Indonesian Journal of Electrical Engineering and Computer Science
Vol. 34, No. 1, April 2024, pp. 322~332
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v34.i1.pp322-332

Implementation of an Arabic spell checker

Rafik Kassmi, Samir Mbarki, Abdelaziz Mouloudi
Department of Computer Science, Faculty of Science, Ibn Tofail University, Kénitra, Morocco

Article Info

Article history:

Received Nov 19, 2023
Revised Jan 15, 2024
Accepted Jan 17, 2024

Keywords:

Arabic language
El-DicAr dictionary
Local grammar
Morphological grammar
NooJ platform
Spell checker
Spelling error

ABSTRACT

This paper outlines the implementation of a spell checker for the Arabic language, leveraging the capabilities of NooJ and its functionality, specifically noojapply. In this paper, we shall proceed to provide clear definitions and comprehensive descriptions of several categories of spelling errors. Next, we will provide a comprehensive introduction to the NooJ platform and its command-line utility, noojapply. In the subsequent section, we shall outline the four main phases of our spell checker prototype. We intend to develop a local grammar in NooJ for the purpose of error detection. Afterwards, a morphological grammar and a local grammar will be created in NooJ with the aim of providing an exhaustive list of possible corrections. Following that, a revised algorithm will be employed to arrange these candidates in descending order of ranking. Subsequently, a web user interface will be developed to visually represent our research efforts. Finally, we will proceed to showcase a series of tests and evaluations conducted on our prototype, Al Mudaqiq.

Corresponding Author:

Rafik Kassmi
Department of Computer Science, Faculty of Science, Ibn Tofail University
Kénitra, Morocco
Email: rafik.kassmi@gmail.com

1. INTRODUCTION

Arabic is a Semitic language with a complex morphology. “The general rule for Arabic is that everything is at least five times more complicated than any European language” [1]. The Arabic alphabet consists of 28 letters, one of which is the long vowel ا /*ā*/. Two letters of them, ي /*y*/ and و /*w*/, represent the long vowels ي /*ī*/ and و /*ū*/, respectively. Depending on their placement in the sentence, glides can be considered consonants such as وَعَدَ /*wa'ada*/, which means “he promises,” or long vowels such as دُرُوسٌ /*durūsun*/, which means “lessons.” The Arabic language also has diacritics, which are brief vowels that are marked above or below the consonants they follow: (◌َ) /*a*/, (◌ِ) /*i*/, (◌ُ) /*u*/, (◌ْ) /*Ø*/, and (◌ّ) /*šadda*/. The short vowels are seldom used in written Arabic. The linguistic phenomenon known as tanwīn relates to the representation of indefinite words without an article or complement. This is achieved by duplicating the short vowels ((◌ً) /*an*/, (◌ٍ) /*in*/, and (◌ٌ) /*un*/) and positioning them towards the end of the word.

The Arabic language is written in a right-to-left direction. The script has a semi-cursive nature, whereby the individual letters possess the ability to link with one another and undergo transformations in shape based upon their position inside a given word, whether at the beginning, middle, or end. Nevertheless, it is worth noting that there are six letters in the Arabic language that do not undergo any kind of linking to the subsequent letter. The letters in question are ا /*ā*/, و /*w*/, ر /*r*/, ز /*z*/, د /*d*/, and ذ /*ḏ*/. A series of graphemes separated by two blanks make up words. Each grapheme represents a form or unit that is likely to show up under a lexical entry or lemma [2]. Arabic is a generative language in which most words, also referred to as lemmas, are derived from a root, also known as a radical, while adhering to a scheme. This pertains to the grammatical categories of verbs, nouns, and a few particles. A root is composed of two, three, or four letters, but approximately 64% of Arabic roots are composed of three letters [3]. In specific cases, particularly for nouns, the root can be made up of more than four letters. These letters are the foundation of the word [4]. A collection of words can be generated from a single root via various schemes [5]. Table 1 shows an example of generated words from the root كتب /*ktb*/.

Table 1. Example of generated words from the root /*ktb*/

Scheme	كتب /*ktb*/	Translation
فَعَلَ /*fa'ala*/	كَتَبَ /*kataba*/	He wrote
فَاعَلَ /*faā'ala*/	كَاتَبَ /*kaātaba*/	He wrote someone
فَاعِلٌ /*faā'ilun*/	كَاتِبٌ /*kaātibun*/	Writer
مَفْعَلٌ /*maf'alun*/	مَكْتَبٌ /*maktabun*/	Desk
فِعَالٌ /*fi'aālun*/	كِتَابٌ /*kitaābun*/	Book
مَفْعُولٌ /*maf'ūlun*/	مَكْتُوبٌ /*maktūbun*/	A writing
مَفْعَلَةٌ /*maf'alatun*/	مَكْتَبَةٌ /*maktabatun*/	Library
فِعَالَةٌ /*fi'aālatun*/	كِتَابَةٌ /*kitaābatun*/	Writing

The term "agglutination" refers to complex words made up of numerous morphemes attached together to provide a variety of morpho-syntaxis information. Arabic is very agglutinative, which means that a lot of different affixes-proclitic, prefix, suffix, and enclitic-can be added to each word, making the vocabulary bigger [6]. For instance, the word أَوَستأكلونه /*ʾawasataʾkulūnahu*/, which means "will you eat it?" is decomposed like (see Table 2).

