

A New Spell-Checking Approach Based on the User Profile

Ahmed AbdAlrhman Saty¹, Si Lhoussain Aouragh² and Karim Bouzoubaa³

¹College of CS and IT, Sudan University of Science and Technology, Sudan.

²High National School for Computer Science and Systems Analysis, Mohammed V University in Rabat, Morocco.

³Mohammadia School of Engineers, Mohammed V University in Rabat, Morocco.

Received 2 Aug. 2022, Revised 2 May. 2023, Accepted 23 May. 2023, Published 30 May. 2023

Abstract: This paper presents a new approach for spell-checking based on the user profile and that can be applied for any language. For this purpose and for the specific case of Arabic, spelling errors are studied and divided into 18 types. Then, a relationship model between users and their errors is obtained. The proposed architecture initially gives apposite values for a current user, then corrects misspelled words by applying the spelling rules, and the remaining words are corrected based on the probability given by an adopted model of the profile values. To show the efficiency of our profile-based approach, we conducted an experiment with a corpus of 11,908 words containing 1,888 errors. It showed that our approach suggests the correct word in 88.43% times and ranks it in the first four positions in 75.14% times. Moreover, using the same corpus we compared our implemented tool with two existing ones where ours ranked better in 69.79% times than Sahehly and 77.63% times than MS word.

Keywords: Arabic linguistic, error-type, intelligent profile detection, multinomial logistic regression, spell-checking

1. INTRODUCTION

Spell-checking is based mainly on checking unacceptable written words in the used language and suggesting a list of correct words that are related or similar to the given incorrect word. The error [1] may be an isolated error, meaning that the written word does not belong to the language in use, or a real-word error, meaning that the written word is correctly spelled but is not used in the correct position [2], [3]. The importance of spell-checking is increasing with the proliferation of computers and smart devices in most aspects of daily life. Spell-checking approaches and algorithms [4], [5], [6], [7] can be adapted to any language, taking into account that a vocabulary and a rule-based module should correspond to the characteristics of the chosen language. In this paper, we focus on the isolated errors in the Arabic language, noting that we use the Buckwalter transliteration [8] convention.

Arabic has a rich morphology and a templatic derivation that makes it possible to form many lemmas and words from a given root. Also, it has affixes (prefixes and suffixes) that can be added to a word [9], [10]. Some letters are spelled differently depending on their position in words. For instance, the hamza (“ء”) is written in five different forms depending on its position such as “سؤال” (“sWAl”), “سأل” (“sOl”), and “أسئلة” (“Os|lp”). Moreover, many Arabic letters are similar in their shapes, such as “ع” (“E”)

and “غ” (“g”), or in pronunciation, such as “ض” (“D”) and “د” (“d”). Besides, the process of abbreviation in some words, such as the words “من ما” (“mn mA”) is abbreviated to “مما” (“mmA”). All these factors and various levels of users make it important to study spell-checking based on a deep understanding of the specificity of the composition of Arabic.

As mentioned earlier, Arabic digital content is constantly increasing in the Arab and Islamic digital world, and the widespread use of electronic services and social multimedia services such as Facebook and WhatsApp have also contributed to the increase in the number of users. Consequently, with the rapid growth of Arabic digital content and users, as well as the specificity of many linguistic phenomena as explained above, user spelling errors are increasing.

Accordingly, it is appropriate to find a new approach for the Arabic Spelling checkers which satisfies and handles all user types, starting with detecting the user profile from the categories, and then choosing the suitable error corrections of the current user.

This paper presents a new technique that improves Arabic spell-checking by developing an approach based on the user’s profile. The rest of the paper is organized as follows. Section 2 reviews the related works. Section

3 explains the proposed architecture. Section 4 discusses the rules, as well as which ones are handled and those that are not taken into consideration. Section 5 shows our collected vocabulary used in this study. Section 6 describes each aspect of the user profile and its relationship to the error types. Section 7 presents the statistics and our results that confirm the importance of this study. Section 8 displays the prototype system implementing the architecture. Finally, we conclude the paper and make a proposition for future works.

2. RELATED WORK:

Although it still requires many improvements, there has not been much research and work on the Arabic spell-checking in the last five years (2022 to 2018) compared to the previous years. Moreover, many of the existing Arabic spell checker works are not available. The recent works can be summarized in several aspects. The aspect of techniques used; many studies generally focused on spell checking using different techniques, such as the rule-based approaches [11], [12], the distance similarity techniques, the exploitation of morphological analysis [13], [14], techniques based on phonetics, and finally hybrid ones [15]–[16]. Despite these efforts, no particular existing system has advantages and overcomes the entire spelling problem.

The second aspect is the vocabulary used. Some authors made efforts to solve an insufficient Arabic vocabulary, such as the author of [13], who developed a prototype checker that provides limited access to all dictionary data. Surface patterns and morphological processing are used to identify the sub-dictionaries with the highest probability and only the suggested words closest to the error word are considered. Also, other authors collect the vocabulary from many resources such as [2] with a vocabulary of 41,170,678 words from Al-Riyadh newspaper articles, and [17] who created QALB corpus (Qatar Arabic Language Bank), a large manual from native and non-native articles and machine translation output. However, due to the peculiarities of the language and the spread of dialects, there is still no composition of the vocabulary that contains or generates all Arabic words.

The third aspect is the user particularity. Some existing works deal with errors arising from special user classification. Authors of [18] captured errors for Egyptian dialects, the author of [2], [19] propose particular checkers for the Iraqi dialect, the authors of [12], [20] design checkers specially helping the non-native learners, and the author of [21] addresses text written by dyslexic writers using a specific dyslexic corpus. Although they work well, each checker is intended for a specific type of users and therefore may not produce the desired results when used by a different type of user.

The last aspect is the types of errors. Although the author of [21] addresses only dyslexic writers, it treats split words, spaces, and repeated characters. The author of [22] proposed a system that handles space deletion error,

insert error, delete error, and transpose or replace errors. It depends on a set of predefined common prefixes and rules. Also, the work of [23] deals with space, insert and delete errors by applying A* lattice search and n-gram. As well, the author of [24] proposes an adapted Levenshtein distance for correcting space and deletion errors. The authors of [25] address hamza, taa marbuta “ة”, yaa “ي” errors and confusion between dah “د” and zah “ذ” by applying regular expressions and a word substitution list. Furthermore, the checker created by the author of [15] consists of a hybrid pipeline that combines five approaches to handle different types of errors (space errors, hamza and yaa errors, transposition errors). However, it is better to perform the correction process based on the user type and then select the appropriate vocabulary and error type.

In summary, we note that despite the efforts that have been made in the field of spell checking, they are scattered and have limitations in some aspects. Table I shows the recent Arabic spellchecking works.

