Computer Science • 24(1) 2023 https://doi.org/10.7494/csci.2023.24.1.4494

Franklin Ọládiípọ̀ Asahiah
Mary Taiwo Onífádé
Adekemisola Olufunmilayo Asahiah
Abayomi Emmanuel Adegunlehin
Adekemi Olawunmi Amoo

DIACRITIC-AWARE YORÙBÁ SPELL CHECKER

Abstract *Spell checking and correction is still in its infancy for the Yorùbá language; existing tools cannot be directly applied to address the problem, as Yorùbá uses diacritics extensively for distinguishing phonemes and for marking tone. A model was formulated as a parallel combination of a unigram language model and a diacritic model to form a dictionary sub-model that can be used by error-detection and candidate-generation modules. The candidate-generation module was implemented as a reverse Levensthein edit-distance algorithm. The system was evaluated by using detection accuracy (calculated from the precision and recall) and suggestion accuracy (SA) as metrics. Our experimental setups compared the performance of the component subsystems when used alone and with their combination into a unified model. The detection accuracies for the different models range from 93.23 to 95.01%, and the suggestion accuracies range from 26.94 to 72.10%. The results indicated that each of the sub-models in the dictionary played different roles.*

Keywords tone, phonemes, diacritic, unigram, tools

Citation Computer Science 24(1) 2023: 31–51

1. Introduction

Writing is a system of recording a language by means of visible or tactile marks [9]. Writing provides permanence to communications and is, therefore, expected to be carefully encoded. Among several others, spelling is a factor that affects the quality of written documents and seems to be an oft-recurring challenge [22]. In fact, the quality of compositional writing is largely influenced by spelling (among other things) according to [10]. The centrality of spelling to writing and written documents is the basis for the development of spell checkers. However, spell checkers can also be applied in other areas of natural language processing (NLP), such as in machine translation, information retrieval, document and text search, optical character recognition, and text-to-speech [14].

Spell checking is the process of detecting (and sometimes providing) suggestions for incorrectly spelled words in a text; it is an application program that flags words in a text that may not be spelled correctly [17]. Naseem [19] stated that the task of a spell corrector can generally be divided into three facets:

- detecting errors,
- finding possible corrections,
- ranking suggested corrections in order to choose best option.

Spelling-error detection and correction tools work mostly at the word level and use a dictionary as a resource. An algorithm chooses each word from a text, which is then looked up in a speller lexicon; if the word is not in the dictionary, an error is detected. Another algorithm generates words that are "close" to the form of the original as replacement candidates. A ranking algorithm then arranges the candidate words in an order of most- to least-likely and present replacement options to the user. Spelling errors can occur for various reasons, and errors that are related to misspelled words can be categorized into two basic classes:

i) **Non-word errors** – where a misspelled word is not a valid word in a language or does not have a dictionary meaning. In the English language, for example, "**bettle**" is a non-word string that could have been a misspelling of any of the following: "*beetle*," "*bottle*," and "*battle*." In Yorùbá, "**Táíwà**" could be a misspelling of any of the following: "*Táwà*," "*Táíwò*," and "*Tiwa*."

ii) **Real-word errors** – where a string of letters form a correct word but is inappropriate in the context of its usage; for example, "a **peace** of cloth" in the English language and "*mo lá* ***àlọ́***″" in Yorùbá. The two bold words are correct words in the respective vocabularies of the two languages but are wrong in their contexts.

As mentioned earlier, there are various reasons for or causes of spelling errors. Errors can occur or be introduced in the following ways (as well as in any combination):

a) Typographic errors – these related to mis-punching the keyboard; i.e., *Àqọn* and *Àwø̣n*; these errors fall into one of the following categories:

   - Substitution errors – happens when one letter is substituted by another letter; i.e., *Funfum* and *Funfun*.
   - Deletion errors – occurs when at least one of the characters is deleted in a word; for instance, Àlej and *Àlejò*.
   - Transposition errors – when two characters in a word are transposed or interchanged; i.e., *Oluok* and *Olùkọ́*.

b) Cognitive errors – caused by author misconception; *Alade* and *Àlàdé*.

c) Phonetic errors – happen as a result of substituting a phonetically equivalent sequence of letters; an example word like Ìbàdàn cannot be written as Ìbàdọ̀n.

2. Review of related works

Yorùbá is spoken by more than 30 million users – mostly in Nigeria, Benin, and Togo [11]; it is also used for religious activities in Cuba, Brazil, Argentina, Trinidad, and parts of the United States and Canada [5]. It is recognized for use in Nigeria National Assembly [12].

The *Yorùbá* language is used by the media, press, radio, and television, It is also used as a language of formal instruction as well as a curriculum subject at some primary, secondary and tertiary levels in Nigeria, the U.S., and the U.K. [6]. In previous research, [3] analyzed spelling-error patterns in typed *Yorùbá* text documents; an examination of the spelling errors was conducted (which is essential in the development of a good spell checker). In the article, the findings revealed that *Yorùbá*'s spelling-error patterns generally followed those of other languages. It was also discovered that the majority of the errors were related to vowels (with consonants accounting for fewer than 15% of all errors), while the number of errors in a word did not appear to be related to the length of the word.

The impact of diacritics on spelling errors is greater in *Yorùbá*, where diacritics are primarily used for tone marking; these account for more than 80% of all spelling errors. With languages such as Brazilian Portuguese and Spanish (where diacritics are primarily used for character differentiation), these account for fewer than 60% of all errors. The authors in [3] concluded that, while the character set that is used in a language influences the distribution of spelling errors to a large extent, the purpose for which diacritics are used in a language also influences the distribution of the spelling errors.