Table 2. Example of segmentation

هُ	ونَ	أكُلُ	تَ	أوَسَ
/*hu*/	/*ūna*/	/*ʾkulu*/	/*ta*/	/*ʾawasa*/
It		Eat		Will you
Enclitic	Suffix	Lemma	Prefix	Proclitic

During the first century of the Hijrah, Arab researchers began studying phonetics in conjunction with other linguistic disciplines, including grammar, lexicography, and rhetoric. The basis of these studies originated from the Quran, with the aim of safeguarding its text integrity against any distortions. During that period, its primary manifestation was seen in the field of tajwid, which pertains to the accurate recitation of the Quran [7]. However, the study of phonetics gained prominence in the fourth century of the Hijrah [8]. Arab linguists have classified Arabic sounds in terms of both point of articulation (مَخْرَج /*maẖraj*/) and manner of articulation (صِفَة /*ṣifah*/) [9]. There are three groups of sounds in Arabic: the first group is plosives or stops شَدِيدَة /*šadīdah*/, which is made up of eight consonants as seen in Table 3. The second group is resonant رَنَّانَة /*rannāna*/, which has four manners of articulation: three nasal sounds, one lateral sound, one trill sound, and two glide sounds, as in Table 4. The third group is fricative رِخْوَة /*riẖwa*/ and is made up of 14 sounds: three are voiceless, and five are voiced, as seen in Table 5.

Table 3. List of plosive sounds

	Voiceless	Voiced
bilabial		ب /*b*/
alveolar non emphatic	ت /*t*/	د /*d*/
alveolar emphatic		ط /*ṭ*/
palatal		ج /*ğ*/
velar	ک /*k*/	
uvular		ق /*q*/
glottal	ء /*ʾ*/	

Table 4. List of resonant sounds

	Bilabial	Alveolar non emphatic	Alveopalatal
nasals	م /*m*/	ن /*n*/	
lateral		ل /*l*/	
trill		ر /*r*/	
glides	و /*w*/		ي /*y*/

Table 5. List of fricative sounds

	Voiceless	Voiced
labiodental	ف /f/	
interdental non emphatic	ث /ṯ/	ذ /ḏ/
interdental emphatic		ظ /ẓ/
alveolar non emphatic	س /s/	ز /z/
alveolar emphatic	ص /ṣ/	ض /ḍ/
alveopalatal	ش /š/	
uvular	خ /ẖ/	غ /ġ/
pharyngeal	ح /ḥ/	ع /ʿ/
glottal	هـ /h/	

In Arabic, phonological change refers to the transformation of a word's primary form into a derived form to facilitate pronunciation [10]. We can differentiate among the three primary categories [11]. The first category is assimilation إِدْغَام /*ʾidġām*/, which is the germination of one letter with another. It's the emphasis of two similar sounds, and it can be written as a single letter with the short vowel (ّ) /*šadda*/ on top of it [12]. Consider, for instance, the past-tense singular first-person singular form of the pattern فَعَلْتُ /*faʿaltu*/ of the verb أَبَتَ /*abata*/, which means "heat up." The phonological rule states that for verbs ending with ت /*t*/, if the first ت /*tØ*/ is unvowelled and followed by a vowelled تُ /*tu*/, only one /*t*/ carrying /*šadda*/ is retained (+ تْ تُ → تُّ). The second category is substitution إِبدَال /*ʾibdāl*/, which is removing a letter and replacing it with another. This phenomenon is seen in verbs representing the pattern إِفْتَعَلَ /*ʾiftaʿala*/ [13]. Consider, for instance, the underlying structure إِدْتَعَى /*ʾidtaʿā*/ of the root دعى /*dʿā*/. According to the rule, the د /*d*/ of /*ʾiftaʿala*/ replaces the ت /*t*/ of the root if the first radical is د /*d*/. Then, the surface structure is إِدَّعَى /*ʾiddaʿā*/, which means "claimed." The third category is weakening إعلَال /*ʾiʿlāl*/, which is a transformation that occurs on long vowels ا /*ā*/, ي /*ī*/ or و /*ū*/, glides و /*w*/ or ي /*y*/, or a /*hamza*/ letter (glottal stop) ء /*ʾ*/. It is composed of three types. The first kind is called glide metathesis إِعْلَال بِالقَلْب /*ʾiʿlāl bil-qalb*/, and it involves replacing a long or short vowel, a glide, or a /*hamza*/ letter with one of the other two. Take the word أَعْدَاوٌ /*ʾaʿdāwun*/, for instance, which comes from the plural أَفْعَالٌ /*ʾafʿālun*/ of the root عدو /*ʿdw*/, which stands for "to feel hatred."

According to the phonological rule, a glottal stop ء /*ʾ*/ will take the place of the glide و /*w*/ if the word's extremity is a long vowel ا /*ā*/ with (َ) /*fatḥa*/. As a result, the structure surface became أَعْدَاءٌ /*ʾaʿdāʾun*/, which means "enemies." The second type, called glide transfer إعلَال بِالنَّقل /*ʾiʿlāl bil-naql*/, involves taking a short vowel out of one sound and transferring it to another. Let's consider the word يَقْوُلُ /*yaqwulu*/. It's formed from the pattern يَفْعُلُ /*yafʿulu*/ of the root قول /*qwl*/, which refers to the verb "to say." According to the phonological rule, the short vowel (ُ) /*ḍamma*/ above the glide و /*w*/ after a vowel (ْ) /*sukūn*/ is removed and replaced with the sukūn for hollow verbs. As a result, the surface structure changed to يَقُولُ /*yaqūlu*/, which means "he says." Finally, glide elision is the third kind إعلَال بِالحَذْف /*ʾiʿlāl bil-ḥaḏf*/, which removes a glide, a /*hamza*/ letter, a long or short vowel, or both. Take the word يَوْعِدُ /*yawʿidu*/, which is formed from the pattern يَفْعِلُ /*yafʿilu*/ of the root وعد /*wʿd*/, which means "promise." According to the phonological rule, assimilated verbs should not have the glide و /*w*/ with the vowel (ْ) /*sukūn*/ above it and a vowel (ِ) /*kasra*/ following it. As a result, the surface structure changed to يَعِدُ /*yaʿidu*/, which means "he promises."

