivot approaches and may be combined with a SMT approach. Using the former requires considerable human linguistic knowledge, thus, does not offer high accuracy rate considering the reality that there is no given convention or pattern to pin-down human naming conventions across different cultures which is one of the causes for machine transliteration in the first place. The common RBMT comparison is often with its contrasting counterpart, SMT/Machine Learning (ML), and the popular consensus amongst researchers is that in many of the areas where RBMT falters (e.g inconsistency, problems with ambiguity), SMT excels. This, however, is not to say that the statistical approach does not have its own struggles.

While RBMT, on the one hand, requires considerable human efforts such as constructing the rules, adapting those rules to complement direct or pivot approaches (whatever the case may be), it has proven sufficient in consistently transcribing words using the context-free and context-sensitive approach from 31 languages (German, Greek, Hungarian, Icelandic, Italian etc.) to the Lithuanian language [3] which is why notwithstanding the advent of machine learning methods for transliteration, the rule-based approach maintains relevance. SMT, on the other hand, requires copious amount of data, hence, causing exertion at the alignment stage - which is considered the most integral of SMT phases as it determines the accuracy of word generation. Its rigors, however, afford for its supposed high transliteration accuracy rate.

## 5 Transliteration Routes and Data

### 5.1 Routes

#### 5.1.1 Direct

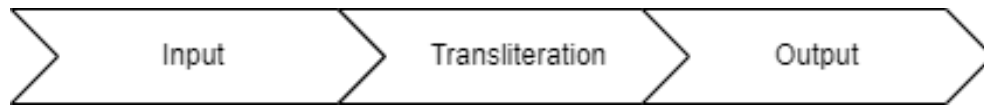


Figure 2: Direct route

The direct route, as its name suggests, is simply the transliteration of a given proper noun in the Yoruba language to Lithuanian without the employment of an external factor in the adaptation procedure. External factors, in this parlance, would mean the use of a language resource such as the international phonetic alphabet, to interpolate the transliteration process with with an attempt to improve word adaptation.

#### 5.1.2 Intermediate

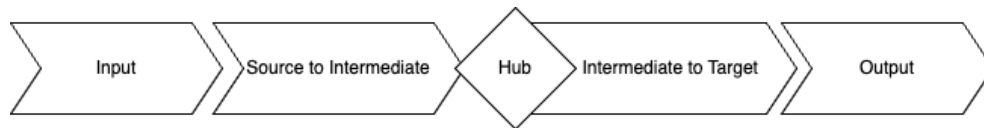


Figure 3: Intermediate route

In this route, a language resource is introduced at the hub of the transliteration process, which acts as an intermediate step. There are two transliteration phases at the intermediate level: transliteration from the source language to the intermediate language; transliteration from the intermediate language to the target language. In the former, the source language is transliterated into the intermediate language using specific transliteration rules and techniques. These rules and techniques are designed to handle the phonetic and orthographic differences between the source and intermediate languages. By converting the source language into the intermediate language, we create a bridge that helps to facilitate the adaptation process. Transliteration from the intermediate language to the target language is similar to what has previously been done, however, this stage leverages the linguistic and phonetic similarities between the intermediate and target languages in attempt to achieve more accurate transliteration.

In the intermediate route, 2 mechanisms were employed at the hub namely Languages and IPA. Let's take a look at them:

#### IPA

The IPA was devised as a standardized representation of speech sounds in written form, making every sound that serves to distinguish one word from another in a language have its distinctive IPA representation. A simple way to retrieve an exemplified IPA representation for any language

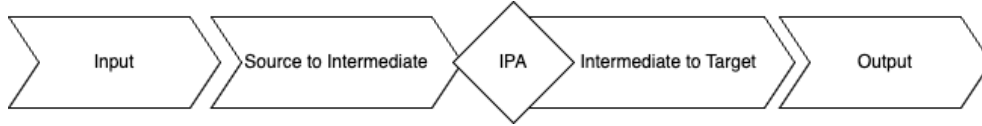


Figure 4: IPA at the hub

is via the Wikipedia uniform resource locator specific for IPA: <https://en.wikipedia.org/wiki/Help:IPA/Lithuanian> where replacing the language path in the domain with a valid language returns the IPA for that language with pertinent pronunciation examples. The tasks completed during the pre-processing stage are similar but different in concept when IPA is at the hub than when a language is: with IPA, tokenization of the input requires syllabification, and alignment involves the mapping of the resulting tokens from the syllabifier to their respective phonetic alphabets. The tasks performed at the hub is demonstrated in the Table 7 below. During post-processing, the graphemes of the target language are mapped to their respective letters.

Table 7: Demonstration of the character transformation process in the IPA model where SG = Source Grapheme, SP = Source Phoneme, SIPA = Source IPA, TIPA = Target IPA, and TG = Target Grapheme.. Let's name this table a matrix  $\mathbf{M}$  for reference.

No.	Input	Process	Output
1	SG	SG $\rightarrow$ SP	SP
2	SP	SP $\rightarrow$ SIPA	SIPA
3	SIPA	SIPA $\rightarrow$ TIPA	TIPA
4	TIPA	TIPA $\rightarrow$ TG	TG

Getting the right input at  $M_{11}$  is integral to getting the correct output at  $M_{33}$ . Though the task of tokenizing characters as graphemes is as simple as *splitting* a given word, more is required than text-splitting when considering the Yoruba nasal sounds which are activated only when certain letters are used in combination, hence, the importance of retrieving the correct SP given a letter's context. To solve this problem, we implemented a syllabification algorithm based on the Yoruba language syllable structure which helps to trap these letter combinations to ensure the derivation of the right IPA value.

### Languages



Figure 5: Language at the hub

As for languages, the English and Georgian languages were utilized. The latter offers a

different writing system than all other mechanisms employed at the hub and is therefore interesting to experiment with an inter-script system, whereas the former offers the benefit of finding a resource that may enhance our adaptation result as we will find out later on in this section, but first let us take a look at how a language work differently than the IPA in the system. Every other component work the same as previously described in the IPA segment above except the language method needs to compile two different language rules at the pivot: for instance, the system must compile Yoruba-Georgian transliteration rules and then Georgian-Lithuanian rules.

## 5.2 Data

We have 4 distinct sets of analogous datasets with 928 entries prepared for both of our transliteration routes. Yoruba proper nouns, including their phonetic alphabet representation, were sourced from Wikipedia, while the transcription equivalence of each of those proper nouns to their respective target languages depended on human knowledge to make a corpus. These datasets were then divided such that we have a group containing 500 entries for training (we called it **xTrain**) and another group of 428 entries for testing (which we called **yTest**). In Anglicizing our Yoruba data, we sought for letters and digraphs that has the tendency to consistently spell a phoneme the same way in varying contexts, given the propensity of English letters to offer different pronunciations in different positions. For instance, the Yoruba name "Balogun"

Table 8: Sample data for our algorithm learning

Yoruba	Lithuanian	English	Georgian	IPA
Aje	Adžai	Ajei	აჯე	ã.ɟé
Ajewọle	Adžaivolai	Ajeiwaulei	აჯევოლე	ã.ɟé.w:ã.lé
Ajike	Adžike	Ajikeh	აჯიკე	ã.ɟí.ké
Akanke	Akonke	Akonkeh	აკონკე	ã.kã.ké
Balogun	Balogunas	Balagoon	ბალოგუნ	ba.ló.gũ
Bibeli	Bibilis	Bibail	ბიბელი	bí.bé.li
Bilisi	Bilaisas	Bilis	ბილისი	bì.lí.si
Bọ̀sẹ̀de	Bosedai	Basedai	ბოსედე	bɔ.sɛ.dé
Epo	Aikpo	Eikpo	ეპო	è.kpò
Eṣu	Aišu	Aishu	ეშუ	è.ʃũ
Efunṣetan	Efunšaitonas	Efoonshaiton	ეფუნშეთონ	ɛ.fũ.ʃe.tã
Egbado	Egbadau	Egbadoe	ეგბადო	ɛ.ɡbã.dò
Eyo	Ejo	Eyau	ეიო	ɛ.jũ
Gẹ̀ṣi	Gęsis	Gessi	გეესი	gɛ́.ɛ́.sí
Ibadan	Ibadonas	Ibadon	იბადონ	ì.bã.dã
Ifon	Ifonas	Ifon	იფონ	ì.fũ
Iṣẹ̀ṣe	Išešai	Isheshei	იშეშე	ì.ʃɛ.ʃe
Lagunna	Laguno	Lagunno	ლაგუნო	lá.gũ.nã
Modakeke	Modakeke	Maudakeke	მოდაკეკე	mã.dã.ké.ké
...	...	...	...	...
Yoruba	Jauruba	Yoruba	იორუბა	jo.rũ.bã

is transcribed as "Balagoon" in English because the chances are quite strong that an English speaker would pronounce the word as so: /ba.lo.gʌn/ given the position of /un/ as in "fun", "gun", "bun" etc., and this will lead to a wrong pronunciation of the word in its source language. Hence, given such context, we proffer the digraph /oo/, and /u/ in contexts with no utterance ambiguity. Lithuanization of the source data, however, requires a more drastic degree of adaptation - we therefore aligned the source data with its Lithuanian equivalence while applying declension to words that are obvious such as:

Ibadan → Ibadonas  
Efunṣetan → Efunšaitonas

Having a corpus alone is not sufficient for transliteration tasks without removing "noises" ahead of learning our data to a decision tree classifier or proceeding with other different phases of the RBMT system. While there is no cause for worrying about eliminating noises like stop words, transformation of words to their base forms through lemmatization among other machine translation text-preprocessing techniques, tokenization and normalization of our data is necessary. Alignment of a word at character-level will be impossible without the conversion of each member of a textual data into tokens. Tokens are entities present in a textual data, while tokenization is the process of converting chunks of a textual data into tokens. Tokenizing a word sample in our algorithm is a fairly straightforward process which involves running an iteration over the word in question where each round of iteration tokenizes members of the textual object. This process is true for both routes and intermediate resources utilized except for the IPA because there is strong probability that the Yoruba nasal sounds are neglected. To solve this problem we devised a syllabifier to detect the context of each part of a given word to translate its sound.

## 6 Rule-based Transliteration

### 6.1 Syllabification

Understanding of basic syllabic structure of a language is sufficient for the making of a program that breaks a word into smaller parts. In linguistics, a syllable template is a type of abstract representation that describes the internal structure of a syllable in terms of its constituent phonemes. A syllable template specifies the positions and types of phonemes that can occur in a syllable, as well as the patterns of stress and tone that are associated with each syllable. In some languages, syllable templates can be quite complex and may involve the combination of multiple consonants and vowels in a single syllable like Lithuanian's (((C)C)V(V)(R)(C(C)))[27]. According to [33], Yoruba has 3 syllable types C, V, and the syllabic nasal N with a typical template of (CV) where C represents a consonant and V represents a vowel. For example, the word "Ogún" meaning "inheritance" has two syllables, "o" and "gún", which follow the CVC template. In the first syllable "O", is a vowel sound with no accent, indicating a middle tone. In the second syllable "gún", is a consonant cluster of /g/ and the nasal /n/ and "ú" is the vowel sound with a high tone. The following syllable directives are true of any language and were taken into consideration in the devising of our syllabifier:

- Every word has at least one vowel.
- One vowel sound is heard in each syllable and each syllable type is related to the sound the vowel will make.
- Vowel Lengthening: When a syllable contains a long vowel or a vowel followed by a certain combination of consonants, it is often considered as a separate syllable.
- Stress and Tone: Syllabification can also be influenced by stress patterns and tone assignment in certain languages. Stressed syllables may have different characteristics or allow different structures compared to unstressed syllables. Tone languages assign pitch patterns to syllables, which can affect their syllable structure and pronunciation.

---

**Algorithm 1** Algorithm for the syllabification of Yoruba words

---

**Require:** Yoruba proper noun

**Ensure:** Tokens (Syllables of Yoruba proper noun)

```
1: syllables ← list
2: word ← input
3: wordLength ← count(word)
4: procedure ASSERTSYLLABLE(start, end)
5:   token ← word[start : end]
6:   syllables[syllablesLength] ← word[start : end]
7:   word ← word[end :]
8:   wordLength ← count(word)
9: end procedure
10: for i ← 0, j ← 1 do
11:   if word starts with a vowel then
12:     if nextLetter ← "N" then
13:       if NasalSounds[nextLetter + "N"] then
14:         AssertSyllable(i, j + 1)
15:         Reinitialize i, j back to initial.
16:       else
17:         AssertSyllable(i, j)
18:         Reinitialize i, j back to initial.
19:       end if
20:     else
21:       AssertSyllable(i, j)
22:       Reinitialize i, j back to initial.
23:     end if
24:   else
25:     if Word starts with a digraph then
26:       if count(wordWithoutConsonant) > 1 then
27:         if NasalSounds[digraph + subsequentLetters] then
28:           AssertSyllable(i, j + 1)
29:           Reinitialize i, j back to initial.
30:         else
31:           AssertSyllable(i, j)
32:           Reinitialize i, j back to initial.
33:         end if
34:       else
35:         AssertSyllable(i, j)
36:         break;
37:       end if
38:     else
39:       if count(wordWithoutConsonant) > 1 then
40:         if NasalSounds[currentLetter + nextLetters] then
41:           AssertSyllable(i, j + 1)
42:           Reinitialize i, j back to initial.
43:         else
44:           AssertSyllable(i, j)
45:           Reinitialize i, j back to initial.
46:         end if
47:       else
48:         AssertSyllable(i, j)
49:         break;
50:       end if
51:     end if
52:   end if
53: end for
```

▷ e.g "IN"

▷ word starts with a consonant

▷ e.g "GBǪN"

▷ e.g "GBA"

▷ Word starts with a single consonant

▷ e.g "KÁN"

▷ e.g "KA"



Table 9: Performance check on the syllabification algorithm by manual evaluation

Category	Word Count	Correctness %
Proper nouns	600	98
Adjectives	856	90

The table above shows that the algorithm is highly able to break down proper nouns and while its performance with other figures of speech, such as adjectives, is also high, we noticed test cases that were wrongly broken or not broken at all which is as a result of the usage of word contraction in such words e.g., "ÓHÚN KÁN" (meaning 'something') → "NKAN" leading to consonant clusters which are not primarily a part of the syllabic template. A work around this issue is to apply consonant cluster rules into the algorithm. There are 4 identifiable parts in the syllabifier as demonstrated in the Figure 6 below. Let us take a look at each of them:

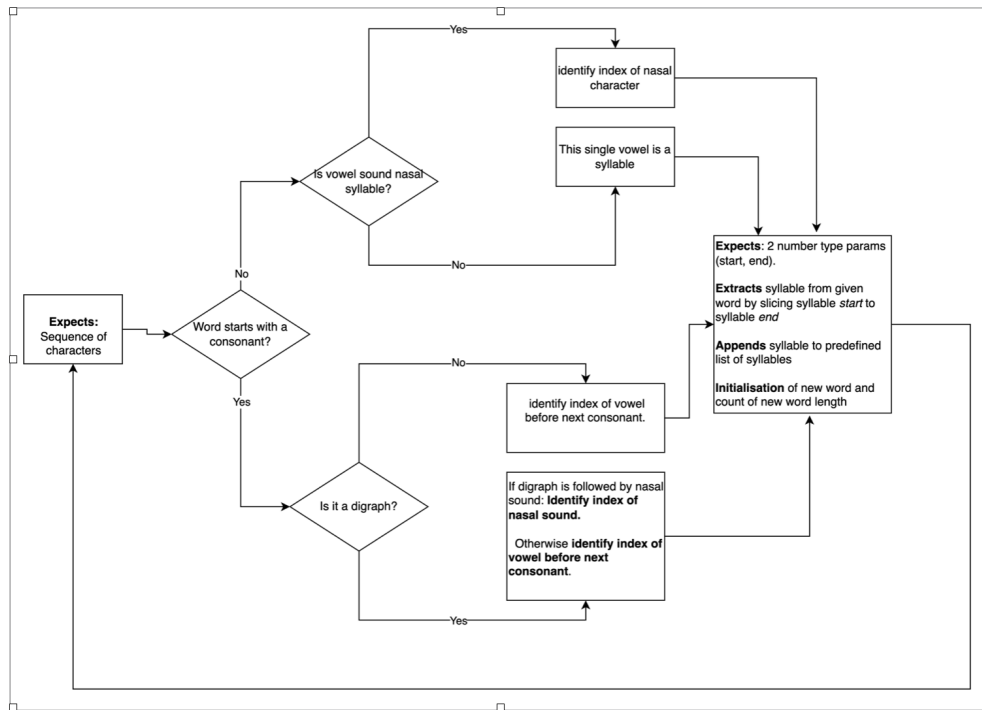


Figure 6: Flow diagram for algorithm 1

**Input** - At the input stage, the algorithm expects a pre-processed proper name.

**Type of the first letter** - Identifying the type of letter (i.e., consonant or vowel) starting a Yoruba word is important in fulfilling the objective of detecting the first syllable in such word. The first vowel starting a proper name is oftentimes an independent syllable e.g., ⟨O⟩ in "Ọba" (meaning, "King") is the first syllable of two. Whereas a consonant starting a word is an indicator that such consonant letter is only the starting point for finding a syllable.