3. THE PROPOSED ARCHITECTURE:

The proposed architecture is shown in Figure 1. Briefly, the user starts typing a text; then, the system initially corrects some mistakes based on the common errors module. To do this, the first module checks whether the typed word respects one of the rules stored in the Common error rules file. The system automatically corrects these errors without performing any similarity calculation. The next module concerns the detection of the remaining errors by comparing the typed words with the vocabulary. Then, in Module 3, the system reads a user profile and ranks the suggested words based on the selected user profile.

Let us recall that this architecture is implementing a new approach to overcome the existing spellcheckers by defining a relationship between the types of errors and the user profile. Consequently, the details of the four modules of our approach are:

- Module 1. The common errors module: a common error is defined as a spelling or grammatical error made by a group of users due to their non-awareness of certain language rules [26], [27] or ignoring some errors that most people agree on. For instance, the hamza error in the first position of the word is considered a common error since some users tend to type “أكل” (“AkI”) instead of “أكل” (“OkI”). Also, changing the pronunciation of some letters [10] due to local dialects are considered common errors, such as “ألب” (“olb”) instead of “قلب” (“qlb”). It is noted that dealing with the common errors separately has many advantages, such as better correction results and higher speed of the checker since the similarity distance is not processed.

TABLE I. SUMMARY OF THE RECENT WORKS

Work	approach	vocabulary	user classification	types of errors
[11]	-Rule-based approach (tri-gram - finite-state automaton - noisy channel model)	-Morphological analysis (9 million words)	-	Substituting letters
[12]	-Rule-based approach	-Bulk-walter's Arabic -morphological analyzer	Non-native learners	Editing errors, phonetic errors, vowel errors, tanween errors, shadda errors and semantic errors
[13]	-Adapted Levenshtein algorithm	-Surface patterns and morphological processing (sub-dictionaries management)		
[14]	-Adopted Levenshtein algorithm by Surface patterns	-Dictionaries		
[6]	-adopted Levenshtein algorithm by Surface patterns	-Dictionaries		
[15]	-Hybrid approaches:(rule-based linguistic techniques statistical methods using language machine translation error-tolerant finite-state automata method.)	-QALB corpus Wordlist (9 million) -Monolingual Arabic corpus		Split, add before, delete, edit, merge, add after, move, common mistakes (hamza and yaa), and punctuation
[16]	-Hybrid approaches: rule-Based Corrector Probabilistic-Based Damerau-Levenshtein Punctuation Recovery	-QALB corpus -KSU corpus -Arabic Corpora (OSAC) -Al-Sulaiti Corpus -KACST Arabic Corpus -madamira corrector -Ghaltawi		edit, add, split, merge, punctuation, phonological, common mistakes (Alif, yaa), and repeated letters
[17]	-tested and evaluated 9 Arabic correction systems	-QALB corpus		
[18]	-Rule-based approach	-morphological analysis	Egyptian dialect	reading errors (similar characters) -hearing errors -touch-typing errors -morphological errors -editing errors-
[20]	-Ngram algorithm	-dictionary lookup	Iraqi dialect	
[12]	-Levenshtein distance	-dictionary lookup	non-native learners	phonemic contrasts visually similar inflected error heard Arabic keyboards typographical errors
[21]	-Tri-gram	-dyslexic corpus	dyslexic writers	split words, spaces, and repeated characters dyslexic error
[22]	-Confusion matrix -Noisy Channel model	-predefined common prefixes and rules		space deletion error, insert error, delete error, and transpose or replace errors
[23]	-Damerau-Levenshtein -A* lattice search -N-gram probability	-		space, insert and delete errors
[24]	-adapted Levenshtein distance	-corpus (Al-watane newspaper)		space and deletion errors
[25]	-Rule-based approach	-word list		hamza, taa marbuta "ة", yaa "ي" errors and confusion between dah "ذ" and zah "ز"

- Module 2. The detection module scans the remaining words, compares them with the vocabulary, and then generates a list of incorrect words.
- Module 3. The user profile module determines the user profile according to his(er) typed errors by applying statistical rules from the profile-based mapping detailed below. This process keeps repeating until the current user profile is determined. Once done, its value is saved in order to be used for manipulating the remaining errors.
- Module 4. The ranking module filters the erroneous words based on the vocabulary and the measured distances. Then, the suggested words are ranked based on the user profile. Consequently, the suitable proposed words are used to correct the erroneous words. In addition, this module has a cache file to keep the previously erroneous words alongside their corrections directly for the same next errors without having to recalculate the distances.