In spell checking, some machine-learning techniques like n-gram have been used, while algorithms like Edit Distance and Wordex have also been used in some of the existing works on other languages (Hausa, Sindhi, Romanian, Punjabi, *et cetera*). The authors of [8, 23] and [16] developed spell correctors for the Hausa, Romanian, and Punjabi languages, respectively. The former authors designed a corrector that operated essentially on the dictionary and characteristics of the Hausa alphabet. They opted for a data structure trie and harsh table to represent the dictionary and used minimum edit distance to rank the correct word; the latter author used effective vocabulary representation, a similar word-detection algorithm, and automatic word inflection (a dictionary of words was also built).

Ahmed et al. [1] developed an automatic spelling-correction tool to improve retrieval effectiveness based on the n-gram method. This was a language-independent spell checker that was based on an enhancement of an n-gram model that detects non-word errors in a text document. The spell checker worked on the basis of a multi-spell algorithm that compares keywords that are provided by a user with the correct words that are contained in the dictionary; if a word is detected as an error, then the algorithm builds an n-gram for the misspelled word, and the correction candidate is selected from the dictionary. Both the n-gram and similarity score are computed for the selected words. The multi-threaded approach was used in order to improve the performance of the tool. [7] proposed a new context-sensitive error-correction model that detected and corrected both non-word and real-word error in a generic computer text document using Goggle Web 1T 5-gram information. Three algorithms were used: the task of the first algorithm was to detect non-word errors using the Google web 1T unigram data set, the second algorithm generated list of candidate spellings for each detected error in the text using the Google web 1T unigram data set and a character-based two-gram model, while the third algorithm was used to perform context-sensitive error correction and select the best appropriate spelling candidate using five-gram word counts from the Google 1T data set (99% of the non-word errors and 70% of the real-word errors were corrected by the method). Similarly, [21] developed a non-word Kannada spell checker that used a morphological analyzer and a dictionary lookup method, and [15] developed a Khmer spell checker (the word segmentation was based on an n-gram) and a hidden Markov model-based string-matching algorithm.

In a hybrid-approach spell checker for Oriya [20], the authors integrated edit distance, n-gram techniques, and a morphological analyzer for the system. The results generated appropriate suggestion sets for misspelled words by matching with dictionary words. Comparing the suggestion accuracy of three experiments that were carried out using by only edit distance was 86%, 83%, and 85%, while the hybrid approach resulted in 94%, 95%, and 95%, respectively.

Spell checker for non-word error detection (survey) [14]: The authors carried out a survey on various spell checkers for word-error detection for various languages. The paper analyzed various techniques in error-detection and error-correction algorithms like the minimum edit distance, rule-based, Soundex, neural net, and probability methods. Spell checkers in languages like Indonesian, Hindi, and Kannada were also reviewed. The study was able to discover that there are many spell checkers for detecting non-word errors in text documents and that different methods are used for various languages. The work will be useful to this study in the area of spell checking *Yorùbá* text; the scope of this work was limited to non-word due to a lack of resources (like a parser and grammar checker).

3. Methodology

To accomplish the development of the spell checker, we gathered data to be used in the model's formulation and implementation. In addition, we formulated a system design that consisted of dictionaries and rule sets, and we implemented the design into the software using the Python programming language and its libraries.

3.1. Data gathering and analysis

Our data was gathered through manual and automated web crawling from various online sources that contained *Yorùbá* text in a reasonable format. A reasonable format is defined as one that attempts to follow the Joint Consultative Committee's 1974 Standard *Yorùbá* orthography [13]. The data for this study was divided into two groups: developmental data, and test data. The developmental data came from various sources such as Internet-related sources (2466 words – accounting for about 10% of the total) and digitized printed materials (4191 words – about 17%). Around 15,000 words (approximately 60%) were acquired from the data from existing research [4], and 3206 words (13%) were from freshly created digital documents.

3.1.1. Data sources

Documents from web-based sources included Yoruba translations of articles from international organizations, non-governmental bodies, *Yorùbá* cultural organizations, various educational boards, councils, and ministries, bulletins, and Newswire as well as online Yoruba versions of religious organization documents. Other sources of online *Yorùbá* content used were discussion boards and social networking messages. Discussion boards and social networking content could be problematic due to the low quality of the text that is often found in such portals. The digitized materials were mainly from educational textbooks. The development data formed the basis for our error analysis, as 4448 words that were selected from different portions of the data were analyzed. The development data was thereafter cleaned through automated foreign word removal and the manual correction of words with errors. The cleaned data was used to create the following language resources: a diacritic dictionary, and a unigram language model of words and frequency counts.

3.1.2. Training and test data

Our training data (1381 words) was randomly selected to fine-tune the performance of the system. The test data was collected separately by using a simple questionnaire to collect *Yorùbá* words from the respondents; these words formed our test corpus. The respondent were requested to write out 20 *Yorùbá* words that they were requested to categorize into either common or uncommon words based on their own levels of understanding and using *Yorùbá*. In the data, vowels accounted for 48.08% of the data, while pure and syllabic consonants made up the remaining 51.92%. Further information concerning the characterization of the text by the letters of the *Yorùbá* alphabet is shown in Table 1.