**Syllable finder** - Different checks are made in this phase depending on what letter starts the word, but in any case, the presence of a nasal sound in the potential syllable is checked. In the case of a vowel, the next letter to the vowel is examined for nasality (previously described in the Table 3) and if the result of this check returns a truthy, e.g., ⟨IN⟩ i.e., /i/ in "INGILANDI" (meaning, "England"), such vowel is no longer an independent syllable, therefore, the position of the nasal consonant is identified for syllable extraction. If however, the nasality of the syllable is falsy, a check for the possibility of the vowel being a long sound is checked e.g. ⟨AA⟩ i.e., /a:/ in "AARE" (meaning, "commander", as in chief officer) - the truthiness of this check finally determines whether the vowel starter is independent or not thereby providing the range of the syllable. The first check made when finding a syllable in a consonant-starter word is to recognize the possibility of the starter consonant being the digraph ⟨gb⟩. Either case where it is a digraph or not, the nasality and long vowel sound check is made. If both nasal and long vowel examination results to be false, index of the vowel before the next consonant is identified for syllable extraction.

**Syllable extractor** - This component expects 2 integer arguments as a requirement to function: start and end. These arguments specify the range of the syllable to be extracted - i.e., the starting and the end of the syllable. For instance, if the algorithm were to be processing the word "GBỌNGAN", a town in Oşun state, Nigeria, the correct argument to be received in order to extract the first syllable in the sample word would be 0 and 4 where the former argument is the starting position and the latter the end, excluding the letter in position 4 due to the zero-based way of indexing a sequence in computer programming. With a syllable's range within a word now known, such syllable (in this example, "GBỌN") is *sliced* and appended to a predefined variable which stores syllable tokens whereas the remainder of the sliced word, "GAN", becomes the *new* word to syllabify. This iterative process continues until the last syllable is extracted, where generated tokens become the final output:

GBỌN GAN

## 6.2 Alignment

### 6.2.1 Alignment for Languages

Our character to sound alignment algorithm, described in [20], generated a further 1176 data among direct and intermediate routes (majority of which came from English being at the hub). In what is an iterative process, the algorithm takes the character on the *current* iteration making it the focal point of the alignment process and then weighs on the context of that character 6

letters to the left and 6 letters to the right (where underscores were added in positions where the character’s context is narrower than 6 letters) to determine the alphabetical equivalence of the grapheme in question to its target language with priority to sound. Four actions are taken in the course of character-context assessment:

MATCHING	Of a grapheme, i.e., $x = x$
INSERTION	Of a single character. If $s = ct$ then inserting the character $x$ produces $cxt$
REPLACEMENT	Of a single character $x$ for a symbol $y \neq x$ changes $cxt$ to $cyt$ i.e., $x \rightarrow y$
SKIPPING	Of a single character, i.e., $x \rightarrow \varepsilon$ where $\varepsilon$ = empty string.

The successive writing of a letter in the Yoruba language, is generally used to denote stress on a word and oftentimes, on a borrowed word’s syllable. Take for example, the Yoruba word ”Geḡesi” (meaning, ”English”). The alignment procedure performs all actions in a successful transcription of the word into Lithuanian language

Table 10: Yoruba  $\rightarrow$  Lithuanian adaptation

Left Context	Current Yoruba Letter	Right Context	Target	Action
_____	[G]	EḡESI_	G	Match
_____G	[E]	ESI__	’ ’	Skip
___Gḡ	[E]	SI___	Eḡ	Replace
__GḡE	[S]	I_____	S	Match
_GḡES	[I]	_____	I	Match
GḡESI	[_]	_____	S	Insert

i.e.,

Geḡesi | GḡESIS

### 6.2.2 Alignment for IPA

The purpose of breaking a given word into syllables is to trap nasal sound existence in such word to afford for the correct mapping of a sound to its correct IPA value. For instance the syllabification output for the Yoruba word ”Baayanni” (meaning, a Yoruba deity), will be as follows:

BAA YAN NI

From the output, the position of stress in the syllables (1st) is identified and the presence of a nasal sound recognized - our alignment job, moving forward, is simplified.

Upon execution of **Algorithm 2** above, the sample word is aligned as follows:

---

**Algorithm 2** IPA Alignment algorithm

---

**Require:** Syllables

**Ensure:** Return IPA equivalent of tokens

YorubaIPAMappings

▷ IPAs mapped to Yoruba letters

**for** token in Syllables **do**

**if** token has nasal **then**

    transcribe preceding grapheme;

    map nasal vowels to IPA equivalent;

**else**

    map grapheme to IPA;

**end if**

**end for**

---

B	A	A	Y	A	N	N	I
B	-	A:	J	-	Ã	N	I

i.e.,

BAAYANNI | BA:JÃNI

## 7 Machine Learning

### 7.1 Decision Tree (CART)

CART, or Classification and Regression Trees, is a specific type of decision tree learning algorithm that was introduced by [4]. In this work we utilize the CART algorithm provided by [34]. CART algorithm creates a binary tree in which each node represents a feature, each branch represents a decision based on that feature, and each leaf node represents a prediction where a splitting criterion is used to determine how to divide the data at each node of the tree. Two popular criteria are entropy and gini impurity. Entropy is a measure of the amount of uncertainty or randomness in a set of data and is used to measure how well a split separates the data into homogeneous groups. The following formula may be used to compute the entropy of a dataset:

$$H(S) = -P_{(+)}\log_2(P_{+}) - P_{(-)}\log_2P_{(-)}$$

Where  $S$  is a set of data,  $P_{(+)}$  the percentage of positive classes and  $P_{(-)}$  the percentage of negative classes.

Gini impurity on the other hand measures the probability that a randomly chosen data point from a group would be incorrectly classified if it were randomly assigned a label from that group.

$$Gini(feature) = 1 - \sum (p(i|feature))^2$$

where  $p(i|feature)$  is the proportion of the samples that belong to class  $i$  for a given feature.

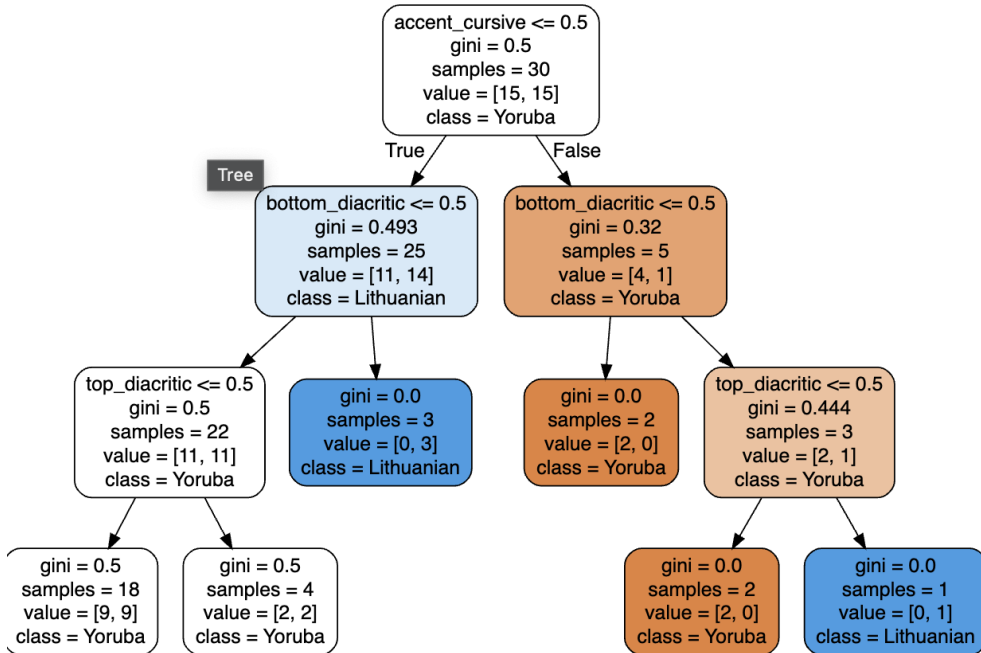


Figure 7: Classification with gini index

To exemplify the splitting process in a decision tree (DT), as in the Figures 7 and 8, we created a CART classifier and trained it on a simple dataset (with a prediction accuracy of 62%) that predicts whether a given proper name is Lithuanian or Yoruba with features described in the table below.