Texting, e-mailing, composing documents, and browsing for information on the Internet are all examples of the increasing importance of writing in our daily lives. Even the most talented among us are susceptible to typing errors for a variety of reasons, including unfamiliarity with the word, fatigue, lack of concentration, and poor keyboard control. A spell checker is a program that analyzes words to identify and correct misspellings [14]. It provides alternative spelling suggestions when dubious of the correct spelling. It is used both independently and as an embedded component in a wide variety of applications, including machine translation, optical character recognition, search engines, and word processors. The spell checker may be either interactive or automatic. Once the interactive spell checker has identified misspelled words and suggested possible corrections for each, the user can select the right choice. The automated spell checker, on the other hand, replaces misspelled words with their most likely alternative spellings without requiring user input. Spell checkers are built into almost all software today, but they are not always accurate, at least not for all languages and especially not for Arabic. Our study's objective is to develop an Arabic spelling checker by leveraging the NooJ platform and its command-line functionality, namely noojapply [15].

There are three main categories in which spelling mistakes may be classified: typographical errors, cognitive errors, and phonetic errors [16]. In some instances, however, it is even challenging to designate a single category for particular errors. Typographical errors include all instances of typing errors resulting from improper manipulation of the keyboard, including instances of hitting an erroneous key or using a malfunctioning keyboard. Hence, the author demonstrates an error in their work by using a term while possessing knowledge of its correct spelling.

Approximately 80% of spelling errors concerning non-existent words in the language, known non-words, are classified as single errors [17]. These single errors may be further classified into four distinct forms, namely insertion, deletion, substitution, and transposition. An insertion error is a linguistic phenomenon that arises when a writer mistakenly includes an additional letter in a word. An instance of the extra insertion of the letter ت /*t*/ may be seen when typing مكتتوب /*makttūb*/ instead of مكتوب /*maktūb*/, which means "written." A deletion error occurs when the author mistakenly omits one of the letters inside a word. As an example, in the case of typing مدسة /*madsah*/ instead of مدرسة /*madrasah*/, which means "school," the letter ر /*r*/ has been omitted. A substitution error occurs when the writer erroneously substitutes the right letter of a word with an incorrect letter. More precisely, 58% of substitution errors involve adjacent keys of the keyboard [18]. For instance, the letter ق /*q*/ is mistakenly replaced with the letter ف /*f*/ while entering حديفة /*ḥadifah*/ for حديقة /*ḥadiqah*/, which means "garden." A transposition or permutation error is a linguistic phenomenon in which the writer exchanges the positions of letters inside a word. As an example, in the case of typing برح /*barḥ*/ instead of بحر /*baḥr*/, which means "see," there is an interchange of the positions of the letters ح /*ḥ*/ and ر /*r*/. Cognitive errors include situations when the writer lacks knowledge of the accurate spelling of a word, has a lapse in memory about it, or holds a mistaken understanding of it. An instance of introducing an unexpected letter, such as adding ا /*ā*/, might occur while typing لاكن /*lākin*/ instead of لكن /*lakin*/, which means "but."

Mispronunciation is well recognized as a prevalent contributing factor to spelling errors. Consequently, the mispronunciation of a word always results in a phonetic inaccuracy in its spelling. Hence, this particular category encompasses errors that arise when the author replaces a word with another one that sounds similar. For instance, when typing عضيم /*ʿaḍim*/ for عظيم /*ʿaẓim*/, which means "great," the letter ض /*ḍ*/ is mistakenly replaced with ظ /*ẓ*/.

NooJ is a linguistic development platform that is used for the purpose of formalizing natural languages [19]. The software offers a range of resources for constructing, evaluating, and managing highly structured representations of natural languages. Additionally, it facilitates the creation of automated applications for natural language processing (NLP), including but not limited to machine translation, text analysis, semantic annotation, grammar and syntax verification, and recognition of named entities. NooJ constructs dictionaries and a structured collection of graphs that depict grammatical structures. These linguistic resources may be used for the purpose of identifying morphological (inflection and derivation), lexicological (spelling variants), syntactic, and semantic patterns within texts. The software has the capability to do a wide range of statistical analyses in the fields of corpus linguistics and digital humanities, in addition to its utility in teaching students in the areas of linguistics and linguistic computing [20].

One notable advantage of NooJ is its use of a command-line program named "noojapply.exe" to provide the majority of its functionalities. This program may be invoked from a simple shell script or other programs using a system command. Hence, the first step involves the user's creation of dictionaries and various grammars on the NooJ platform, which are afterwards used for direct application to texts via the noojapply function.

2. METHOD

The spell checker we propose has four primary steps, as seen in Figure 1. The first step is to use the El-DicAr dictionary and our morphological grammar, which is built in NooJ, to find all of the non-lexical words in a given text corpus. Next, the second phase involves the generation of corrections or suggestions for candidates by using morphological and local grammars within the NooJ framework. Next, arrange the possibilities in a decreasing order based on the highest probability of the right word being suggested. Lastly, the system chooses the most appropriate choice as the correction.

Figure 1. The main steps of our spell checker

2.1. Detecting errors

The error detection technique used in our study is based on a dictionary lookup approach specifically designed to identify isolated non-words. For this purpose, we have opted to utilize the El-DicAr dictionary [21]. It is a morpho-syntactic analyzer designed to recognize named entities as well as a lemma-based dictionary of the standard Arabic language. It is available for free on the official NooJ website [22].