Table 1
Distribution of letters of *Yorùbá* alphabet in text

Consonants	Percentage of consonants	Vowels	Percentage full-forms	Percentage base-forms
b	2.35	a	3.57	12.95
d	1.60	à	5.72	
f	1.13	á	3.65	
g	2.00	e	1.34	4.93
gb	1.42	è	1.56	
h	0.48	é	2.03	
j	1.62	ẹ	1.01	4.37
k	2.81	ẹ̀	1.67	
l	3.69	ẹ́	1.68	
m	1.88	i	4.06	14.97
n	8.46	ì	4.23	
p	1.49	í	6.68	
r	3.61	o	2.60	6.57
s	1.47	ò	1.70	
ṣ	1.91	ó	2.27	
t	4.13	ọ	3.16	6.47
w	3.15	ọ̀	1.65	
y	2.27	ọ́	1.66	
–	–	u	0.79	4.29
–	–	ù	1.07	
–	–	ú	2.43	
Consonant subtotal	51.92%	Vowel subtotal		48.08%

The diacritic characterization of the text included the fact that 16.53% of the vowels carried no tone mark (because they carried a middle tone), while the low-tone (17.61%) and high-tone (20.42%) words were marked with grave and acute accents, respectively. A full 12.75% of all of the letters had a dot-below (under-dot) diacritic. Information about the training and test data is shown in Table 2.

Table 2
Some features of training and test data

Description Parameter	Train	Test
Number of words	1381	300
Number of characters	108,108	19,381
Length of longest word	30 chars.	15 chars.
Length of shortest word	1 char.	2 chars.
No. of monosyllabic words	544	12
No. of disyllabic words	5027	103
No. of trisyllabic words	4979	115
No. of polysyllabic words	583	170

3.2. Spelling errors

Spelling errors are of two types: real-word, and non-word. A real-word error is said to have occurred when the spelling of a word that is needed in a particular position (or context) is replaced with an alternate spelling that is also a dictionary entry, while a non-word error is said to have occurred when the spelling of a word in a text is replaced with an alternate spelling that is not a valid dictionary entry. The approaches and resources that are needed to address real-word errors are different from those that are intended for non-word errors. In research on spelling error patterns in *Yorùbá* text, [3] reported that 67.72% of all misspelled words were non-word errors, while the remaining 32.28% were real words. We therefore decided to develop a spelling corrector for the more prevalent non-word errors of *Yorùbá* text. Our subsequent discussion of spelling errors also addresses non-word errors. We discuss the different categories of errors from linguistic and mechanistic viewpoints and present a model that represents our understanding of the underlying issues.

3.2.1. Linguistic description of spelling errors

The various causes of spelling errors have been classified in previous studies in various ways [18, 24]; the generally agreed-upon broad categories are phonological, orthographic, and morphological. *Yorùbá* is an isolating language with lexically contrastive tones; our own categorization for the linguistic descriptions of misspellings or factors for spelling errors are phonological, orthographic, and typographical. We follow the definition of [2] for these categories (except for typographical, which has a dictionary definition).

Phonological errors: spelling errors are deemed to be of a phonological origin when a phoneme in the pronunciation of a word is not represented by a grapheme when writing it down (or it is substituted). Some examples that can be found in *Yorùbá* text are shown in Table 3.

Table 3
Sample of phonological errors in *Yorùbá*

SN.	Misspelling		Correct	Comment
a.	"oun"	⇒	"ohun"	common mistake
b.	"bobo"	⇒	"gbogbo"	mostly among YSL speakers
c.	"ǹkan"	⇒	"nǹkan"	common mistake
d.	"isẹ́"	⇒	"iṣẹ́"	dialectical substitution

Orthographic errors: the grapheme that is used to represent a phoneme is incorrect in a specific word or syllable context, or a spelling rule is not applied. Some examples are shown in Table 4.

Table 4
Sample of orthographic errors in *Yorùbá*

SN.	Misspelling		Correct	Comment
a.	"mọ̀n"	⇒	"mọ̀"	spelling rule violation
b.	"Ìbàdọ̀n"	⇒	"Ìbàdàn"	spelling rule violation
c.	"ìban"	⇒	"ìbọn"	spelling rule violation
d.	"fún u"	⇒	"fún un"	nasality non-indication

Typographical errors: This refers to the unintentional errors that happen when one accidentally hits the wrong key on a keyboard or enters a different letter than one that was intended due to a mechanical issue such as a slip of the fingers. Some examples are shown in Table 5.

Table 5
Sample of typographical errors in *Yorùbá*

SN.	Misspelling		Correct	Comment
a.	“pkùnrin”	⇒	“okùnrin”	key “p” pressed instead of key “o”
b.	“opkùnrin”	⇒	“okùnrin”	keys “p” & “o” pressed together
c.	“okùrin”	⇒	“okùnrin”	omission

3.2.2. Edit errors

An alternate mechanistic model of a spelling error is referred to as an edit error. This error model has been investigated for *Yorùbá*, and the outcome was reported in [3]. Edit errors can be classified into four traditional categories: insertion, deletion, substitution/transposition, and concatenation. Their descriptions and mathematical modeling follow these definitions: an alphabet Σ, $|\Sigma| = 43$ for *Yorùbá*, a dictionary Λ consisting of strings in Σ, and a given misspelled word, W as a sequence of letters: $W = x_1 \cdots x_i \cdots x_n; x_i \in \Sigma$ and $|W| = n$.