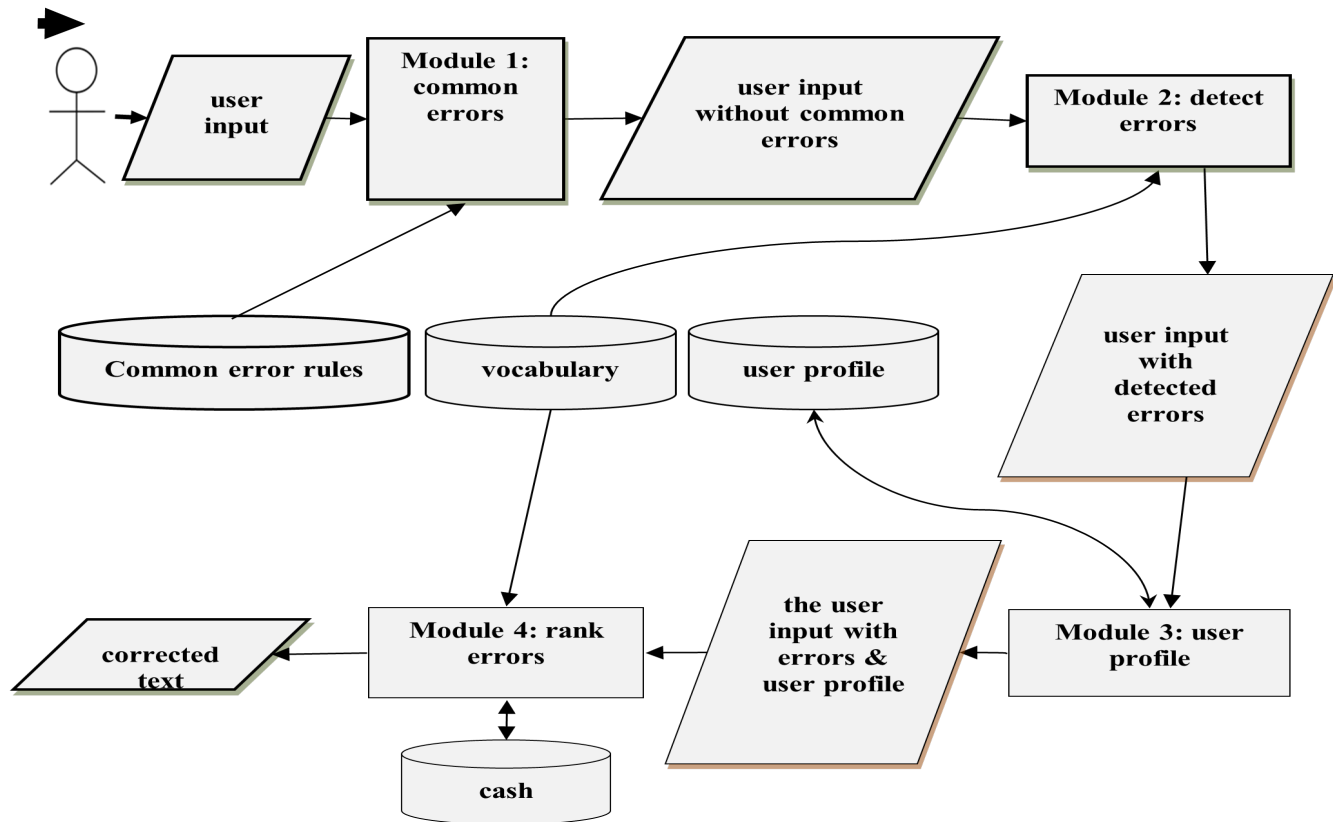


Figure 1. The proposed architecture.

```

load vocabulary;
input text;
common_errors (text);
error_words= detect_error(text);
calculate profile_values(userProfile);
foreach word:error_words
  foreach cashword: cash_file
    if word.equal(cashword.error_word)
      return cashword
    else
      cashword= profile_based_correct(word,profile_values);
      cash_file.add(cashword);
      return cashword;
    end if
  end foreach
end foreach
end foreach
  
```

Figure 2. Algorithm of the proposed approach.

The Figure 2 briefly shows an algorithm that can be used to implement the proposed architecture.

4. IMPLEMENTED RULES:

Numerous studies have been conducted to manipulate the common errors. Some of them used the predefined confusion sets [3], [28] by defining a list of the common errors and their correction, such as the set of “ألْب” (“olb”)

and “قلب” (“qlb”). However, the manipulation by this approach does not provide the desired results because it has two main drawbacks. First, it fails in handling errors related to some letters that have their own specificity in writing. For instance, the hamza letter has many forms depending on its position in a word and its diacritical mark, as well as the diacritical mark of the previous and the next letter. Second, the approach necessitates a huge corpus to cover whole expected. On the other hand, the rule-based approach has been used by some authors. The authors of [25] designed rules using regular expressions and word substitution lists to correct common errors, such as dealing with hamza errors, the confusion between the “ض” (“D”) and “ط” (“Z”), and the omission dots with taa-marbuta “ة” and yaa “ي”. Moreover, the works of [16], [23], [29], [30] also captured various kinds of common errors. The rule-based approach is better than the previous one but requires more effort to cover them all. The main reason for this shortfall is the complexities of the Arabic language, such as derivation and ignoring diacritics in texts.

The most comprehensive list of common error rules can be found in [31]. It consists of 104 rules divided into 10 categories. For example, the category “taa-mabsouta” and “taa-marbuta” contains six rules. Of the 104 rules,

we were able to implement 24 of them. Analysis of the remaining rules shows that additional work is required, such as diacritization of the corresponding word or invoking a morphological tool. For instance, when a hamza is preceded by a broken letter, it should be written on “alif magsora” such as “بارئ” (“bAr”), and this cannot be processed without diacritizing the text.

Since diacritizing and morphological analysis extra works are outside the scope of the present work, we decided to limit ourselves to the 24 rules. These were created in an XML file, and each set of rules was then classified with a category tag, as is the case [31].

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Common_Error_Rulesxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Rule_category id="1" value="الهجرة المتوسطة">
    ...
  </Rule_category>
  ...
  <Rule_category id="5" value="حذف الألف من كلمة (اسم)">
    <Rule id="5.1"
      description="تحذف في البسمة الكاملة (بسم الله الرحمن الرحيم)
        وتبقى في غير الكاملة (باسم الله)
        words="باسم"
        next_word="الله الرحمن الرحيم"
        convert_to="بسم"
        error_example="باسم الله الرحمن الرحيم"
        example_correction="بسم الله الرحمن الرحيم"/>
    ...
  </Rule_category>
  ...
</Common_Error_Rules>
```

Figure 3. Xml file of the common error rules.

As an example, Figure 3 shows the fifth category named deletion of the alif letter from the word “اسم” (“Asm”). This category has rule 1 manipulates alif in “باسم” (“bAsm”). If this word comes before “الله الرحمن الرحيم” (“Allh AlrHmn AlrHym”), the alif is omitted “بسم” (“bsm”). For sharing with the research community, this XML file can be freely downloaded [32].

5. THE VOCABULARY:

The vocabulary is one of the most important components in the proposed architecture as it is used to detect errors and rank candidate words. Therefore, it must be as large as possible, containing all possible vocabulary of the Arabic language in order to provide proper responses.

The vocabulary was collected from many corpora to ensure an extensive coverage of correct Arabic words. The corpora were Khaleej corpus [33], Nemlar corpus [34], Tashkeela corpus [35], Watan corpus [36], Arabic Learner

corpus [37], Abu El-Khair Corpus[38], and some files collected from Aljazeera and Al-Arabiya newspapers. The whole collection contained 58,298 text files with 79,687,338 words equaled 2,281,655 unique words. Then, a filtering process was performed to eliminate incorrect words by getting the stem of each word of the collected vocabulary and verifying its existence in the Calem lexicon [39]. This latter is one of the SAFAR [39] framework resources and is considered as one of the largest stem vocabularies [40] in the ANLP community, reaching more than 7 million stems. The process allowed us to detect a list of 77,791 incorrect words where part of them were corrected and the others were deleted. For instance, the word “الشديق” (“Al\$dyq”) was corrected to “الصديق” (“AlSdyq”) while the incorrect word “سو” (“sww”) was deleted.

6. THE USER PROFILE:

By a user profile, we mean knowing the characteristics of a group of people who have a common thread in making a particular mistake. For example, a group of people with a low level of education has the tendency to make similar mistakes. For instance, those who have attended less than elementary school make most mistakes in lamshamsia “ال” to write “الشمس” (“A\$m”) instead of “الشمس” (“Al\$m”). Consequently, having more accurate information about users’ features help the spellchecker be more accurate and adapt to the correct user profile.

The profile-based module is the most important component of the proposed architecture and requires knowing the nature of the user while typing and then selecting the right spell-checking parameters that best suit him(er). To achieve this, we had to perform an analytical study of the users and the mistakes they often make and find a relationship between them. Since there is no benchmark corpus that contains user profiles and matches with their errors, we compiled a specific corpus that was given to a sample of users, then provided observations and views about the relationship between users and their errors.

A. The corpus:

To compile the aforementioned corpus, a group of people edited different files (at least one page). Next, we conducted a survey to gather information about all the individuals participating in the study. A comprehensive representation of the profile of these individuals were considered: gender (male, female), age (old-aged, middle-aged, young-aged), computer usage (high, low), etc.

After receiving all the documents, we corrected the spelling errors separately in each document. Then, a file was created containing the following sheets: people -documents -types of possible errors -errors they made.

In the end, we converted this work into a corpus file using TEI format [41] which is a standard form using the XML language to represent texts in digital form for

online research, teaching, and preservation, where it can be used and shared among those interested. The corpus file contains the following sections (tags): people, documents they printed, types of possible errors, and mistakes they made. Each section (tag) contains some data explaining its details and content. The “people” section contains basic information about each person and their relationship to computer use, while the “documents” section contains all the sentences in each document, with each sentence numbered for use in the errors section. We also add the “type of errors” section in which we list all possible errors with their description in Arabic and provide an illustrative.