We combined a morphological grammar that could recognize Arabic agglutination with a local grammar [23] that exclusively retrieved unknown words i.e., words that are not present in the dictionary.

Consider the following sentence in Figure 2: كضف باحث بريطاني عن هذا في دراشة متيرة /*kaḍafa bāḥiṯun brīṭānī ʿan haḏā fī dirāšah mutirah*/, which means "A British researcher revealed this in an exciting study." When our local grammar is applied to this sentence, we detect three misspelled words: كضف /*kaḍafa*/, دراشة /*dirāšah*/, and متيرة /*mutirah*/.

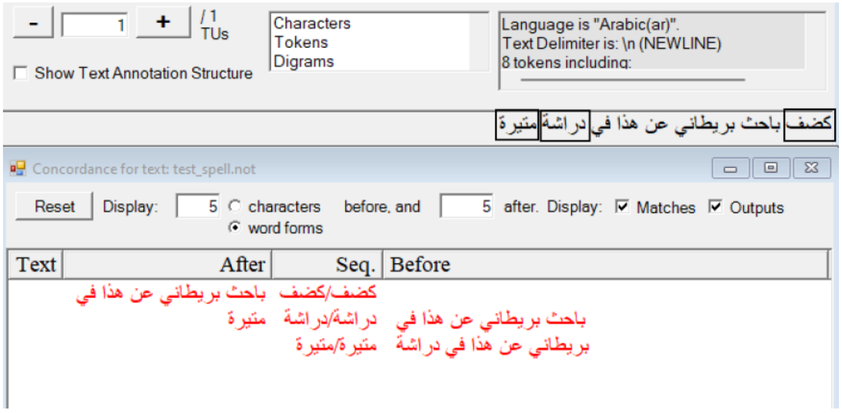

Figure 2. The three misspelled words identified by the NooJ concordance tool

2.2. Generating corrections

The previous section showed the detection of misspelled words by using a combination of morphological and local grammars. Through this phase, we produced all the valid candidates arising from these misspelled word transformations. In order to achieve this goal, we implemented three sequential steps. Initially, we have gathered an exhaustive list of neighboring letters for each Arabic letter. Subsequently, we have developed a morphological grammar that performs the four editing operations by using the previous list. Finally, we have created a local grammar that, based on the output of the morphological grammar, generates a list of potential candidates for every misspelled word.

2.2.1. List of neighboring letters

We take into consideration the neighboring letters in order to optimize the editing operations that are involved in the correction process and to prevent the generation of invalid candidates (non-words). Letters of adjacent keys on the keyboard, typographical letters that have similar shapes [24], or phonetic letters [25] that have a similar sound are all examples of neighboring letters. Using this idea as a foundation, we produced a list of neighbors of all Arabic letters. Table 6 illustrates an excerpt of the list of neighboring letters.

Table 6. An excerpt of the list of neighboring letters

Character	Neighbors					
ء /ʾ/	ؤ /ʾū/	ئـ /ʾi/	ا /ā/	أ /āa/	إ /āi/	آ /ʾā/
ئـ /ʾi/	ء /ʾ/	ؤ /ʾū/	ا /ā/	أ /āa/	إ /āi/	آ /ʾā/
إ /āi/	لإ /lāi/	ئـ /ʾi/	ا /ā/	أ /āa/	آ /ʾā/	
أ /āa/	لأ /lāa/	ا /ā/	أ /āa/	إ /āi/	آ /ʾā/	
ا /ā/	ت /t/	ل /l/	أ /āa/	إ /āi/	آ /ʾā/	ى /á/
ؤ /ʾū/	ء /ʾ/	ر /r/	ئـ /ʾi/	أ /āa/	إ /āi/	آ /ʾā/
آ /ʾā/	لآ /lʾā/	ا /ā/	أ /āa/	إ /āi/	ؤ /ʾū/	

2.2.2. Morphological grammar

On the basis of the neighboring letter list, we have created a morphological grammar (see Figure 3) that performs the following four editing operations: the addition of missing letters, substitution of incorrect letters, removal of surplus letters, and exchange of two letters. Additionally, special identifiers have been added to differentiate between the four categories of editing operations in order to enhance readability and comprehension of the results. So, we have decided to add SUB for the substitution operation, DEL for deletion, INS for insertion, and TRS for transposition.

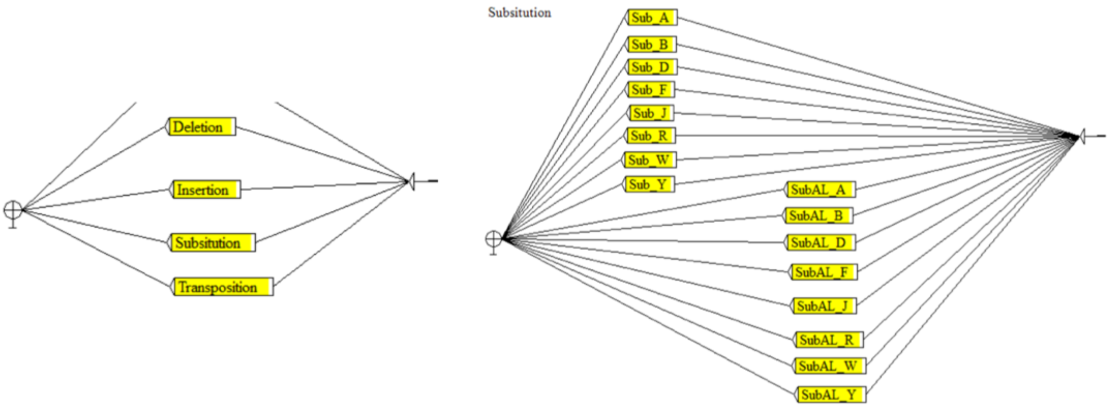

Figure 3. Morphological grammar

2.2.3. Local grammar

We next used the propositions that our morphological grammar produced to construct a local grammar illustrated in Figure 4. Truly, by using the appropriate tags, this local grammar is able to evaluate and validate each proposition. Following that, a final list of appropriate candidates that align with the valid suggestions will be produced. Each possible candidate on this list will additionally have a label indicating the editing operation that was performed. As a result, the suggested candidate will be presented as follows: #candidate#/#misspelled word#-label-#.