Table 11: Features used in training a CART to predict the source of a given word: Yoruba or Lithuanian

Feature	Meaning
accentCursive	Defines accent shape
bottomDiacritic	Position of accent beneath the word
topDiacritic	Position of accent at the top of the word
hasAccent	Specifies the presence of an accent on the word

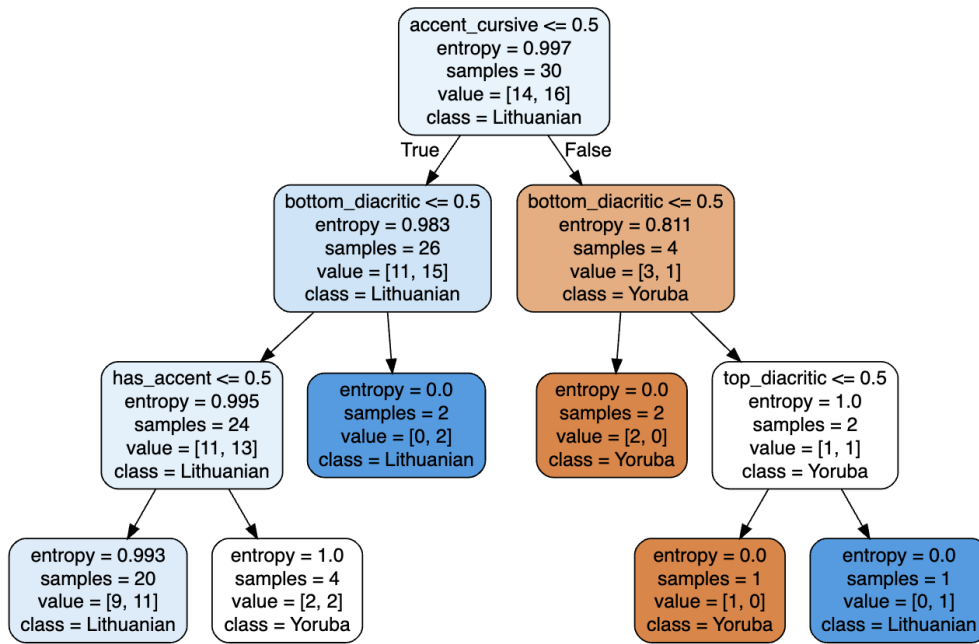


Figure 8: Classification with entropy

We asked the classifier to predict the origin of the words "Şegun" and "Şedurskas" using gini as the criterion for the former and entropy for the latter. Though the model made correct predictions for both words (Yoruba and Lithuanian, respectively) and with similar tree depth of 3 and decision leaves of 6, the difference in computational efficiency is noticeable. Gini computes twice as fast than entropy to reach a decision, with entropy ranging from 0 to 1 (where 0 means that all data points in dataset  $S$  belong to the same class (i.e., no uncertainty), and 1 means that the data points in  $S$  are evenly split across all classes (i.e., maximum uncertainty)) whereas it only ranges up to 0.5 for gini index. This is due to the fact that computation of logarithmic calculations, as is the case with entropy, are naturally expensive, the benefit, however is that it

tends to create more balanced trees. On the other hand, gini index is mostly preferred due to its computational advantage which is noticeable when work is being done on datasets that are more complex and in greater volume. The gini index is the default parameter used in a CART algorithm which adds up to the reason why it is our chosen algorithm among others such as C4.5.

### 7.1.1 Alignment

Once the various alignments of our various transliteration routes and pairs were completed we had more data for the DT induction. The Yoruba-Lithuanian word alignment exemplified in the Table 10, for example, gives 6 mappings. So, we have 500 word pairs and 3698 data mappings for our training data. Machine learning algorithms can only work on numbers, they cannot understand labels. As such, we converted inputs into numbers using [34]’s label encoder. The categorical features (i.e., transliteration class) though were not encoded using this method because our categorical features have considerably more than two values, whereas nominal categorical features having more than two values may get treated as ordinal ones by the machine learning model. Although model performance won’t suffer much, it is recommended to use One-Hot encoder (dummy encoding), also available in [34], for categorical features having more than two different types of value.

```
In [32]: xTrain
Out[32]:
```

	lcont3	lcont2	lcont1	lprev	lcurrent	lnext	le_rcont1	le_rcont2	le_rcont3
0	0	0	0	10	0	0	0	0	0
1	0	0	0	0	0	5	0	0	0
2	0	0	0	3	0	7	0	0	0
3	0	0	0	5	0	7	0	0	0
4	0	0	3	6	0	7	0	0	0
...	...	...	...	...	...	...	...	...	...
166	0	0	0	12	7	0	0	0	0
167	0	0	0	7	7	0	0	0	0
168	0	0	0	0	7	7	0	0	0
169	0	0	0	0	7	5	0	0	0
170	0	0	0	0	7	0	0	0	0

171 rows x 9 columns

Figure 9: Jupyter notebook snippet of our xTrain data at its label encoding stage ahead of learning it to a decision tree classifier

For our character-context technique, the training data consists of 9 context attributes which are used to predict the transliteration generation (TG) of a current character among a string of word to transcribe. The root node is split by the context of the letter in question where the decision tree stops *growing* if the condition in the left leaf node is true of the context-freeness of the letter. If it is context-sensitive, a decision node is built to determine the correct TG of

such letter based on our previously mentioned context-rules and compiles an information gain to determine the best attribute for the next split.

### 7.1.2 Pruning

One of the drawbacks of the DT is its propensity to *overfit* data. This occurs when a model is too complex and fits the training data too closely - in other words, it continues to split data until it finds purity in the leaf node. So, even though the DT may almost perfectly perform on the training data, it will perform very poorly on unseen data. This phenomenon is called overfitting [15]. Our classifier is also not immune to overfitting as slight difference in performance is noticed when we *pre-pruned* the tree. But this problem, first of all, poses a pertinent question as to where and how the problem occurred in our learning and the answer to this is that the model noticeably creates extra layers for context positions that are empty, therefore, causing depth in the tree hierarchy. In the figure 10 below, we ran an experiment to get the adaptation of the Yoruba letter ⟨E⟩ to English i.e., ⟨AY⟩ from the name "ADELEKE", and got the correct transcription. The depth and quantity of the leaves, however, are evidence of overfitting and this is where the complexity of a model comes to play, for instance, comparing the previously presented word-source classifier and the transliteration model.

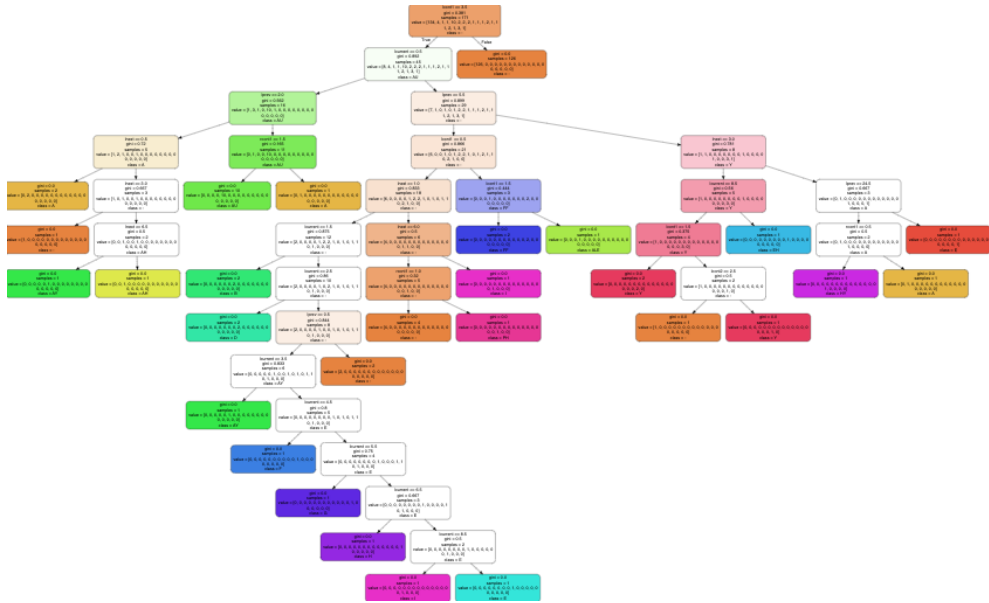


Figure 10: Visualization of the transcription of the Yoruba letter ⟨E⟩ to its English equivalent. The tree has a depth of 13 levels with 28 decision leaves

Since our approach is context-sensitive, the tree considers all contexts that may apply to a feature thereby finding the gini index for each of those feature until it attains leaf nodes with complete purity as validated in the figure 11 below.