```

<TEI>
  <teiHeader>
    . . .
  </teiHeader>
  <persons xml:id="persInfo">
    . . .
  </persons>
  <documents xml:id="docInfo">
    <document>
      . . .
    </document>
  </documents>
  <errorTypes xml:id="errorInfo">
    . . .
    <errorType xml:id="err9">
      <causeDesc>forget press on space</causeDesc>
      <causeArabicDes>حذف المسافة</causeArabicDes>
      <example>
        <errorWord>ينغمسفي</errorWord>
        <correctWord>ينغمس في</correctWord>
      </example>
    </errorType>
  </errorTypes>
  <errors xml:id="errorsInfo">
    . . .
    <error>
      <errorId>117</errorId>
      <personId>per17</personId>
      <docId>doc17</docId>
      <statementId>22</statementId>
      <errorWord>تسمعاو</errorWord>
      <correctWord>وتسمعا</correctWord>
      <causes>
        <errorTypeId>err9</errorTypeId>
      </causes>
    </error>
    . . .
  </errors>
</TEI>

```

Figure 4. Example of the error type element

Figure 4 shows a sample of the errors-types named “forget press on space” and its Arabic name “حذف المسافة” as well as an illustrative example of the error word “ينغمسفي”

(“yngmsfy”) and its corrected word “ينغمس في” (“yngmsfy”).

B. The mapping between errors and user profile

1) Classification of errors:

Based on the compiled corpus, we find that a group of users with similar attributes make similar errors. Therefore, we strive to discover the relationship between their profiles and the type of errors they make. To achieve this, it is necessary to study these categories of errors.

There are many existing classifications of errors. The simplest is to classify an error as either isolated or context-sensitive. Some researchers [15], [21], [42] mentioned that most typos are as follows: inserting letters, deleting letters, substituting letters, and interchanging two adjacent letters. Others [43] studied the spelling errors to classify them into three categories based on user knowledge: typographical errors, cognitive errors, and phonetic errors. According to the analysis of our corpus, it is necessary to extend the existing classifications and consider all what the users have misspelled.

Indeed, some of them have been taken as they are, such as the “adjacent letters” error. Other types have been divided into newly created ones. For example, the “letter substitution” type is divided into four types; exchange letter, replace error, shift keyboard error, and shape error. At last, we have considered 18 different types. Table II lists all of them where the second column describes each type, the third column gives an example, and the last one provides the correct word.

2) The relationship between profile and errors:

The compiled corpus contains 31 user profiles. Each attribute representing the profile has been used to classify the error by matching them with the users’ errors. Then, we conducted a set of analyzes to find a relationship between the error type and the user profile.

After identifying the error types and collecting the user profile data alongside their errors, we divided the mapping into two parts. The first one is the statistical test, which we use only once to measure the mapping relationship. The second is the calling of the functions that are executed for each new user profile to obtain his(er) probability mapping values.

Firstly, we implemented a function named error-determination, which determines the error types by comparing the misspelled word with the suggested words. For example, if the misspelled word is “هرةب” (“hrpb”), the function then returns “insertion error” for “هرة” (“hrp”) and “adjacent letter” for “هروب” (“hrwb”).

Second, to find the relationship between the profile and the errors, these data were statistically analyzed with each other. The chi-square test was first applied to evaluate

TABLE II. TYPES OF ERRORS.

No	Error type	Description	Error example	Correction
1	add space	A user adds space	و التخطيط	والتخطيط
2	adjacent letter	Pressing the letter whose location on the keyboard is close to the required letter	والإبتزاز	والإبتزاز
3	dialect error	Substitution of letters due to local dialects	زكري	زكري
4	deletion error	A letter is omitted from the word	للملم	للمعلم
5	double letter	Repeating one letter of the word more than once	يتسبب	يتسبب
6	double word	Repeating a word more than once	انخفاضانخفاض	انخفاض
7	exchange latter	Preceding a letter over another letter	مطلبات	مطلبات
8	forget press on space	Deleting a space between two words	تجري بشكل	تجري بشكل
9	hamza error	Not writing the correct hamza in a word	الاربع	الأربع
10	insertion error	Adding a supernumerary letter	الجماعي	الجماعي
11	lam-shamsiya	The lam-shamsiya is written but not pronounced	اتعرض	التعرض
12	press adjacent letters at the same time	Pressing at the same time on two adjacent letters	الجهاز	الجهاز
13	replace error	Writing a letter that is not the desired letter	قهوخ	قهوة
14	shift keyboard	On the keyboard, there are letters at the top of the buttons; when you type them, you should press the Shift button with the desired letter.	عجاء	إجراء
15	speed error	With speed in writing, a mistake may occur	فبينما	فبينما
16	taa-marbutaerror	There is confusion between taa-marbuta and the haa (without double dot) at the end of the words	الذكية	الذكية
17	shape error	Some letters are similar in writing. Therefore, a user may make a mistake in choosing the required letter	منطقة	منطقة
18	alif magsura and Yaa confusion	There is confusion in writing alif and yaa at the end of the words	التي	التي

the correlation of the profile variables with the types of errors. This test allowed us to exclude two independent variables whose p-value was below the alpha-level [44] table, namely the sex and document variables. The other variables were kept because they were correlated with the errors. Moreover, Multinomial Logistic Regression [45] is a classification model that can be used when the dependent variable is nominal. The model was applied to obtain error probabilities, as detailed in the next section.