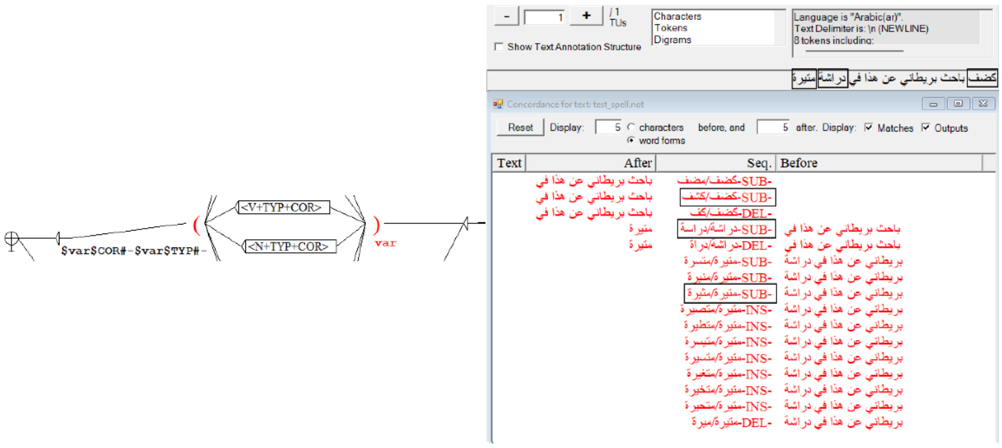

Figure 4. The local grammar and results in the NooJ concordance tool

2.3. Ranking candidates

The classification phase of the candidate list follows the creation of all possible corrections for a misspelled word. The goal is to arrange this list of possibilities in decreasing order of the most likely corrected word. The Levenshtein distance [26], often known as the edit distance, is one of the most well-known metrical procedures in the world of spell-checking [27]. We have chosen to use this method to rank candidates. The Levenshtein distance algorithm calculates the minimum number of basic editing operations required to transform a misspelled word into a correctly spelled word. It especially focuses on three specific sorts of spelling errors, namely insertion, deletion, and substitution, out of the four types that were previously identified. Subsequently, additional versions of edit distance have been developed, customized to certain authorized operations and application areas. The longest common subsequence (LCS) distance, which has applications in computational linguistics, bioinformatics, and revision control systems, allows insertion and deletion but not substitution [28]. The Hamming distance only applies to strings of the same length and only allows substitution [29]. It is used in coding theory. The Jaro distance allows only transposition and has uses in statistics and computer science [30]. The Damerau-Levenshtein distance allows the four editing operations: insertion, deletion, substitution, and transposition. Finally, we decided to use that latter distance.

Consider the following two strings: $X = x1x2...xm$, of length m, and $Y = y1y2...yn$, of length n. Recursively computing the distance between various X and Y substrings is the method for determining the edit distance between two strings. When the length of the candidates and the misspelled word are equal, the conventional approach yields identical edit distances, indicating an insufficient and unhelpful ranking. This necessitates the conversion of the initial version into a weighted version [31] by assigning a distinct weight to each editing operation, as in (1). When the enhanced algorithm is applied to the example, " كضف باحث بريطاني عن هذا في دراشة متيرة/A British researcher revealed this in an exciting study," the candidates are ranked in descending order, assigning a high score to the correct suggestion, as shown in Table 7.

$$D(i,j) = D(X_1^i, Y_1^j)$$

$$D(i,j) = min \begin{cases} D_{ins}(i-1,j) + 1 - w_{ins} \\ D_{del}(i,j-1) + 1 - w_{del} \\ D_{sub}(i-1,j-1) + cost \\ D_{trs}(i-2,j-2) + 1 - w_{trs} \end{cases} \quad (1)$$

$D_{ins}(i-1,j) + 1 - w_{ins}$ *corresponds to an insertion*
$D_{del}(i,j-1) + 1 - w_{del}$ *corresponds to a deletion*
$D_{sub}(i-1,j-1) + cost$ *corresponds to a substitution*
$D_{trs}(i-2,j-2) + 1 - w_{trs}$ *corresponds to a transpositin*
$where\ cost = min \begin{cases} 0 & if\ x_{i-1} = y_{j-1} \\ 1 - w_{sub} & otherwise \end{cases}$
$D(i,\emptyset) = i\ and\ D(\emptyset,j) = j\ where\ \emptyset\ empty\ string$
w_{ins} *is the weight of insertion operation*
w_{del} *is the weight of deletion operation*
w_{trs} *is the weight of transposition operation*
w_{sub} *is the weight of subsitution operation*

Table 7. Candidates ranking

Misspelled words	Candidates	Weight
كضف /*kaḍafa*/	كشف /*kašafa*/	95.33%
	مضف /*maḍafa*/	95.33%
	كف /*kaffa*/	77.50%
دراشة /*dirāšah*/	دراسة /*dirāsah*/	97.20%
	دراة /*dirāh*/	88.75%
متيرة /*mutirah*/	مثيرة /*muṯirah*/	97.20%
	منيرة /*munirah*/	97.20%
	متسرة /*muttasirah*/	97.20%
	متصيرة /*mutaṣirah*/	96.88%
	متطيرة /*mutaṭayyirah*/	96.88%
	متيسرة /*mutayassirah*/	96.88%
	متسيرة /*mutasayyirah*/	96.88%
	متغيرة /*mutaġayyirah*/	96.88%
	متخيرة /*mutaḫayyirah*/	96.88%
	متحيرة /*mutaḥayyirah*/	96.88%
	ميرة /*mīrah*/	88.75%