Insertion Spelling Errors (ISEs) are committed whenever extraneous letters are placed within strings of characters that would have been correct spellings without such characters. The **ISE of a character** in W is described in Equation 1 and corrected via the delete-edit operation that is shown in Equation 2.

$$\begin{gathered} W \notin \Lambda \ \text{ and } \ \exists x_i \in W, 1 \leq i \leq n \\ f(W : x_i \to \varepsilon) = \bar{W} \\ \bar{W}| = n - 1, \ \bar{W} \in \Lambda \end{gathered} \tag{1}$$

$$f(W : x_i \to \varepsilon) = \bar{W} \tag{2}$$

Deletion Spelling Errors (DSEs) are committed whenever letters that are needed to correctly spell words are absent from strings of characters that are used to spell words such that their absences are the causes of misspellings. Equation 3 describes the **DSE of a character** in W, and Equation 4 shows the insert-edit operation for correcting it.

$$\begin{gathered} W \notin \Lambda; \exists x_j \in \Sigma, 1 \leq j \leq m, 1 \leq i \leq n - 1 \\ f(W : x_i \varepsilon x_{i+1} \to x_i x_j x_{i+1}) = \bar{W} \\ \bar{W} \in \Lambda, \ |\bar{W}| = n - 1 \end{gathered} \tag{3}$$

$$f(W : x_i \varepsilon x_{i+1} \to x_i x_j x_{i+1}) = \bar{W} \tag{4}$$

Substitution Spelling Errors (SSEs) occur when one letter that makes up a string of letters for a correct spelling is replaced with another letter, which leads to a misspelling. The **a single character SSE** in W is described in Equation 5, and a substitution edit operation (Equation 6) corrects it.

$$\begin{gathered} W \notin \Lambda \text{ and } \exists x_j \in \Sigma \\ x_i \in W, 1 \leq j \leq m, 1 \leq i \leq n-1 \\ f(W : x_i \to x_j) = \bar{W} \\ |\bar{W}| = n,\ \bar{W} \in \Lambda \end{gathered} \tag{5}$$

$$f(W : x_i \to x_j) = \bar{W} \tag{6}$$

Transposition Spelling Errors (TSEs) occur when the positions of two adjacent letters within strings of letters that make up words are swapped. The **transposition error of a single symbol** in W is described in Equation 7 and is corrected via the transpose edit operation that is shown in Equation 8.

$$\begin{gathered} W \notin \Lambda \text{ and } \exists x_i, x_{i+1} \in W, 1 \leq i \leq n-1 \\ f(W : x_i x_{i+1} \to x_{i+1} x_i) = \bar{W} \\ |\bar{W}| = n,\ \bar{W} \in \Lambda \end{gathered} \tag{7}$$

$$f(W : x_i x_{i+1} \to x_{i+1} x_i) = \bar{W} \tag{8}$$

Concatenation Errors are whitespace-related (non-print character) errors. These lead to errors in single words or in two words that follow one another, so these are divided into Types 1 and 2.

- Type-1 Concatenation Error: concatenation errors (Type 1) are committed when the spaces between two words are inadvertently left out so that each pair of words is merged to form a single non-word (or real word). If the outcome is a real word, then isolated-term spell checking will be unable to flag it as a spelling error.
- Type-2 Concatenation Error: concatenation errors (Type 2) are committed when single words are mistakenly broken into two with inadvertent introductions of whitespaces at some point, leading to the formation of non-words or real words (or both).
- A slightly different kind of error happens when a print character is substituted for a whitespace between two words, which leads to a Type-1+ concatenation error. We will not be addressing Type-1+ or Type-2 variations in this research.

A **concatenation error of two words into a single string** W is described in Equation 9 and is corrected via the concatenation edit operation that is shown in Equation 10, where $W = x_1 \cdots, x_i \cdots, x_n$; $x_i, x_{i+1} \in W$ and $|W| = n$.

Note that the whitespace between words $U\ and\ Y$ that were generated from W could just as well have been replaced by a hyphen.

$$\begin{gathered} W \notin \Lambda \text{ and } \exists x_i, x_{i+1} in W, 1 \leq i \leq n-1 \\ f(W : x_i \varepsilon x_{i+1} \to x_1, ..., x_i,\ x_{i+1}, ..., x_n) = U\ Y \\ |U| + |Y| = n, \quad U, Y \in \Lambda \end{gathered} \tag{9}$$

$$f(W : x_i \varepsilon x_{i+1} \to x_1, ..., x_i,\ x_{i+1}, ..., x_n) = U\ Y \tag{10}$$

3.2.3. Diacritic errors

According to [25], diacritics are used in Latin-based systems to supplement the basic Latin alphabet. Diacritics are used in *Yorùbá* orthography to indicate tones and to differentiate between two closely related phonemes. An accent mark is used to mark tones, while a dot-below mark distinguishes characters. [3] reported that more than 86% of words have their spelling errors originating from wrong or misapplications of diacritics. Thus, a diacritic error is any spelling error that is due to placing a wrong tone mark, not placing a tone mark or dot-below where it is required, or placing a tone mark or dot-below where it is not expected. Given a letter alphabet $\sigma(\subset \text{ of } \Sigma)$, a diacritic alphabet τ of tone-marks (including a null-tone mark) and a dot-below diacritic, and a diacritic dictionary $\mathcal{D}$ whose keys are drawn from strings of σ and whose values are drawn from sequences of τ. Note that $|\sigma| = 25$, while $|\tau\ | = 7$.