Thirdly, we created a function named error-type probability that uses these results, receives the profile of the current user, and then computes the appropriate probability values for each error type. This function is called whenever we process a new profile.

Finally, the Levenshtein distance is used to candidate the suggested words for each error. Also, the error-type determination and error-type probability functions are called to obtain a probability value for each error. Then, a new measure is calculated by passing these values for the current profile by applying equation 1. Subsequently, the suggested words are reordered based on these measures.

$$Profile_measure = dis * freq/error_value \quad (1)$$

Where:

dis= represents Levenshtein distance
error_value = represents values of error type for the current profile
freq = represents frequencies of a word in the vocabulary

To illustrate the mapping, let us consider the following example. Table III contains a user profile who is mistaken in “مطمئنة” (“mTmn}p”) instead of “مطمئنة” (“mTm}np”). The system starts by calculating the profile value for each error type, as shown in Table IV. Then, the above-mentioned operations are executed to order the suggested words based on the given profile. Table V compares the suggested words ordering by Levenshtein distance alongside ordering by the profile.

TABLE III. THE USER PROFILE

Gender	male
Age	middle-aged
Educational level	High school
Occupation	not yet
Specialization	not yet
Print documents	no
Computer usage	low

TABLE V. COMPARING THE RANKING OF THE SUGGESTED WORDS

Based on Levenshtein			Based on the profile			
No	Suggested word	Value	No	Suggested word	Error type	Value
1	تمنية	2	1	مطمئة	exchange letter	0. 015050487
2	مطأطة	2	2	تمنية	permanent error	0. 01058635
3	مطمورة	2	3	مطمورة	adjacent letter	0. 004541111
4	مطموسة	2	4	منمنة	permanent error	0. 001085779
5	مطمومة	2	5	مطموسة	adjacent letter	0. 000454111
6	مطمئة	2	6	مطأطة	permanent error	0. 000271445
7	منمنة	2	7	مطمومة	adjacent letter	0. 000181644

TABLE IV. THE ERROR VALUES OF THE GIVEN PROFILE.

Error type	Its value	Error type	Its value
add space	0.000132089	insertion error	0.00023304
adjacent letter	4.22523E-05	lam shmsia	0.994998864
dialect error	5.86621E-05	permanent error	0.00011803
deletion error	8.42015E-05	press adjacent letters at same time	0.000822038
double letter	5.90891E-05	shift keyboard	0.000600016
double word	1.74498E-05	speed error	0.002525368
exchange letter	5.76059E-05	taa marbota error	2.62772E-05
for press on space	9.86843E-05	alef magsora and Yaa confusion	5.17917E-05
Hamza error	7.13821E-05	shape error	3.15812E-06

When comparing the wrong word “مطمئة” (“mTmn}p”) with the suggested words, the error type is determined then the probability values (Table IV) are used in equation 1, as we see on the right side in Table V (based on the profile). It is noticeable that (Table V) the desired word is in the first order when the profile is used.

3) Calculation of error probabilities:

To evaluate our approach, we used the multinomial logistic regression to model the probabilities of a nominal random variable $Y = y_1, y_2, \dots, y_K$ as a function of a number of explanatory variables $X = X^1, X^2, \dots, X^n$. In our case, Y represents the types of errors made in Arabic, and X the set of variables that define the profile of a user.

Thus, this model estimates the probability of an erroneous word belonging to a class and consequently allows for correcting based on the most probable error type. To do so, taking the K^{th} modality as a reference error type, the approach is to define the following K-1 regression equations and keep only the most probable equations:

$$\ln \left(\frac{P(Y = y_{j/X})}{P(Y = y_{K/X})} \right) = \beta_0^j + \beta_1^j x_1 + \dots + \beta_n^j x_n, \quad (2)$$

for $j \in \{1, 2, \dots, K-1\}$

With

$$\sum_{j=1}^K P(Y = y_{j/X}) = 1$$

if we use the exponential value and add the terms from 1 to K-1, we get:

$$P(Y = y_{j/X}) = \frac{e^{\beta_0^j + \beta_1^j x_1 + \dots + \beta_n^j x_n}}{1 + \sum_{j=1}^{K-1} e^{\beta_0^j + \beta_1^j x_1 + \dots + \beta_n^j x_n}} \quad (3)$$

The model is, therefore, completely defined by the estimation of $(K-1) \times (n+1)$ parameters $(\beta_i^j)_{\substack{0 \leq i \leq n, \\ 1 \leq j \leq K-1}}$. The model is estimated using the maximum likelihood method given by the following formula:

$$L(Y, \beta) = \prod_{1 \leq j \leq K-1} P(Y = y_{j/X})^{1_{\{Y=y_j\}}} \quad (4)$$

$$\text{where } 1_{\{Y=y_j\}} = \begin{cases} 1 & \text{if } Y = y_j \\ 0 & \text{if } Y \neq y_j \end{cases}$$

The following table shows the obtained results:

TABLE VI. LIKELIHOOD RATIO TESTS (AN EXCERPT FROM THE RESULTS)

Effect	Model Fitting Criteria	Likelihood Ratio Tests		
	-2 Log Likelihood of Reduced Model	Chi-Square	df	Sig.
Intercept	1039.222	81.028	17	2.52E-10
ageGroup	1052.341	94.146	17	1.07E-12
eduLevel	1020.966	62.772	17	3.64E-07
jobTitle	1017.782	59.588	17	1.23E-06
printDoc	1055.575	97.381	17	2.71E-13
year	1033.868	75.674	17	2.23E-09
compDeal	1043.082	84.888	17	5.13E-11

For the tests of the model parameters, the Wald test [46] was used to test the influence of each user profile X^i with the following hypotheses:

$$\begin{cases} H_0 : \beta_i^1 = \beta_i^2 = \dots = \beta_i^{(K-1)} = 0 \\ H_1 \exists j \in \{1, \dots, K-1\} / \beta_i^j \neq 0 \end{cases} \quad (5)$$

TABLE VII. TEST THE HYPOTHESES FOR AGE_GROUP VARIABLE

No	Error types	Estimate parameters	Std. Error	Wald	p.value
1	add space	-2.089	1.049	3.964	0.046
2	adjacent letter	-1.13	0.635	3.164	0.075
3	dialect error	0.126	0.847	0.022	0.882
4	deletion error	0.784	0.474	2.73	0.099
5	double letter	-12.622	1181.56	0	0.991
6	double word	-12.905	3958.682	0	0.997
7	exchange latter	-14.6	1114.419	0	0.99
8	forget press on space	0.807	0.39	4.274	0.039
9	hamza error	0.397	0.231	2.947	0.086
10	insertion error	1.506	0.577	6.814	0.009
11	lam shmsia	-1.799	0		0.0
12	press adjacent letters at same time	2.837	1.374	4.264	0.039
13	permanent error	1.019	0.949	1.152	0.283
14	shift keyboard	-11.404	1874.656	0	0.995