2.4. Correcting

The candidate with the highest score will be chosen to conclude the spell-checking process. Unlike an interactive spell checker that enables the user to select the correct word, an automated spell checker will automatically replace a misspelled word with the best candidate. In our case, we have made the decision to use an interactive spell checker. Taking the example cited in our study, the correction of the three misspelled words gives كشف /kašafa/ instead of كضف /*kaḍafa*/, دراسة /dirāsah/ rather than دراشة /*dirāšah*/, and مثيرة /muṯirah/ in place of متيرة /*mutirah*/.

3. RESULTS AND DISCUSSION

Once the grammars were created and assessed using the NooJ platform, we proceeded to implement them in a Web application that had been designed based on the flowchart shown in Figure 5. The Noojapply tool was used at this step, as previously noted, to leverage the extensive capability offered by NooJ.

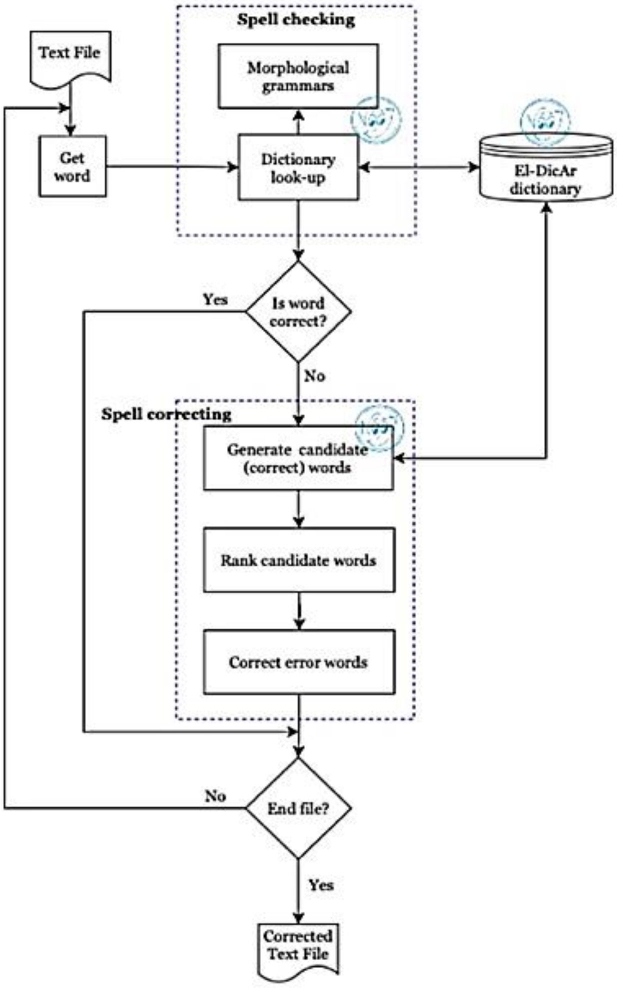

Figure 5. Flowchart for spell-checking and correcting

Consider again the previous example: “كضف باحث بريطاني عن هذا في دراشة متيرة/A British researcher revealed this in an exciting study.” Our spellchecker has highlighted in red the three misspelled words in this sentence: كضف /*kaḍafa*/, دراشة /*dirāšah*/, illustrated in Figure 6, and متيرة /*mutirah*/, illustrated in Figure 7. The user is able to choose the correct word by clicking on a misspelled word to display a selection of candidates, and so on until all errors have been corrected.

Figure 6. List of candidates for the misspelled word دراشة /*dirāšah*/

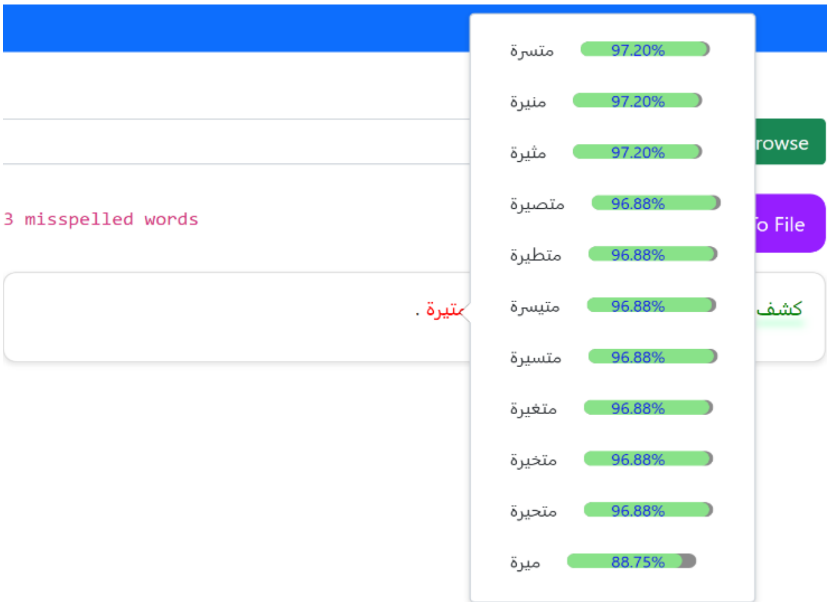

Figure 7. List of candidates for the misspelled word متيرة /*mutirah*/

During the concluding stage of our project, we will conduct an evaluation of our prototype by calculating its level of accuracy. This evaluation will enable us to gauge the effectiveness and precision of our prototype. The following section will outline the results of evaluations carried out on a selection of Arabic press articles, which were chosen based on four distinct topics: media, economy, sport, and society in Tables 8 and 9. In order to validate our prototype, it is necessary to assess its performance using the proven metrics: precision, defined as stated in (2); recall, defined as specified in (3); and F-measure, defined as described in (4). The overall evaluation reveals that our Arabic spell checker, Al Mudaqiq, outperforms "Word 2019" in terms of accuracy. In comparison to Word 2019, we acquired an average F-measure of 89.23% for our prototype.