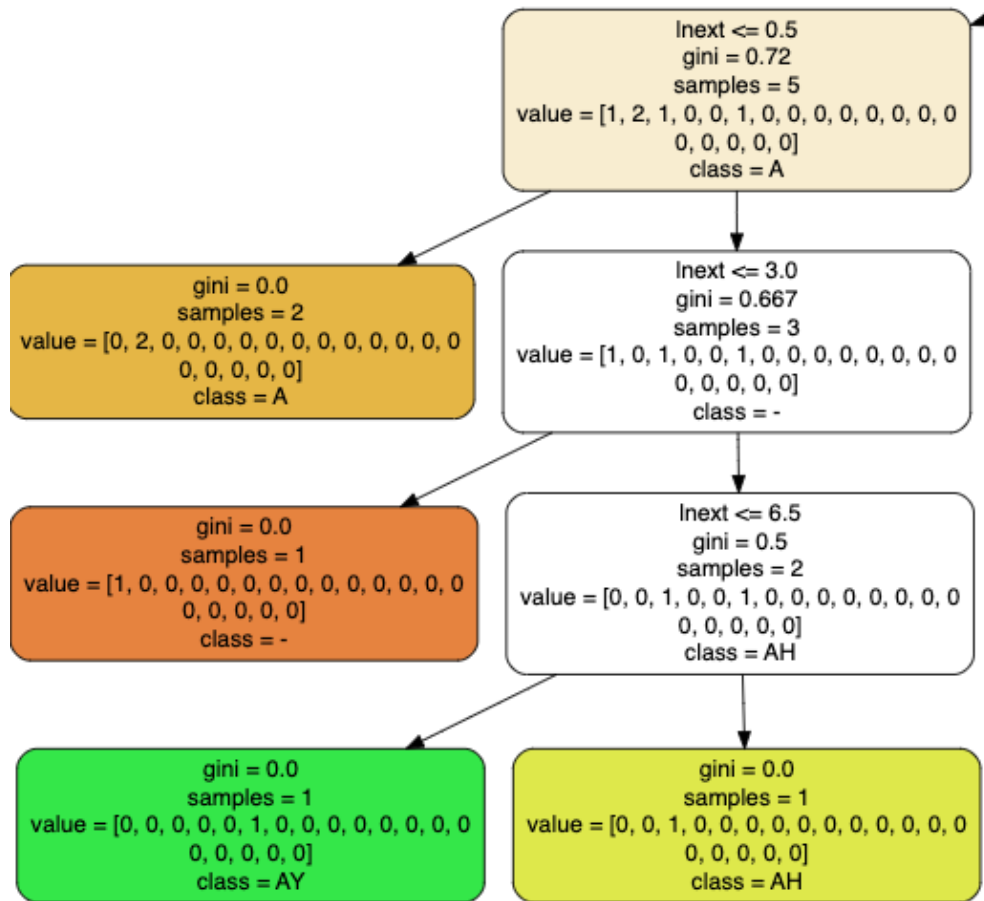


Figure 11: This diagram is cropped from the figure 10 above to show the decision made by the tree

There are two types of pruning: pre-prune and post-prune. The former is a technique used to eliminate unnecessary nodes of a tree prior to its construction and is generally done by hyperparameter tuning i.e., specifying additional criteria at the initialization stage of a classifier. While **post-pruning** is the prune of a DT after it has been constructed. In a study by [39], pre-pruning is said to be more efficient than post-pruning because it stops the tree from growing before it reaches its maximum size. Post-pruning, on the other hand, requires growing the tree to its maximum size before pruning back. There are several techniques used in pruning decision trees, including: reduced error pruning (removal of nodes that do not contribute to the overall accuracy of the tree), depth-based pruning (limiting the depth of the tree to reduce overfitting), hyper-parameter tuning (optimization of the values of the hyper-parameters that control the tree's growth and complexity) among others.

### 7.1.3 Hyper-parameter tuning

The process of hyper-parameter tuning typically involves selecting a range of values for each hyper-parameter that control the tree's growth and complexity and then training and evaluating the performance of the model with different combinations of these values [2]. One of the most common hyper-parameters to tune is the maximum depth of the tree, which controls the number of levels of the tree and, thus, its complexity. Other hyper-parameters that can be tuned include the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node, and the maximum number of features to consider when looking for the best split. Below are the hyper-parameters that we tweaked for our DT pruning:

- **max\_depth**: This parameter is used to control the depth of the decision tree. It specifies the maximum number of levels in the decision tree. As with any other parameter, it is important to find the optimal balance between model complexity and performance. According to [12], if the max\_depth is set too high, the decision tree may overfit the training data, leading to poor performance on the test data. This is because the tree becomes too complex and captures noise in the data. On the other hand, if the max\_depth is set too low, the decision tree may not capture all the relevant patterns in the data and underfit the data, leading to poor performance on both the training and test data.
- **max\_leaf\_nodes**: This parameter grows the tree with a specified number of leaf nodes based on reduction in impurity. In other words, it specifies the maximum number of leaf nodes that a decision tree can have.
- **min\_samples\_split** The minimum number of samples that are required to split an internal node. If the value of min\_samples\_split is too low, it can result in overfitting the model to the training data. On the other hand, if the value is too high, it can lead to underfitting, where the model is not able to capture the complexity of the underlying data.

[34] provides a function that carries out an exhaustive search over specified hyperparameter values for a classifier, called GridSearchCV. As arguments, this function takes a model, a parameter grid in form of a python dictionary, and cross-validation scheme - a number which specifies the amount of cross-validation folds to use for evaluation, and returns the best set of hyperparameters for the model based on the specified scoring metric. The results from our various experiments including the recorded effects of tuning the DT classifier's hyperparameters are presented in the next section.

## 8 Experiment Appraisal and Conclusion

Transliteration accuracy is generally measured by the percentage of the number of correctly transliterated words divided by the total number of words tested. The correct transliteration here means the standard transliteration as listed in the dictionary. This measure is called word accuracy (WA)[15]. Another way to evaluate the accuracy of an algorithm is by letter accuracy (LA), which is the percentage of correctly transliterated letters of total number of letters in a given test data. The evaluation process of transliteration though, is not straightforward. Transliteration is a creative process that allows multiple variants of a source term to be valid, based on the opinions of different human transliterators. Different dialects in the same language can also lead to transliteration variants for a given source term.

$$WA = \frac{\text{numberOfCorrectTransliterations}}{\text{TotalNumberOfTestWords}}$$

$$LA = \frac{\text{numberOfCorrectlyTransliteratedLetters}}{\text{TotalNumberOfLettersInTest}}$$

Our algorithm experiments are evaluated based on 2 criteria:

- **Performance:** In terms of WA and LA in both direct and intermediate routes with and without machine learning. The numbers that calculate the result of the performance including the result itself are demonstrated in a tabular form later in this chapter.
- **Lithuanization:** The second criterion, also demonstrated in a tabular form, evaluates Lithuanization when the intermediate route is utilized.

The following evaluation criteria can be found in the result tables:

- **Number of Rules:** Specifies the total amount of grammatical rules defined for a language.
- **Number of words in test set:** Specifies the total number of words present in a testing dataset.
- **Number of misspelled words:** Specifies the total number of words that were incorrectly spelled by the transliteration system.
- **Number of letters in test set:** Specifies the total number of characters or alphabets present in a test dataset.
- **Number of incorrect letters:** Specifies the total number of characters or alphabets that were incorrectly spelled by the transliteration system.
- **Model Score:** In the machine learning parlance, a model score quantifies the quality of a model's prediction by comparing the train and test data i.e., the predictability of  $X_i$  being  $Y_i$  given  $Y$ . Loosely speaking, the DT's accuracy.
- **Tree Depth:** Specifies the total number of levels required for the DT to reach a decision

## 8.1 Evaluation of RBMT approach

### 8.1.1 Direct transliteration

One advantage that this route holds over the intermediate route is the computational time taken to transliterate a word from source to the final target. Using JavaScript’s performance API, we computed the overall time taken for a transliteration process between both routes and the result from that experiment validates that the direct route computes an adaptation process than the intermediate route as shown in the table below.

Table 12: Comparing both transliteration routes based on computational speed.

Direct	Intermediate (Georgian)	Intermediate (English)	Intermediate (IPA)
0.41ms	0.72ms	0.60ms	0.83ms

Table 13: Performance evaluation of the direct route in respect to WA and LA.

Measure	Direct
Number of Rules	214
Number of words in test set	428
Number of misspelled words	41
Number of letters in test set	2190
Number of incorrect letters	96
Word Accuracy	90.4%
Letter Accuracy	95%

### 8.1.2 Transliteration using English and Georgian as intermediate languages

From gathered results, the composition of transliteration rules for each of the Yoruba phonemes require noticeably lesser rules in Georgian than English as 198 rules were sufficient to compound a 97% word accuracy into the former on the same amount of words in our test set whereas the latter, with almost fourfold the number of its counterpart’s rules, compounded 82% of word accuracy. Now, though the Georgian language appears to have good numbers in performance as well as in the measurement metrics of our Lithuanization effect test, the English language fairs better in the purpose of our ”multiple transliteration roots” devising, which is finding the route that better adapts a foreign word into the Lithuanian language.