15	speed error	10.457	2095.928	0	0.996
16	taa marbota error	1.785	0.298	35.923	0.0
17	shape error	3.787	1.254	9.118	0.003

TABLE VIII. EXAMPLE OF COMPARING THE SORTING DESIRED WORDS BY LEVENSHTAIN ALONGSIDE THE PROFILE

Error word	Desired word	Frequencies	Sort by Levenshtein	Error type	Sort by profile
جرب	جذب	552	8	dialect_error	3
الاجتمع	المجتمع	6775	1	permanent_error	1
ولاكني	ولكني	880	6	press_adjacent_letters_at_same_time	1
سخرؤا	سخرؤا	23	1	double_letter	2
الفقرى	الفقرى	113	9	alef_magsora_and_Yaa_confusion	5
شدييد	شديد	2074	1	double_letter	1
صارؤا	صارؤا	587	4	press_adjacent_letters_at_same_time	1
والخاصه	والخاصة	9	2	taa_marbota_error	4

Table VII shows the results testing the age_group variable with 7 error types having a p-value less than 0.05 leading us to accept the H1 hypothesis. These tests were also applied to the other variables (X^i).

7. EXPERIMENTS AND RESULTS

To test the proposed architecture, we studied and analyzed all the users. We found a total of 1888 errors, including 149 spacing errors (adding and deleting a space). These spacing errors were excluded from the study as they are outside the scope of the current approach and can be handled with existing works such as [23], [24]. The remaining 1739 errors contain repetitions and are reduced to 1052 unique errors.

As mentioned earlier, the proposed system initially applies the rules before using the profile. Six errors are handled by applying them, and the remaining 1046 ones are then handled using the profile. The experiments are then evaluated using four approaches. In the first two experiments, the profile approach is compared to the Levenshtein distance, while in the second two experiments, our tool called "Safar Spell Checker" is compared to the Sahehly [47] and MS-Word tools to see which tool gets the better desired words. Sahehly is one of the latest websites that specializes in Arabic content and provides spelling and

grammar correction. For instance, it has been used for testing in the work of [21]. Also, MS-Word was chosen because it is considered one of the most widely used word processors.

The first one compares the position of the correct word between the value returned using the profile and the value using the Levenshtein distance, as shown in Table VIII. For instance, for the word error "جرب" ("jzb"), the correct word is "جذب" ("j*b") taken from the corpus (see Table VIII). In this case, the profile returns the correct word at position 3, while the Levenshtein makes it at position 8.

As shown in Table IX, the first evaluation shows that in 866 cases, the profile position is better than the Levenshtein one; in 59 times, they are equal, and in 121 cases, the profile ranks lower than Levenshtein. In total, out of 1046 errors, the profile position ranks better or equal in 925 times representing a percentage of 88.43% times.

TABLE IX. SUMMARY OF STATISTICS EXPERIMENTS

No	The Comparing Results	Times	%
1	the profile position ranks better or equal	925	88.43
2	the Levenshtein ranks better	121	11.57



TABLE X. COMPARING THE PROFILE APPROACH ALONGSIDE WITH SAHEHLY AND MS WORD CHECKERS

Error types	Error types %	Sahehly alongside with the profile-based		MS Word alongside with the profile-based	
		Sahehly	profile	MS Word	profile
adjacent letter	4.49%	0.67%	3.82%	0.48%	4.02%
alef magsora and yaa confusion	8.89%	2.77%	6.12%	5.07%	3.82%
deletion error	4.02%	0.38%	3.63%	0.48%	3.54%
dialect error	0.48%	0.00%	0.48%	0.29%	0.19%
double letter	0.38%	0.10%	0.29%	0.10%	0.29%
exchange latter	0.19%	0.00%	0.19%	0.10%	0.10%
hamza error	58.22%	19.79%	38.43%	7.46%	50.76%
insertion error	2.01%	0.00%	2.01%	0.10%	1.91%
multiple error	10.42%	2.39%	8.03%	2.77%	7.65%
permanent error	1.24%	0.10%	1.15%	0.19%	1.05%
press adjacent letters at same time	0.48%	0.00%	0.48%	0.00%	0.48%
shape error	0.10%	0.00%	0.10%	0.10%	0.00%
shift keyboard	0.10%	0.00%	0.10%	0.00%	0.10%
taa marbota error	8.99%	4.02%	4.97%	5.26%	3.73%
	100.00%	30.21%	69.79%	22.37%	77.63%

TABLE XI. COMPARING THE SPELLING CORRECTIONS FOR A USER WHO HAS A CERTAIN PRIVACY

Error types	Error types %	Sahehly alongside with the profile-based		MS Word alongside with the profile-based	
		Sahehly	profile	MS Word	profile
Hamza_error	88.89%	24.44%	64.44%	4.44%	84.44%
multiple_error	6.67%	0.00%	6.67%	0.00%	6.67%
taa_marbota_error	4.44%	2.22%	2.22%	0.00%	4.44%
	100.00%	26.67%	73.33%	4.44%	95.56%

The second approach does not take into consideration the Levenshtein distance but checks whether the position of the correct word is among the first top 4 positions as it is the average number of proposed words given in most spellcheckers. For instance, Table VIII shows that all the desired words rank among the first four positions, except for the desired word of the misspelled word “الفقرى” (“AlfqrY”). The experiment in the entire corpus shows that the profile ranks the desired word as part of the 4 candidates in 625 times representing a percentage of 75.14%.

In the third approach, the 1046 errors in the compiled corpus were corrected with Sahehly and then compared with our tool. In the third approach, the 1046 errors in the compiled corpus were corrected with Sahehly and then compared with our tool. ours ranked better in 69.79% times than Sahehly. In the last approach, the errors were also tested with MS-Word and then compared with our tool. The result of the tool was also 77.63% times better than MS-Word (30.21%). Table X shows the details of the comparison of the tools. It shows the percentage share of each type of error in the total errors. In addition, the percentage of correction of each type of error is compared between the profile and the other tools.

We also find that the profile approach gives better results in terms of privacy of a particular user. For example, Table XI shows the results for a 19-year-old user who has no

experience in using computer. The profile achieved 73% times of the results compared to 27% for Sahehly and 96% times compared to 4% for MS-Word.