Define a word **W**:

$$\begin{gathered} \mathbf{W} = \mathbf{b} \circ \mathbf{d} : \\ \mathbf{b} = \begin{bmatrix} b_1 & \cdots & b_n \end{bmatrix} \\ \mathbf{d} = \begin{bmatrix} d_1 & \cdots & d_n \end{bmatrix} \end{gathered} \tag{11}$$

b base-form, letters sequence deprived of all diacritics
d the diacritic sequence that correlated with base-form
$\circ$ is the hadamard product operator.
(Example: $\begin{bmatrix} b_1 & b_2 \end{bmatrix} \circ \begin{bmatrix} d_1 & d_2 \end{bmatrix} = \begin{bmatrix} b_1 d_1 & b_2 d_2 \end{bmatrix}$)
Thus, a diacritic error occurred in **W** if, for **b**, **d** in **W**, $\exists\, \mathcal{D}(b)|$ d $\notin \mathcal{D}$(b).
Interpreted as: for a given word **W**, whenever **b** is found as a key in $\mathcal{D}$ but **d** is not found in the set of values for the key **b** in the diacritic dictionary $\mathcal{D}$, then a diacritic error has occurred. If **b** is not found as a key in $\mathcal{D}$, then an edit error has occurred.

$$f(\mathbf{W} : \mathbf{b} \circ \mathbf{d} \to \mathbf{b} \circ \mathcal{D}(b)) \tag{12}$$

(sorted by similarity) is a diacritic replacement operation.

Note that all consonants takes null diacritics except for “s” (which takes a dot-below → ṣ), “n” (which takes a low tone, high tone, and macron → ǹ, ń, and n̄), and “m” (which takes a low tone → m̀).

3.3. System design

The model for the spell checker was formulated as a system that was made up of a component for error detection in an input word, a second for generation, and the pruning of possible alternatives/candidates for a word that is detected to be in error. A ranking subsystem ranks the candidates before they are placed on the suggestion list as replacement options for the erroneous word. The model is comprised of algorithms, rule-sets, and language resources.

3.3.1. Model design

The major language resources (which we developed from scratch for the spell checker) are the diacritic dictionary and the language's word-rank dictionary (which we named the language unigram model). The *Yorùbá* orthography consists of the letters of a modified Latin alphabet and tone marks. The alphabet is comprised of 21 letters, but the full repertoire of characters with tone marks is 45 (as shown in Table 6).

An architectural model of the spell checker that shows the components that are described above is shown in Figure 1.

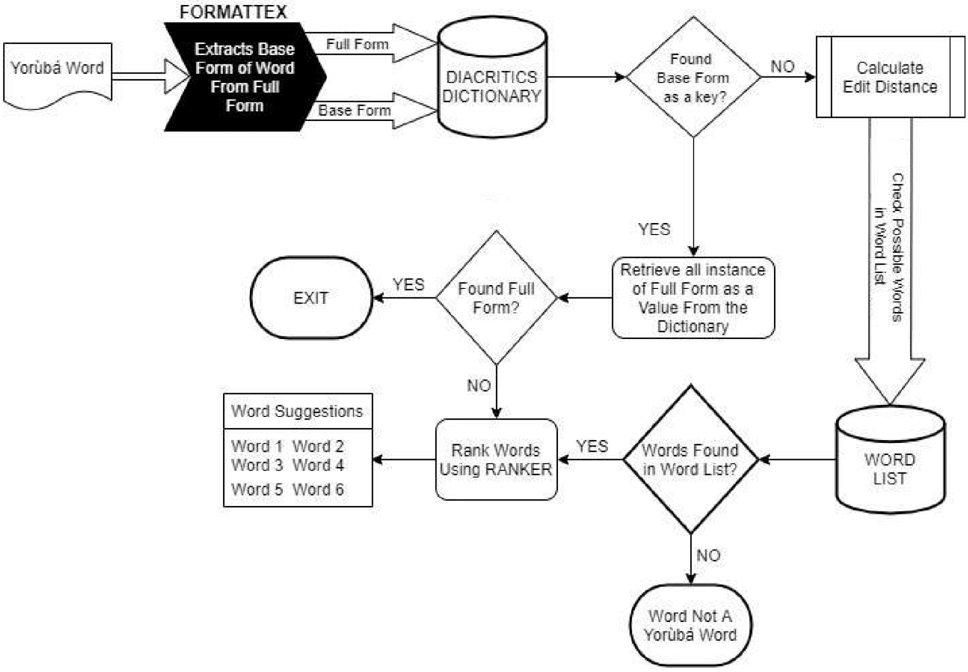

Figure 1. Spell checker system model

Table 6
Full repertoire of graphemes in *Yorùbá* text

Consonants	: :	b, d, f, g, gb, h, j, k, l, m, n, p, r, s, ṣ, t, w, y
Vowels* + low tone	:	à è ẹ̀ ì ò ọ̀ ù ǹ m̀
Vowels* + mid tone	:	a e ẹ i o ọ u n̄ m̄
Vowels* + high tone	:	á é ẹ́ í ó ọ́ ú ń ḿ

Vowels* $\Longrightarrow$ vowels and syllabic consonants

3.3.2. Diacritic dictionary

The diacritic dictionary was created as a key-and-value pair, where the key is made up of words and letters from the *Yorùbá* alphabet, and the value for each key is a list that is made up of valid *Yorùbá* words whose base letter forms (hence, base forms) are equivalent to the string in the key when all of the diacritics (tone marks and dot-belows) that are present in these words are removed. Some words without any of its diacritics (i.e., words in which both tone marks and dot-belows have been removed) can derive up to seven valid words with different combinations of diacritics. Table 7 shows an illustration of the diacritic dictionary.

Table 7
Sample table model of diacritic dictionary

key≡ x_i	List of Values
agbalagba	àgbàlagbà
akoko	akòko àkókò àkókó àkọ́kọ́
baba	baba bàbà bàbá
igba	igba igbà igbá ìgbà ìgbá
oko	oko okó òkò ọkọ ọkọ̀ ọkọ́ ọ̀kọ̀

3.3.3. Language unigram model

The language unigram dictionary (otherwise called the language unigram model) is a simple dictionary where a lexical word with its full diacritics is the key and the frequency count of the word as derived from a corpus is the value. This is the resource that an input word is initially checked against to determine whether the spelling is correct. If the input word exists as a key in the language unigram model, its spelling is judged to be correct; otherwise, the spelling is invalid.