The English language thrives in this adaptation domain because there are quite a number of resources in existence that establishes transliteration conventions from English to Lithuanian, for example, in [28]. With a clear standard in place, we were able to construct Yoruba rules that gives possibility for Anglicization. Let us exemplify the application of our stress removal rule on these two Yorubalized words, ”JIIMU” and ’KARIIMU’ , literally transliterated as ”JIM” and ”KAREEM” , respectively.

Table 14: Stress rule. This word pattern is mostly common amongst Yorubalized words. Note: LC in the table is an abbreviation for "Left Context".

LC 3	LC 2	LC 1	Current Letter
A	A	B	U
A	A	D	U
...	...	...	...
E	E	B	U
E	E	D	U
...	...	...	..
$\dot{E}$	$\dot{E}$	B	U
$\dot{E}$	$\dot{E}$	D	U
...	...	...	...
...	...	...	...

Table 15: Long sound rule. Note: LC, CL, and RC in the table are abbreviations for "Left Context", "Current Letter", and "Right Context", respectively.

LC2	LC1	CL	RC1	RC2	OUTPUT
		A	A		" "
	A	A			A

A borrowed word in the Yoruba language is always stressed in such a way that it has consecutive vowel letters and then ends with phonemes <u> and <i>. Rules in the Tables 14 & 15 above, simply put, asserts that for every *current letter* preceded by a consonant whose preceding letters are diphthongs, must result to an empty string; and, the first letter of any diphthong must result to an empty string while the last is mapped to its transliteration target;, respectively. And though the transliteration spelling might not be accurate (penalized in both WA & LA), the result affords for one that can be Lithuanized. Let us look at the rule 58 in [28], we will see that it agrees with the transliteration of our sample words:

JIIMU → JIM → DŽIMAS  
KARIIMU → KARIM → KARIMAS

Because the Georgian language is not inflected and neither does adaptation of a foreign word to it seem practical (because the Georgian name either ends with a *dze* or *shvili*, connotations of which are symbolic to the culture, and will also transform the essence of the foreign word) the above examples will result to the following as in direct transliteration:

JIIMU → ჯიიმუ → DŽĪMU  
KARIIMU → კარიიმუ → KARĪMU

On the one hand of what transliteration is, the result provided by the Georgian route is acceptable since it preserves the sound of the word in the source language while representing the word in the target language. The English route, on the other hand adapts the source word to meet the phonetic regularities of the target language.

Table 16: Performance of the intermediate route when the English and Georgian languages are used at the hub of the transliteration process in pertinence to Word and Letter Accuracy.

Measure	English	Georgian
Number of Rules	764	198
Number of words in test set	428	428
Number of misspelled words	78	29
Number of letters in test set	3390	2980
Number of incorrect letters	682	87
Word Accuracy	82%	97%
Letter Accuracy	80.1%	97.3%

The tables 14 and 15 demonstrate the performance of the transliteration route in terms of WA and LA as well as how the route helps in achieving the Lithuanization of a given Yoruba word. The former table shows that the Georgian language outperforms the English language in WA and LA, whereas the latter table demonstrates that the English language affords for Lithuanization better than its counterpart.

Table 17: Performance of the intermediate route when the English and Georgian languages are used at the hub of the transliteration process in pertinence to how they facilitate Lithuanization of a Yoruba word.

Measure	English	Georgian
Number of words in test set	350	399
Number of misspelled words	108	93
Number of transcribed words	242	306
Number of Lithuanized words	139	79
Lithuanization aid	40%	20%

### 8.1.3 Transliteration using IPA as an intermediate language

Going by our assessments of the IPA as an intermediate language, we are able to establish the fact that evaluating this method by its ability to facilitate the adaptation of a foreign word to meet the phonetic regularities of the target language will be inconsistent. For instance, given 1000 foreign words with obvious Lithuanian-adaptable ending, we would have a 100% Lithuanization effect. Drawing comparison with its counterpart, it is important to highlight though that the IPA records better word and letter accuracy. The problem of missing sounds in the mapping of source-IPA to target-IPA, however, is noticeable and requires manual effort to substitute missing sounds for the closest phoneme that sounds alike. The IPA result table contains measurement criteria that were not previously described, let us look at them:

- Words syllabified : Specifies the total number of words that were broken down into syllables in the course of the experiment.
- Letters in words syllabified: States the total number of letters present in those words that were broken down.
- Failed SP → TIPA mapping: Specifies the number of source language IPA symbols that were either mapped incorrectly or not mapped at all.

- TIPA → TG mapping: Specifies the total number of target IPAs that were successfully mapped to their graphemes.
- Number of words Lithuanized: Specifies the total number of transliteration result with Lithuanian morphological identity.

Table 18: Performance of the IPA as an intermediate language.

Measure	Count
Words in test set	428
Number of syllables	1217
Number of Letters	2567
Words syllabified	400
Letters in words syllabified	2439
Failed SP → TIPA mapping	30 (missing sounds)
TIPA → TG mapping	370
Number of words Lithuanized	69
Word accuracy	96.4%
Letter accuracy	98.3%
Syllable accuracy	93%

The table above shows that the IPA is a powerful resource to utilize for transliteration as it records better word and letter accuracy than its language counterparts. A low count of only 7.5% of missing sounds, also suggests that its drawback is minimal and above all, resolvable.

## 8.2 Evaluation of machine learning approach

Like the evaluation of experiments done in RBMT for both transliteration routes including the language resources employed at the hub of the intermediate route, we make similar evaluations on machine learning as means of adaptation on basis of tree pruning i.e., performance before and after pre-pruning of the decision tree.

### 8.2.1 Direct transliteration using ML

Table 19: Model performance before and after pruning when making direct adaptation. Note, TD in the table means Tree Depth.

	Mean word size	Model Score	WA	LA	Mean Leaves	TD
Before pre-pruning	6	72.1	69.1	89.4	714	102
After pre-pruning	7	73.3	69.3	89.8	603	85

Using [34]’s `train_test_split` function (to split a dataset into a training set and a testing set) we had dual sets of training and testing data where 30% of the overall data is allocated to testing and the remainder for training the DT classifier. The model scored a prediction accuracy of 72.1% before pruning and a slight increase in the score with 1.2% increment as a result of pre-pruning

the model. Before pre-pruning the tree, we exposed the classifier to unseen names that averaged a word length of 6, for this, the model recorded WA of 69.1% but returned a better score of 89.4% for LA. Using the `get_n_leaves` function available to a classifier for retrieving the number of decision leaves amassed by a DT, we found that the model generated up to 714 leaves with a depth of 102. After pre-pruning the tree, we made another round of testing using the same methods, and on rare occasions tried names with longer word length: we noticed minute differences in WA and LA (0.2% and 0.4% increments, respectively), however, there were noticeable improvements in the number of leaves and tree depth with 15.5% decrement in average number of leaves and 16.6% decrement in tree depth, respectively.

### 8.2.2 Intermediate transliteration using ML

Table 20: Model performance before and after pruning when making adaptation using intermediate languages. TD, like in Table 19, means Tree Depth.

Georgian	Mean word size	Model Score	WA	LA	Mean Leaves	TD
Before pre-pruning	6	80.6	69.1	89.4	598	96
After pre-pruning	6	80.4	69.3	89.8	710	80
English						
Before pre-pruning	6	58.1	70.3	74.6	697	176
After pre-pruning	7	58.0	73.8	74.9	803	109
IPA						
Before pre-pruning	7	89.2	85.2	97.6	441	76
After pre-pruning	7	93.2	85.1	97.5	519	63

Similar performance methods were used on the intermediate route as with direct. In terms of model score, the classifier recorded the lowest confidence on our English dataset with 58.1% before pre-pruning and a 0.1% decrement after pre-pruning whereas the highest model confidence was recorded on the IPA with 89.2% and 93.2% before and after pre-pruning, respectively. Though the estimated accuracy with Georgian is second to the IPA with a noticeable margin of 9.4% before we tweaked hyper-parameters and 13.4% after doing so, the difference between the Georgian and English languages is even greater, with the model having more confidence in the Georgian dataset than English before pre-pruning by 22.5%, and 22.4% after pre-pruning. In WA and LA, the English language fared better than its counterparts, when observing the effect of pre-pruning the classifier: 3.8% increase in WA and 0.5% increase in LA for English; 0.2% increase in WA and 0.8% increase in LA for Georgian; 0.1% decrease in WA and LA for IPA. Finally, there seem to be an inverse relationship between a tree’s depth and the number of leaves it has: the latter seem to decrement when the former grows, and vice versa.