$$Precision = \frac{Number\ of\ errors\ correctly\ detected}{Number\ of\ detections} \tag{2}$$

$$Recall = \frac{Number\ of\ errors\ correctly\ detected}{Number\ of\ introduced\ errors} \tag{3}$$

$$F - measure = 2 * \frac{Precision * Recall}{(Precision + Recall)} \tag{4}$$

Table 8. Experience with our prototype (Al Mudaqiq)

Variables	Text n1	Text n2	Text n3	Text n4
Number of words	165	310	262	238
Number of errors correctly detected	10	18	15	14
Number of detections	13	24	17	16
Number of introduced errors	10	18	16	14
Precision	76.92%	75%	88.24%	87.50%
Recall	100%	100%	93.75%	100%
F-measure	86.96%	85.71%	90.91%	93.33%

Table 9. Experience with word 2019

Variables	Text n1	Text n2	Text n3	Text n4
Number of words	165	310	262	238
Number of errors correctly detected	10	16	15	12
Number of detections	14	26	18	15
Number of introduced errors	10	18	16	14
Precision	71.43%	61.54%	83.33%	80%
Recall	100%	88.89%	93.75%	85.71%
F-measure	83.33%	72.73%	88.24%	82.76%

4. CONCLUSION

In this paper, we have demonstrated how we used NooJ (and noojapply) to develop an Arabic spell checker. After giving an overview of NooJ and its command-line utility, noojapply, we started by defining and describing various types of spelling errors. We have contributed by presenting the four major steps of our spell checker, which include the development of a NooJ local grammar for error detection. In order to generate all possible candidates for corrections, we then constructed a morphological and local grammar in NooJ. Then, in order to classify these candidates, we created an improved algorithm. Then, to illustrate our work, we developed a web interface. Finally, we have created a comparison table of tests done on different topics in Arabic press articles between our prototype and word 2019.

Regarding perspectives, our goal is to improve grammar error detection to take real words into account as well as context-dependent and context-sensitive errors, which refer to errors involving textual and linguistic context as well as missed spaces between words. We also aim to improve our ranking algorithm to be more precise, to avoid having candidates with the same weight, and finally, to improve and enrich the presentation layer of the results.

REFERENCES

[1] K. Beesley, "Arabic finite-state morphological analysis and generation," Jan. 2003, doi: 10.3115/992628.992647.

[2] L. Tuerlinckx, "Lemmatization of non-classical Arabic (in French: *La lemmatisation de l'arabe non classique*)," in Le poids des mots. Actes des 7èmes JADT, vol. 2, Louvain-la-Neuve: Presses Universitaires Louvain, 2004, pp. 1069–1078. [Online]. Available: https://api.semanticscholar.org/CorpusID:159942632.

[3] S. Khoja, R. Garside, and G. Knowles, "An Arabic tagset for the morphosyntactic tagging of Arabic," Lancaster University, 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:62244178.

[4] A. A. Abdelwahed, "Verb structure reading in Arabic conjugation (in Arabic: *Binyat alfiʿl qiraʾa fi altasrif alʾarabi)*," Sfax University, 1996.

[5] F. S. Douzidia, "Automatic Arabic text summarization (in French: *Résumé automatique de texte arabe*)," 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:61729731.

[6] M. Boudchiche, A. Mazroui, M. Bebah, A. Lakhouaja, and A. Boudlal, "AlKhalil Morpho Sys 2: A robust Arabic morpho-syntactic analyzer," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, pp. 141–146, Apr. 2017, doi: 10.1016/j.jksuci.2016.05.002.

[7] I. Youssef, "Place assimilation in Arabic: contrasts, features, and constraints," 2013.

[8] Ibn Jinnī, "Secret of Syntax creation" (in Arabic: *"Sirr ṣināʿat al-ʾiʿrāb"*), 2nd ed., vol. 2. Beirut: Dār al-Kutub al-ʿIlmiyya, 2007.

[9] A. I. Alfozan, "Assimilation in classical Arabic: a phonological study," 1989. [Online]. Available: https://api.semanticscholar.org/CorpusID:142485870.

[10] A. R. Altakhaineh and A. Zibin, "Phonologically conditioned morphological process in modern standard Arabic: an analysis of Al-ibdal 'substitution' in Ftaʕal pattern using prosodic morphology," *International Journal of English Language and Linguistics Research*, vol. 2, pp. 1–16, Mar. 2014.

[11] M. Al-Galaayiini, "A set of Arabic lessons part one (in Arabic: *ǧāmiʿ addurūs alʿarabiyah alǧuzʾ alʾawwal*)," vol. 1. 1991.

[12] A. Ar-rajihi, "Morphological application (in Arabic: *Attaṭbīq assarfī*)," Beirut: Dar Annahda Alʿarabiyah, 1984.

[13] N. Al-Alwani, "Morphological replacement in (Ifta'ale) formula in Majma`a Al-Bayan," *Journal of Arabic language and its Arts*, University of Kufa, pp. 93–120, 2010, [Online]. Available: https://api.semanticscholar.org/CorpusID:180915404.