3.4. System implementation

Based on our experiences in processing Yoruba text, we have noted that digital text with the same visual appearance might have different underlying representations for the characters. Thus, we implemented a pre-processing module that takes the text input and normalizes it into a single common representation before passing it to the spellchecker. For example, ọ́ may be represented with any of the following codepoints: u1ECD+u0301, u00F3+u0323, u006F+u0323+u0301, or u006F+u0301+u0323. Our pre-processing module (Formattex) handles these and normalizes them to u1ECD+u0301. Similar situations include all tone-marked vowels and syllabic consonants and letters with dot-below diacritics.

The module that handles the edit operations actual performs an inverse edit upon an input string that is determined to be a misspelling. The module implements the edit operations that are defined in Equations 2, 4, 6, and 8. Separate modules implement the concatenation-edit and diacritic-correction operations that are defined in Equations 10 and 12, respectively. A filter module that uses a simple regular expression prunes the number of strings from the edit operation, removes strings that are incompatible with Yoruba word structure, and reduces the checking edit module by between 30.6 to 97.5% (mean: 69.7%). The ranking module takes the output from both the diacritic-correction and -edit modules to generate the suggestion list.

4. Experimental setup

The setup for the experiments to evaluate the performance of the spellcheck were divided into two parts (as follows):

4.1. Experimental setup 1

a. The first setup was for evaluating the spell checker with different configurations of the used dictionary. The test data that was used for this aspect of the experiment contained only 100 words. These words were randomly selected from a pool of 300 words in the test database.

- **Setup 1A** In one experiment, we used only the unigram language model-based dictionary. This was to test how much of the performance of the spell checker was due to the unigram model.
- **Setup 1B** In the second experiment, we substituted the unigram language model dictionary with the diacritic language model dictionary and evaluated the performance of the spell checker.
- **Setup 1C** We then combined the two different models via a multiple search for each word in both dictionaries – both for the checking the correct spelling and making suggestions. The performance was also evaluated.

4.2. Experimental setup 2

The second setup was to evaluate the impact of the dictionary sizes on the system performance of the spell checker. The corporal was used to build the different unigram and diacritic language models; this was the driving engine of the spell checker. The test data consisted of all of the 300 words in the test database.

- **Setup 2A** The first experiment was with a spell checker whose models were built from a randomly selected corpus of the full corpus that generated 1000 unique words.
- **Setup 2B** The second experiment was with a spell checker whose models were built from the full corpus that generated 1700 unique words.

An evaluator module was implemented over the spell checker; this takes a file that contains the words to be checked as input and produces an output file that contains the evaluation result. The evaluator output contains the wordlist in the input file such that each word is tagged with a label that indicated whether the word was correctly spelled or not. In addition, the suggestions were also listed in front for those words that were determined to be misspellings in an order of most likely to least likely.

4.3. Performance metrics

We proposed a definition of a metric for spelling-detection accuracy to be used in evaluating spell checkers. Spelling-detection accuracy (or detection accuracy, for short) will be defined as the harmonic mean of the precision and recall (which will be calculated slightly differently from existing norms).

1. The precision had hitherto being calculated was premised on TP, which measures the performance of the spell checker in recognizing the valid spelling of a word; it does not measure the ability to correctly recognize misspellings as invalid words. We created a second computation for such a recognition of invalid words; thus, we have $Precision_+$ (P_+) (as defined in Equation13) and $Precision_-$ (P_-) (as defined in Equation 14) for recognizing valid (TP) and invalid (TN) words as related to what the spell checker considers to be valid (TP+FP) and invalid (TN+FP), respectively. Precision (P) is then the weighted average of P_+, and P_-. N = TP + TN + FP +FN
2. Similarly, we adjusted our definition of the recall as the ability of the spell checker to correctly recognize valid words (TP) and invalid words (TN) in the midst of the total number of valid words and total invalid words, respectively. Thus, we define $Recall_+$ (R_+) as the (TP) proportion to the number of words that are correctly recognized in the text (TP+TN) and $Recall_-$ (R_-) as the (TN) proportion of the words that are recognized as incorrect (TN+FN). The recall determines how the spell checker is able to recognize the number of words that were valid in the text (true positives) as related to the number of words that were correct in the text.

3. The suggestion adequacy (SA) was determined by summing the score that was allocated to the suggestion that was made for each word in the test document (where S is the score for a suggestion) and dividing it by the total number (N) of all of the negatives (TN +FN). Using our own metric: if a valid suggestion occurs within the suggested list, it is graded as a 1; no suggestion at all scores a 0 (and wrong suggestions are not penalized).

The measurement parameter notations that were used in the computation of the detection accuracy (DA) and suggestion adequacy were as follows:

$$P_+ = \frac{TP}{TP + FP}; \; R_+ = \frac{TP}{TP + FN} \tag{13}$$

$$P_- = \frac{TN}{TN + FN}; \; R_- = \frac{TN}{TN + FP} \tag{14}$$

$$w_+ = \frac{TP + FN}{N}; \; w_- = \frac{TN + FP}{N} \tag{15}$$

$$\begin{aligned} P &= P_+ \cdot w_+ + P_- \cdot w_- \\ R &= R_+ \cdot w_+ + R_- \cdot w_- \end{aligned} \tag{16}$$

$$Detection\ Accuracy\ (DA) = 2 \cdot \frac{P \cdot R}{P + R} \tag{17}$$

$$Suggestion\ adequacy\ (SA) = \frac{\sum S}{N} \tag{18}$$

5. Results

The results for the experimental setups that evaluated the contribution of the different dictionary components is presented in Table 8; the computations of the metrics follow each table. Thereafter, we also presented the impact of the size of the dictionary on speller performance using Table 9.