### 8.3 Conclusion

Based on our study, we found that the intermediate route for transliteration between Yoruba and Lithuanian fares better than the direct route when using tools that facilitate the adaptation process, such as the English and Georgian languages and IPA. We also found that the IPA was the most effective language resource in aiding the adaptation process. The syllabification of words proved to be integral to the efficiency of IPA as an intermediate language resource, and this technique can be applied in other areas of NLP, such as text-to-speech synthesis of the Yoruba language to Lithuanian.

We also observed that, as long as machine transliteration continues to require grammatical rule of thumbs, the rule-based approach will continue to maintain relevance and produce better results than machine learning. Our study revealed that the English language also played a significant role in Lithuanizing a Yoruba word and, when combined with IPA, resulted in better transliteration accuracy in both word and letter.

In conclusion, our study has highlighted the effectiveness of the intermediate route, IPA, and syllabification of words in transliteration between Yoruba and Lithuanian. These findings could have significant implications for NLP, especially in areas such as text-to-speech synthesis. Our study also emphasized the continuing relevance of the rule-based approach in machine transliteration.

## Bibliography

- [1] University of Georgia African Studies Institute. *Yoruba Phonology: A description of the Yoruba orthography, based on its standard dialect*. URL: <http://www.africa.uga.edu/Yoruba/phonology.html>. (accessed: 23.04.2023).
- [2] J. Bergstra and Y. Bengio. “Random Search for Hyper-Parameter Optimization”. In: 13 (Feb. 2012), pp. 281–305. ISSN: 1532-4435. URL: <https://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf>.
- [3] M. Bollmann. “A Large-Scale Comparison of Historical Text Normalization Systems”. In: *Proceedings of the 2019 Conference of the North. Association for Computational Linguistics*, 2019. URL: <https://doi.org/10.18653%2Fv1%2Fn19-1389>.
- [4] L. Breiman. *Classification And Regression Trees*. Routledge, Oct. 2017. URL: <https://doi.org/10.1201%2F9781315139470>.
- [5] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer. “The Mathematics of Statistical Machine Translation: Parameter Estimation”. In: *Machine Translation* 19.2 (June 1993), pp. 263–311. URL: <https://aclanthology.org/J93-2003.pdf>.
- [6] P. Chormai, P. Prasertsom, J. Cheevaprawatdomrong, and A. Rutherford. “Syllable-based Neural Thai Word Segmentation”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, 2020. URL: <https://doi.org/10.18653%2Fv1%2F2020.coling-main.407>.
- [7] C. Davidson. “Transcription: Imperatives for Qualitative Research”. In: *International Journal of Qualitative Methods* 8.2 (June 2009), pp. 35–52. URL: <https://doi.org/10.1177%2F160940690900800206>.
- [8] L. Dhore, R. Dhore, and P. Rathod. “Survey on Machine Transliteration and Machine Learning Models”. In: *International Journal on Natural Language Computing* 4.2 (Apr. 2015), pp. 09–30. URL: <https://doi.org/10.5121%2Fijnlc.2015.4202>.
- [9] M. Divay and M. Guyomard. “Grapheme-to-phoneme transcription for French”. In: *ICASSP ’77. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Institute of Electrical and Electronics Engineers. URL: <https://doi.org/10.1109%2Ficassp.1977.1170351>.
- [10] Ethnologue. *A reference for making informed decisions in every language context worldwide*. URL: <https://www.ethnologue.com/>. (accessed: 23.04.2023).
- [11] S. Fine, Y. Singer, and N. Tishby. “The Hierarchical Hidden Markov Model: Analysis and Applications”. In: *Machine Learning* 4.2 (Apr. 1998), pp. 41–62. URL: <https://doi.org/10.1023/A:1007469218079>.
- [12] J. Franklin. “The elements of statistical learning: data mining, inference and prediction”. In: *The Mathematical Intelligencer* 27.2 (Mar. 2005), pp. 83–85. URL: <https://doi.org/10.1007%2Fbf02985802>.
- [13] W. Gao, K. Wong, and W. Lam. “Phoneme-Based Transliteration of Foreign Names for OOV Problem”. In: *Natural Language Processing – IJCNLP 2004*. Springer Berlin Heidelberg, 2005, pp. 110–119. URL: [https://doi.org/10.1007%2F978-3-540-30211-7\\_12](https://doi.org/10.1007%2F978-3-540-30211-7_12).
- [14] A. Hautli-Janisz. “Pushpak Bhattacharyya: Machine translation”. In: *Machine Translation* 29.3-4 (July 2015), pp. 285–289. URL: <https://doi.org/10.1007%2Fs10590-015-9170-7>.

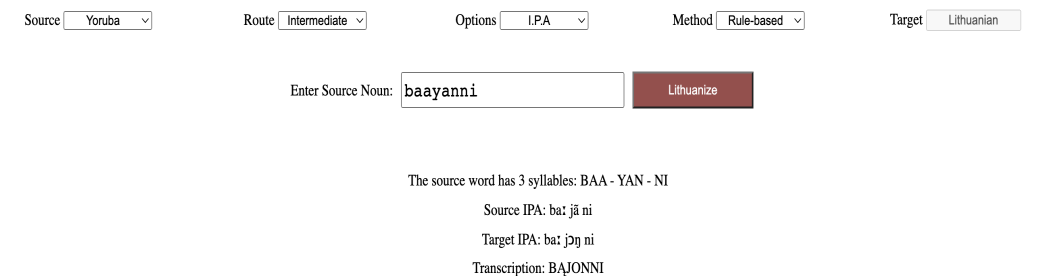
- [15] R. Hierons. "Machine learning. Tom M. Mitchell. Published by McGraw-Hill, Maidenhead, U.K., International Student Edition, 1997. ISBN: 0-07-115467-1, 414 pages. Price: U.K. £22.99, soft cover." In: *Software Testing, Verification and Reliability* 9.3 (Sept. 1999), pp. 191–193. URL: <https://doi.org/10.1002%2F%28sici%291099-1689%28199909%299%3A3%3C191%3A%3Aaid-stvr184%3E3.0.co%3B2-e>.
- [16] J. Hutchins and H. Somers. *An Introduction to Machine Translation*. London: Academic Press Limited. URL: <https://www.computer.org/csdl/magazine/co/1992/12/rz118/13rRUB6Sp0K>. (accessed: 23.04.2023).
- [17] GLM Institute. *The Global Language Monitor Institute*. URL: <https://www.languagemonitor.com/>. (accessed: 23.04.2023).
- [18] M. Al-Kabi, S. Kazakzeh, B. Abu Ata, and S. Al-Rababah. "A novel root based Arabic stemmer". In: *Journal of King Saud University - Computer and Information Sciences* 27.2 (Apr. 2015), pp. 94–103. URL: <https://doi.org/10.1016%2Fj.jksuci.2014.04.001>.
- [19] S. Karimi, F. Scholer, and A. Turpin. "Machine transliteration survey". In: *ACM Computing Surveys* 43.3 (Apr. 2011), pp. 1–46. URL: <https://doi.org/10.1145%2F1922649.1922654>.
- [20] P. Kasparaitis. "Automatic Transliteration of Polish and English Proper Nouns into Lithuanian". In: *Information Technology and Control* 52.1 (Mar. 2023), pp. 128–139. URL: <https://doi.org/10.5755%2Fj01.itc.52.1.32353>.
- [21] K. Kaur and P. Singh. "Review of Machine Transliteration Techniques". In: *International Journal of Computer Applications* 107.20 (2014), pp. 13–16. URL: <http://dx.doi.org/10.5120/18866-0061>.
- [22] K. Knight and J. Graehl. "Machine transliteration". In: *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics - Association for Computational Linguistics*, 1997. URL: <https://doi.org/10.3115%2F979617.979634>.
- [23] K. Knight and Y. Al-Onaizan. "Translation with Finite-State Devices". In: *Machine Translation and the Information Soup*. Springer Berlin Heidelberg, 1998, pp. 421–437. URL: [https://doi.org/10.1007%2F3-540-49478-2\\_38](https://doi.org/10.1007%2F3-540-49478-2_38).
- [24] P. Koehn, F. Och, and D. Marcu. *Statistical Phrase-Based Translation*. Tech. rep. Jan. 2003. URL: <https://doi.org/10.21236%2Fada461156>.
- [25] S. Krauwer. *The Basic Language Resource Kit (BLARK) as the First Milestone for the Language Resources Roadmap*. URL: <http://elsnet.let.uu.nl/dox/krauwer-specom2003.pdf>. (accessed: 23.04.2023).
- [26] J. Kuang. "The Tonal Space of Contrastive Five Level Tones". In: *Phonetica* 70.1-2 (Oct. 2013), pp. 1–23. URL: <https://doi.org/10.1159%2F000353853>.
- [27] Y. Kushnir. *Structure of Lithuanian*. URL: [https://home.uni-leipzig.de/yuriykushnir/strucclith/class\\_2.pdf](https://home.uni-leipzig.de/yuriykushnir/strucclith/class_2.pdf). (accessed: 13.11.2022).
- [28] The State Commission of the Lithuanian Language. *he State Commission of the Lithuanian Language. Rules for transcription of proper names from English into Lithuanian*. URL: <https://vlkk.lt/media/public/file/Svetimzodziai/anglu-patvirtinta-VLKK-2021-09-30.pdf>. (accessed: 30.04.2023).