Experiments have shown that the proposed approach closely candidates the desired words associated with the current user profile. This relationship has not been clearly demonstrated in previous studies.

8. SAFAR SPELL CHECKER UI

The prototype is implemented to execute the proposed architecture (Figure 5). It contains, firstly, the profile variables that a current user fills in. Then, h(sh)e types a text. When the ‘spell-check’ button is clicked, the correction steps are displayed in the result area.

Figure 5 shows all the details of the results frame after filling the following profile (age=between 21 and 40, education=above university, job title=employee, print document=yes, year=11 to 15 years, computer dealing=a lot of dealing) and then typing the text “عن ما يتحدث القوم وفي أي الشئون يعملوت” (“En mA ytHdS Alqwm wfy Oy Al\$}wn yEmlwt”). When the “Spell check” button is clicked, the text is displayed and the incorrect words are highlighted. Also, by right-clicking on an incorrect word, the suggested words are displayed in a pop-up menu to choose the appropriate word. Moreover, the “Details” button shows details of the execution (Figure 6).

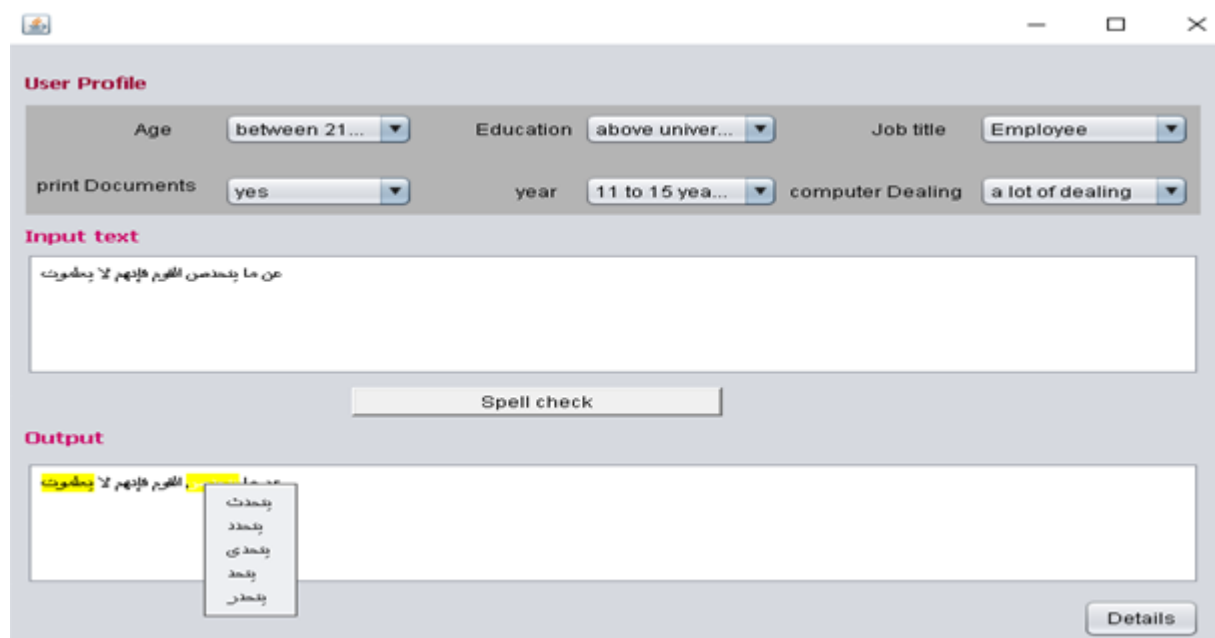


Figure 5. The prototype system

Phase1: starting rules checking ...

word=ما عن, changed to =عما

done rules checking

Downloading vocabulary ... done

Phase 2: starting detect incorrect words using corpus ...

incorrect word 1: يتحدث

incorrect word 2: يعلمون

done detect incorrect words

Phase 3: calculating user profile values ...

add_space = 4.6659044698932746E-4

dialect_error = 2.596726687609965E-4

double_letter = 2.893543704800627E-4

exchange_latter = 2.533926383925373E-4

Hamza_error = 3.336859260315717E-4

lam_shmsia = 0.9798471514101957

press_adjnet_letters_at_same_time = 0.004066840022306117

speed_error = 0.008351261170125326

alef_magsora_and_Yaa_confusion = 2.3158953823509702E-4

adjacent_letter= 1.8664970289697708E-4

deletion_error= 4.012277868077643E-4

double_word= 6.535650324000482E-5

forget__press_on_space= 4.7206997381166253E-4

insertion_error= 0.0011437190422881882

permanent_error= 5.44910393917437E-4

shift_keyboard= 0.002971319899758517

taa_marbota_error= 1.0998743666360168E-4

shape_error= 5.2210690990595054E-6

done calculating user profile values

Phase 4: ranking suggested words...

incorrect word 1: يتحدث

/ Leventesion(1- يتحاص, 2- يتحد, 3- يتحدا, 4- يتحدث, 5- يتحدث)

/ User profile (1- يتحدث, 2- يتحدد, 3- يتحدى, 4- يتحد, 5- يتحدر)

incorrect word 2: يعلمون

/ Leventesion(1- يعلمو, 2- يعلموا, 3- يعلموش, 4- يعلموك, 5- يعلمون)

/ User profile (1- يعلمون, 2- يعلمه, 3- يعلموا, 4- يعلموه, 5- يعلموك)

done.

Figure 6. Example of output



In phase 1, the execution of the spelling rules is checked, allowing the “عن ما” (“En mA”) error to be directly corrected to “عما” (“EmA”) without processing the similarity. In phase 2, the execution of the remaining words is processed using the vocabulary, allowing us to recognize that “يتحدث” (“ytHdS”) and “يعملوت” (“yEmlwt”) as misspelled words. Based on the filled profile variables, the user profile execution for each error type is calculated in phase 3. In the last fourth phase, the execution of the misspelled words is processed, and then the suggested words for each of them are ranked by Levenshtein alongside the profile distance. For the misspelled word “يتحدث” (“ytHdS”), our tool ranks the desired word “يتحدث” (“ytHdv”) in the fourth position when using Levenshtein. While using the profile, it ranks in the second position. For the second misspelled word, “يعملوت” (“yEmlwt”), our tool ranks the desired word “يعملون” (“yEmlwn”) in the third position with Levenshtein while the profile ranks it in the first position.

CONCLUSION

The paper presented a new spell-checking approach based on the user profile and that can be customized for any language. For this purpose, we analyzed the collected users' files and then obtained 18 error types and statistical values for each of them.

This approach determines a current user profile to give h(im/er) appropriate statistical values. Then, spelling errors are initially corrected by applying the rules. Next, the remaining words are identified based on their error types and then corrected using their profile values. In the experimental results, the profile position ranked better 88.43% times compared with the Levenshtein rank.

Based on our implemented approach experiments, the next generation spell checker should rely on the profile instead of relying directly on the similarity distances. It will suggest the nearest words based on a current user profile and provide accurate results for confusing words. In addition, this checker will interact with users, especially those whose native language is not Arabic or who are unfamiliar with Arabic rules.

The key contribution of our work opens the space for researchers to create a platform that can be applied for any language, that accommodates all users with their different levels and provides them with satisfactory results.

In the future, we plan to extend the work in order to address the following issues:

- Consider and directly process other kinds of errors without calling the distances.
- Take into consideration the space and double-word errors.

- Dynamically detect a current user profile instead of requesting to fill the profile form as is the case today.
- We were able to consider some of the user profile variables and there are certainly other variables that need further investigation such as those based on psychological studies.
- Since the composed vocabulary is large, we load it first, but we aspire to study the time complexity in the future to improve this limitation.

REFERENCES





- [1] D. A. H. F. Altamimi, R. Ab Rashid, and Y. M. M. Elhassan, “A Review of Spelling Errors in Arabic and Non-Arabic Contexts,” *English Language Teaching*, vol. 11, no. 10, pp. 88–94, 2018.
- [2] M. M. Al-Jefri and S. A. Mahmoud, “Context-Sensitive Arabic Spell Checker Using Context Words and N-Gram Language Models,” in *2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences*. IEEE, 2013, pp. 258–263.