5.1. Experiment 1 results

Two dictionaries (the unigram danguage dictionary [ULD], and the diacritic dictionary [DD]) formed the major resources that were used by the spell checker. To find the contribution of each dictionary, the spell checker was used with only that dictionary, and the performance for spell checker was measured. The test data that was used in this experiment consisted of 100 words, and the percentages of the error in it were between 53 to 59% (since the data was randomly chosen for each test). We did not adjust for difference in the error size, as stability requires a much larger test data set. Table 8 shows the performance indices for using ULD alone, DD alone, and with both ULD and DD working together as proposed in the system. Working with ULD alone had the worst performance indices – represented by vector (DA, SA) = (93.23%, 26.94%). Overall, the model that combined ULD and DD had the best DA (95.01%), while the model that utilized DD had the best SA (72.10%).

Table 8
Spell checker performance for different dictionary configurations

	Unigram	Diacritic	Combined
P_+ [%]	87.96	91.11	93.75
R_+ [%]	97.56	95.35	95.75
w_+	0.41	0.43	0.47
P_- [%]	98.15	96.36	96.15
R_- [%]	89.83	92.98	94.34
w_-	0.59	0.57	0.53
P_w [%]	93.56	94.10	95.02
R_w [%]	93.00	94.00	95.00
DA [%]	93.23	94.10	**95.01**
SA [%]	26.94	**72.10**	65.89

5.2. Experiment 2 results

The second set of experiments measured the impact of the corpus size that was used to build the dictionaries on the spell checker's performance using the model that combined ULD and DD. Three hundred words were used as the test data for both of the different corpus sizes. The results are as is shown in Table 9. As expected, the spell checker whose dictionaries were built from a larger corpus had better performance indices. However, both performed worse than the first set of the experiment that used the combination of ULD and DD (except in the SA for the larger corpus). This was despite the fact that the error size was less than 50% in these experimental setups.

Table 9
Spell checker performance for dictionaries vs. corpus sizes

	Corpus size = 1k	Corpus size = 1.7k
P_+ [%]	90.54	89.81
R_+ [%]	86.45	90.97
w_+	0.52	0.52
P_- [%]	86.18	90.21
R_- [%]	90.34	88.97
w_-	0.48	0.48
P_w [%]	88.45	90.00
R_w [%]	88.32	90.00
DA [%]	88.38	90.00
SA [%]	63.93	67.33

5.3. Discussion

The first experimental setups (1A, 1B, and 1C) did not not show many differences in the detection accuracy for the different dictionary configurations for the spell checker. The differences could be due to the slight difference in the data as reflected in the absolute number of errors in each test data. However, the suggestion adequacy of Setup 1B with the diacritic dictionary component was much higher than in Setup 1A (which used the unigram language model) and still better than both combined. This can be only explained by the ability of the system to focus more on correcting diacritic errors, which [3] reported to have accounted for more than 86% of all of the spelling errors in the *Yorùbá* text. In fact, a simple examination of the 65.89% SA for the combined dictionary setup is close to the simple addition of 86% of the performance of the diacritic dictionary setup and 14% of the performance of the unigram language model setup that predicted a value of 65.78%.

The second experimental setup (2A and 2B) followed the expected pattern, as the spell checker that used the dictionaries from the larger corpus performed better than the one with the dictionaries from the smaller corpus. However, the increased performance was again minimal on the detection accuracy index as compared to the suggestion adequacy. This seems to suggest that the addition of more data affected the ability of the spell checker to suggest more valid candidates than its ability to detect more errors. It is likely that its failure to detect more misspellings was due to the spelling errors in the data that was used to build the dictionaries. This may be supported by the fact that, when the size of the test data is multiplied by a factor of three, there is a general decrease in DA.

It is a well-known fact that not only the size of the data but also the quality of the data affects the performance of spell checkers; this was not an exception, and efforts are ongoing in this direction.

6. Conclusion

The study concluded that, while increases in training data does not give a linear proportional increase in the performance of the spell checker, each of the sub-models in the dictionary played different roles in the spell-checking process. Spell checking *Yorùbá* text requires the unigram language model and diacritic dictionary to achieve more-robust result. This system has been designed with the assumption of human intervention in the selection of the most appropriate suggestions.

To enhance the performance of the detection accuracy, a cleaner data set needed to be deployed in the building of the dictionaries. Finally, the auto correction of spelling errors will be more practical if a bigram or trigram language model is used in ranking the suggested candidates instead of a simple frequency list (as was adopted here). All of these suggestions will be considered in future works.

Acknowledgements

The research that was presented in this paper was partially supported by the Africa Center of Excellence ICT-Driven Knowledge Park (ACE OAK Park).

References

[1] Ahmed F., De Luca E., Nürnberger A.: Revised N-Gram based Automatic Spelling Correction Tool to Improve Retrieval Effectiveness, *Polibits*, vol. 40, pp. 39–48, 2009. doi: 10.17562/PB-40-6.

[2] Arndt E., Foorman B.: Second Graders as Spellers: What Types of Errors Are They Making?, *Assessment for Effective Intervention*, vol. 36(1), pp. 57–67, 2010. doi: 10.1177/1534508410380135.

[3] Asahiah F., Oníf́ádé T., Adégùnlẹ̀hìn A.: Spelling Error Patterns in Typed Yorùbá Text Documents, *IJ Information Engineering and Electronic Business*, vol. 6, pp. 28–30, 2020.