- [29] The State Commission of the Lithuanian Language. *The State Commission of the Lithuanian Language. Resolution No. 60 On the Spelling and Punctuation of the Lithuanian Language, 19 June 1997*. URL: <https://www.vlkk.lt/vlkk-nutarimai/suvestines-nutarimu-redakcijos/del-lietuviu-kalbos-rasybos-ir-skyrybos>. (accessed: 09.12.2021).
- [30] F. Och and H. Ney. “A Systematic Comparison of Various Statistical Alignment Models”. In: *Computational Linguistics* 29.1 (Mar. 2003), pp. 19–51. URL: <https://doi.org/10.1162/2F089120103321337421>.
- [31] A. Odugbemi, T. Falola, and A. Akinyemi (Eds). “Encyclopedia of the Yoruba. Bloomington and Indianapolis: Indiana University Press, 2016”. In: *Yoruba studies review* 3.1 (2021), p. 371. URL: <http://dx.doi.org/10.32473/ysr.v3i1.129933>.
- [32] J. Oh and K. Choi. “An Ensemble of Grapheme and Phoneme for Machine Transliteration”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2005, pp. 450–461. URL: [https://doi.org/10.1007/2F11562214\\_40](https://doi.org/10.1007/2F11562214_40).
- [33] J. Okanlawon. *AN ANALYSIS OF THE YORUBA LANGUAGE WITH ENGLISH*. URL: <https://cos.northeastern.edu/wp-content/uploads/2018/09/Jolaade-Okanlawon-An-Analysis-of-Yoruba-with-English.pdf>. (accessed: 13.11.2022).
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
- [35] “Publication of the iHandbook of the International Phonetic Association/i”. In: *Journal of the International Phonetic Association* 27.1-2 (June 1997), p. 175. URL: <https://doi.org/10.1017/2Fs002510030000548x>.
- [36] P. Zemlevičiūtė R. Vladarskienė. *P. Spelling of the Lithuanian Language. Rules, Comments, Tips. Lithuanian Language Institute publishing house, Vilnius, 2022*. URL: [https://vlkk.lt/media/public/file/Nutarimai/Ra%C5%A1yba\\_2022.pdf](https://vlkk.lt/media/public/file/Nutarimai/Ra%C5%A1yba_2022.pdf). (accessed: 18.01.2023).
- [37] R. Rabiner, G. Wilpon, and K. Soong. “High performance connected digit recognition using hidden Markov models”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.8 (1989), pp. 1214–1225. URL: <https://doi.org/10.1109/2F29.31269>.
- [38] A. Rollings. “System and chaos in English spelling: the case of the voiceless palato-alveolar fricative”. In: *English Language and Linguistics* 7.2 (Oct. 2003), pp. 211–233. URL: <https://doi.org/10.1017/2Fs1360674303001084>.
- [39] L. Salzberg. “C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993”. In: *Machine Learning* 16.3 (Sept. 1994), pp. 235–240. URL: <https://doi.org/10.1007/2Fbf00993309>.
- [40] B. Siertsema and A. Bamgbose. “Yoruba Orthography: A Linguistic Appraisal with Suggestions for Reform”. In: *American Anthropologist*, 70.2 (1968), pp. 429–430. URL: <http://dx.doi.org/10.1525/aa.1968.70.2.02a00970>.
- [41] B. Stalls and K. Knight. “Translating names and technical terms in Arabic text”. In: *Proceedings of the Workshop on Computational Approaches to Semitic Languages - Semitic '98*. Association for Computational Linguistics, 1998. URL: <https://doi.org/10.3115/2F1621753.1621760>.

- [42] K. Toutanova, H. Tolga Ilhan, and C. Manning. “Extensions to HMM-based statistical word alignment models”. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*. Association for Computational Linguistics, 2002. URL: <https://doi.org/10.3115%2F1118693.1118705>.
- [43] L. Trask. *A Dictionary of Grammatical Terms in Linguistics*. Routledge, May 1997. URL: <https://doi.org/10.4324/9781315832531>.
- [44] A. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2 (Apr. 1967), pp. 260–269. URL: <https://doi.org/10.1109%2Ftit.1967.1054010>.
- [45] S. Vogel, H. Ney, and C. Tillmann. “HMM-based word alignment in statistical translation”. In: *Proceedings of the 16th conference on Computational linguistics -*. Association for Computational Linguistics, 1996. URL: <https://doi.org/10.3115%2F993268.993313>.
- [46] B. Weinberg. “TRANSLITERATION IN DOCUMENTATION”. In: *Journal of Documentation* 30.1 (Jan. 1974), pp. 18–31. URL: <https://doi.org/10.1108%2Feb026567>.
- [47] Wikipedia. *The pronunciation key for IPA transcriptions of Yoruba on Wikipedia*. URL: <https://en.wikipedia.org/wiki/Help:IPA/Yoruba>. (accessed: 03.10.2021).
- [48] L. Sun X. Jiang and D. Zhang. “A syllable-based name transliteration system”. In: *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*. 2009, pp. 96–99.

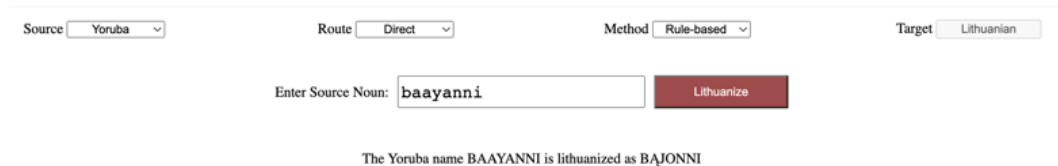
## 9 Appendix. Transliteration Software

Our transliteration web software (<https://yoruba-lithuanian-adaptation.000webhostapp.com/vu/>) provides a user-friendly solution for transliterating Yoruba words into Lithuanian. The software is built on a combination of two programming languages, Javascript and Python (Flask). The Javascript is responsible for the user interface design, as well as the rule-based transliteration algorithm. This means that users can easily input Yoruba words and receive near-instant transliterations into Lithuanian. The Python's Flask framework, on the hand, is used to expose the decision tree classifier to the client through REST API. This enables the model to improve its accuracy over time by learning from the user inputs.



The screenshot shows the user interface of the transliteration software. At the top, there are four dropdown menus: 'Source' set to 'Yoruba', 'Route' set to 'Intermediate', 'Options' set to 'I.P.A', and 'Method' set to 'Rule-based'. To the right, a 'Target' dropdown is set to 'Lithuanian'. Below these is an input field labeled 'Enter Source Noun:' containing the text 'baayanni'. To the right of the input field is a red button labeled 'Lithuanize'. Below the input field, the following text is displayed: 'The source word has 3 syllables: BAA - YAN - NI', 'Source IPA: baː jā ni', 'Target IPA: baː jɔŋ ni', and 'Transcription: BAJONNI'.

Figure 12: A snapshot of our transliteration software's user interface when using the IPA as an intermediate language



The screenshot shows the user interface of the transliteration software. At the top, there are four dropdown menus: 'Source' set to 'Yoruba', 'Route' set to 'Direct', 'Method' set to 'Rule-based', and 'Target' set to 'Lithuanian'. Below these is an input field labeled 'Enter Source Noun:' containing the text 'baayanni'. To the right of the input field is a red button labeled 'Lithuanize'. Below the input field, the following text is displayed: 'The Yoruba name BAAYANNI is lithuanized as BAJONNI'.

Figure 13: A snapshot of our transliteration software's user interface when using the direct route

The Figures 12 and 13 above are snapshots of our described software. The system comprises of 3 components:

- **Navigation:** The navigation provides a set of options used in configuring the transliteration procedure such as the transliteration route, intermediate options (if the desired route is intermediate), and the adaptation method i.e., rule-based or machine learning.
- **Input:** The input component provides an input field to the user as well as a button which on click event triggers the pre-processing phase of the transliteration procedure.

- **Result:** Right beneath the Input is the result component. This component is responsible for presenting the outcome of an adaptation process in a format depending on the configured options