[14] G. Olani and D. Midekso, "Design and implementation of morphology based spell checker," *International Journal of Scientific & Technology Research*, vol. 3, pp. 1–8, Dec. 2014.

[15] M. Silberztein, "NooJ manual," 2003. Accessed: Oct. 19, 2023. [Online]. Available: http://www.NooJ4nlp.org/NooJ/Manual.pdf

[16] B. Haddad and M. Yaseen, "Detection and correction of non-words in Arabic: a hybrid approach," *International Journal of Computer Processing of Languages*, vol. 20, no. 04, pp. 237–257, Dec. 2007, doi: 10.1142/S0219427907001706.

[17] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Commun. ACM*, vol. 7, pp. 171–176, 1964, [Online]. Available: https://api.semanticscholar.org/CorpusID:7713345.

[18] K. Kukich, "Techniques for automatically correcting words in text," *ACM Comput. Surv.*, vol. 24, pp. 377–439, 1992, [Online]. Available: https://api.semanticscholar.org/CorpusID:5431215.

[19] M. Silberztein, "Formalizing natural languages: the NooJ approach," 2016. doi: 10.1002/9781119264125.

[20] M. Silberztein and A. Tutin, "NooJ, a NLP tool for language teaching. Application for the study of lexical morphology in FLE (in French: *NooJ, un outil TAL pour l'enseignement des langues. Application pour l'étude de la morphologie lexicale en FLE)*," *ALSIC*, vol. 8, Feb. 2005, doi: 10.4000/alsic.336.

[21] S. Mesfar, "Analyse morpho-syntaxique automatique et reconnaissance des entités nommées en arabe standard," 2008. [Online]. Available: https://api.semanticscholar.org/CorpusID:160716402.

[22] M. Silberztein, "The official NooJ website. 2003. Accessed: Oct. 21, 2023. [Online]. Available: http://www.NooJ4nlp.org/

[23] R. Kassmi, M. Mourchid, A. Mouloudi, and S. Mbarki, "Processing agglutination with a morpho-syntactic graph in NooJ," vol. 811. 2018. doi: 10.1007/978-3-319-73420-0_4.

[24] A. Yousfi and H. Gueddah, "The impact of Arabic inter-character proximity and similarity on spell-checking," 2013. doi: 10.1109/SPIRE.2000.878178.

[25] R. Kassmi, M. Mourchid, A. Mouloudi, and S. Mbarki, "Recognition of Arabic phonological changes by local grammars in NooJ," vol. 117, no. 1, pp. 3-14, CCIS. 2020. doi: 10.1007/978-3-030-38833-1_1.

[26] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet physics*. Doklady, vol. 10, pp. 707–710, 1965, [Online]. Available: https://api.semanticscholar.org/CorpusID:60827152.

[27] K. Bacha and M. Zrigui, "Contribution to the achievement of a spellchecker for Arabic," *Research in Computing Science (RCS)*, vol. 117, no. 1, pp. 161–172, Dec. 2016, doi: 10.13053/rcs-117-1-14.

[28] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE*, vol 84, no. 406, pp. 39-48, 1989, doi: 10.1109/SPIRE.2000.878178.

[29] A. Bookstein, V. Kulyukin, and T. Raita, "Generalized hamming distance," *Information Retrieval Journal Boston*, vol. 5, Oct. 2002, doi: 10.1023/A:1020499411651.

[30] M. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 Census of Tampa, Florida," *Journal of The American Statistical Association*, vol. 84, pp. 414–420, Jun. 1989, doi: 10.1080/01621459.1989.10478785.

[31] H. Gueddah, A. Yousfi, and M. Belkasmi, "Introduction of the weight edition errors in the Levenshtein distance," *International Journal of Advanced Research in Artificial Intelligence*, vol. 1, Aug. 2012, doi: 10.14569/IJARAI.2012.010506.

BIOGRAPHIES OF AUTHORS

Rafik Kassmi received his B.S. degree in applied mathematics from Hassan II University of Casablanca, Morocco, in 1998, and his degree in computer engineering from Polytech Tours, France, in 2002. He is now nearing the completion of his Ph.D. in the area of computational linguistics, specifically focusing on Arabic Natural Language Processing. His research interests include phonetics, phonological changes in Arabic, and artificial intelligence. He can be contacted at email: rafik.kassmi@gmail.com.

Samir Mbarki received his B.S. degree in applied mathematics from Mohammed V University, Rabat, Morocco, in 1992, and his doctorate of high graduate studies in computer sciences from Mohammed V University, Rabat, Morocco, in 1997. In 1995, he joined the faculty of science at Ibn Tofail University, Morocco, where he is currently a professor in the Department of Computer Science. He obtained an HDR in computer science from Ibn Tofail University in 2010. He does research in model driven engineering, software engineering, artificial intelligence, and NLP. Their current project is resource allocation in wireless networks. His research interest score is 557.1, the number of citations is 630, and his h-index is 12. His research interests include programming languages, software engineering, artificial intelligence, MDA, and NLP. He can be contacted at email: mbarkisamir@hotmail.com.

Abdelaziz Mouloudi received his doctorate of high graduate studies in computer sciences from Mohammed V University, Morocco, in 1988, and he obtained an HDR in computer science from Mohammed V University, Morocco, in 2008. In 1988, he joined the faculty of sciences at Ibn Tofail University, Morocco, where he is currently a professor in the department of computer science. He does research in artificial neural networks, artificial intelligence, computer security, and reliability. His research interest score is 329.2, the number of citations is 502, and his h-index is 13. His research interests include evolutionary algorithms encryption cryptography, information security, and data security. He can be contacted at email: mouloudi_aziz@hotmail.com.