[4] Asahiah F.O., Ọdẹ́jobí O.A., Adagunodo E.R.: Restoring Tone-Marks in Standard Yorùbá Electronic Text: Improved Model, *Computer Science*, vol. 18(3), pp. 301–315, 2017. doi: 10.7494/csci.2017.18.3.2128.

[5] Awoyale Y.: The LDC Corpus Catalog (Global Yoruba Lexical Database v. 1.0), Web, 2008. Retrieved Decemeber 04, 2020 from http://www.language-archives.org/item/oai:www.ldc.upenn.edu:LDC2008L03.

[6] Balogun T.A.: An Endangered Nigerian Indigenous Language: The Case of Yorùbá Language, *African Nebula*, vol. 6, pp. 70–83, 2013.

[7] Bassil Y., Alwani M.: Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information, *Computer and Information Science*, vol. 5(3), 2012. doi: 10.5539/cis.v5n3p37.

[8] Colesnicov A.: The Roumanian spelling checker ROMSP: the project overview, *Computer Science Journal of Moldova*, vol. 3, pp. 40–54, 1995.

[9] Coulmas F.: *Writing systems: An introduction to their linguistic analysis*, Cambridge University Press, 2003.

[10] Daffern T., Mackenzie N.M., Hemmings B.: Predictors of writing success: How important are spelling, grammar and punctuation?, *Australian Journal of Education*, vol. 61(1), pp. 75–87, 2017. doi: 10.1177/0004944116685319.

[11] Enikuomehin A.O.: A computerized identification system for verb sorting and arrangement in a natural language: Case study of the Nigerian Yoruba Language, *European Journal of Computer Science and Information Technology*, vol. 3(1), pp. 43–52, 2015.

[12] Federal Government of Nigeria: Official Gazette of Federal Republic of Nigeria, Web, 2007. Retrieved March 12, 2021 from http://www.population.gov.ng/pop_figure.pdf.

[13] Federal Ministry of Education, Nigeria: *The 1974 Revised Official Orthography for the Yoruba Language (Joint Consultative Committee on Education)*, The Committee, 1974.

[14] Hema P., Sunitha C.: Spell Checker for Non Word Error Detection: Survey, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 5(3), pp. 30–38, 2015.

[15] Hok P.: *Khmer Spell Checker*, Master's thesis, Australian National University 2005.

[16] Kaur A., Singh P., Rani S.: Spell Checking and Error Correcting System for text paragraphs written in Punjabi Language using Hybrid approach, *International Journal of Latest Trends in Engineering and Technology (IJLTET)*, vol. 2(4), pp. 156–160, 2014.

[17] Mishra R., Kaur N.: A Survey of Spelling Error Detection and Correction Techniques, *The International Journal of Computer Trends and Technology*, vol. 4(3), pp. 372–374, 2013.

[18] Moats L.: *Spelling: Development, disability, and instruction*, York Press, Baltimore, 1995.

[19] Naseem T.: *A Hybrid Approach for Urdu Spell Checking*, Ph.D. thesis, National University of Computer & Emerging Sciences, 2004.

[20] Padhy H.H., Mohanty S.: Designing hybrid approach Spell checker for Oriya, *International Journal of Latest trends in Engineering and Technology (IJLTET)*, vol. 2(4), pp. 156–159, 2013.

[21] Rajashekara M.S., Vadiraj M., Sachin D., Ramakanth K.P.: A Non-Word Kannada Spell Checker Using Morphological Analyzer And Dictionary Lookup Method, *International Journal of Engineering Sciences and Emerging Technologies*, vol. 2(2), pp. 45–52, 2012.

[22] Renkema J.: Improving the Quality of Governmental Documents: A Combined Academic and Professional Approach. In: W. Cheng, K.C.C. Kong (eds.), *Professional Communication. Collaboration between Academics and Practitioners*, pp. 173–190, Hong Kong University Press, 2008.

[23] Salifou L., Naroua H.: Design of A Spell Corrector For Hausa Language, *International Journal of Computational Linguistics (IJCL)*, vol. 5(2), pp. 14–26, 2014.

[24] Sawyer D., Wade S., Kim J.: Spelling errors as a window on variations in phonological deficits among students with dyslexia, *Annals of Dyslexia*, vol. 49, pp. 135–159, 1999.

[25] Wells J.: Orthographic diacritics and multilingual computing, *Language Problems & Language Planning*, vol. 24(3), pp. 249–272, 2000. Retrieved July 12, 2010 from http://www.phon.ucl.ac.uk/home/wells/dia/diacritics-revised.htm.

Affiliations

Franklin Ọládiípọ̀ Asahiah
Obafemi Awolowo University, Ile-Ife, Nigeria, sobsuola@oauife.edu.ng

Mary Taiwo Onífádé
Kings University, Gbongan-Osogbo Road, Ode-Omu, Nigeria,
tm.onifade@kingsuniversity.edu.ng,
Obafemi Awolowo University, Ile-Ife, Nigeria, maryonifade@pg-student.oauife.edu.ng

Adekemisola Olufunmilayo Asahiah
Obafemi Awolowo University, Ile-Ife, Nigeria, asahiahkemi@oauife.edu.ng

Abayomi Emmanuel Adegunlehin
Obafemi Awolowo University, Ile-Ife, Nigeria, eadegunlehin@pg-student.oauife.edu.ng

Adekemi Olawunmi Amoo
Obafemi Awolowo University, Ile-Ife, Nigeria, aamoo@oauife.edu.ng

Received: 28.09.2021
Revised: 28.03.2022
Accepted: 24.11.2022