- [3] A. M. Azmi, M. N. Almutery, and H. A. Aboalsamh, “Real-Word Errors in Arabic Texts: A Better Algorithm for Detection and Correction,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1308–1320, 2019.
- [4] A. Protopapas, A. Fakou, S. Drakopoulou, C. Skaloumbakas, and A. Mouzaki, “What do spelling errors tell us? Classification and analysis of errors made by Greek schoolchildren with and without dyslexia,” *Reading and Writing*, vol. 26, no. 5, pp. 615–646, 2013.
- [5] J. Atserias, M. Fuentes, R. Nazar, and I. Renau, “Spell-checking in Spanish: The case of diacritic accents,” in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 2012, pp. 737–742.
- [6] A. Kaur, P. Singh, and S. Rani, “Spell checking and error correcting system for text paragraphs written in Punjabi language using hybrid approach,” *International Journal Of Engineering And Computer Science*, vol. 3, no. 9, pp. 8030–8032, 2014.
- [7] D. Hládek, J. Staš, and M. Pleva, “Survey of automatic spelling correction,” *Electronics (Switzerland)*, vol. 9, no. 10, pp. 1–29, 2020.
- [8] “Buckwalter transliteration.” [Online]. Available: https://en.wikipedia.org/wiki/Buckwalter_transliteration
- [9] T. Zerrouki and A. Balla, “Implementation of infixes and circumfixes in the spellcheckers,” in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009.
- [10] S. Awwad, “Arabic Word stemming Based on Pattern Affixes Removal,” in *2019 10th International Conference on Information and Communication Systems (ICICS)*. IEEE, 2019, pp. 1–6.
- [11] K. Shaalan, M. Attia, P. Pecina, Y. Samih, and J. van Genabith, “Arabic Word Generation and Modelling for Spell Checking,” in *LREC*, 2012, pp. 719–725.
- [12] K. Shaalan, R. Aref, and A. Fahmy, “An approach for analyzing and correcting spelling errors for non-native Arabic learners,” in *2010 The 7th international conference on informatics and systems (INFOS)*. IEEE, 2010, pp. 1–7.
- [13] N. Mohammed and Y. Abdellah, “The vocabulary and the morphology in spell checker,” *Procedia Computer Science*, vol. 127, pp. 76–81, 2018.
- [14] M. Nejja and A. Yousfi, “A lightweight system for correction of Arabic derived words,” in *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*. Springer, 2016, pp. 131–138.

- [15] H. Bouamor, H. Sajjad, N. Durrani, and K. Oflazer, "QCMUQ@QALB-2015 Shared Task: Combining Character level MT and Error-tolerant Finite-State Recognition for Arabic Spelling Correction," in *Proceedings of the Second Workshop on Arabic Natural Language Processing*, 2015, pp. 144–149.
- [16] N. AlShenaifi, R. AlNefie, M. Al-Yahya, and H. Al-Khalifa, "Arib @ QALB-2015 Shared Task: A Hybrid Cascade Model for Arabic Spelling Error Detection and Correction," in *Proceedings of the Second Workshop on Arabic Natural Language Processing*, 2015, pp. 127–132.
- [17] B. Mohit, A. Rozovskaya, N. Habash, W. Zaghouni, and O. Obeid, "The First QALB Shared Task on Automatic Text Correction for Arabic," in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 39–47.
- [18] K. Shaalan, A. Allam, and A. Gomah, "Towards automatic spell checking for Arabic," in *Proceedings of the 4th Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE), Cairo, Egypt*, 2003, pp. 21–22.
- [19] H. F. Alshahad, "Arabic Spelling Checker Algorithm for Speech Recognition," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 15, no. 12, pp. 228–235, 2018.
- [20] S. C. Wayland, C. A. Rytting, D. Zajic, T. Buckwalter, J. White, C. Miller, J. Carnes, N. Lynn, P. Rodrigues, M. Maxwell et al., "Finding Entries in an On-line Arabic Dictionary," in *Human-Computer Interaction Lab 27th Annual Symposium*. Citeseer, 2010, pp. 1–2.
- [21] M. M. Alamri and W. J. Teahan, "Automatic correction of Arabic dyslexic text," *Computers*, vol. 8, no. 1, p. 19, 2019.
- [22] H. M. Noaman, S. S. Sarhan, and M. Rashwan, "Automatic Arabic Spelling Errors Detection and Correction Based on Confusion Matrix- Noisy Channel Hybrid System," *Egypt Comput Sci J*, vol. 40, no. 2, pp. 54–64, 2016.
- [23] M. I. Alkanhal, M. A. Al-Badrashiny, M. M. Alghamdi, and A. O. Al-Qabbany, "Automatic stochastic arabic spelling correction with emphasis on space insertions and deletions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 7, pp. 2111–2122, 2012.
- [24] Y. Abdellah, A. S. Lhoussain, G. Hicham, and N. Mohamed, "Spelling correction for the Arabic language-space deletion errors," *Procedia Computer Science*, vol. 177, pp. 568–574, 2020.
- [25] T. Zerrouki, K. Alhawiti, and A. Balla, "Autocorrection Of Arabic Common Errors For Large Text Corpus QALB-2014 Shared Task," in *Proceedings of the EMNLP 2014 workshop on arabic natural language processing (ANLP)*, 2014, pp. 127–131.
- [26] N. Sharma, "Automatic Question Generation from Punjabi Text: A Review," *International Journal of Computer Engineering & Application*, pp. 7–19, 2015.
- [27] K. Shaalan, "Rule-based Approach in Arabic Natural Rule-based Approach in Arabic Natural Language Processing," *The International Journal on Information and Communication Technologies (IJICT)*, vol. 3, no. 3, pp. 11–19, 2010.
- [28] A. A. Saty, K. B. Bouzoubaa, and A. S. Lhoussain, "Survey of Arabic Checker Techniques," *Journal of Engineering and Computer Science (JECS)*, vol. 21, no. 1, pp. 34–41, 2020.
- [29] Y. Hassan, M. Aly, and A. Atiya, "Arabic Spelling Correction using Supervised Learning," in *the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, 2014, pp. 121–126.
- [30] M. Attia, M. Al-Badrashiny, and M. Diab, "Priming Spelling Candidates with Probability," in *Proceedings of the Second Workshop on Arabic Natural Language Processing*, vol. 10, 2015, p. 138–143s.
- [31] F. Al-Najjar, "Spelling rules in ten easy lessons," *Al Kawthar Library*, vol. Fourth Edi, 2008.
- [32] "Resources – alelm." [Online]. Available: http://arabic.emi.ac.ma/alelm/?page_id=273/#Corpus
- [33] "Arabic Corpus." [Online]. Available: <https://sourceforge.net/projects/arabiccorpus/files/khaleej-2004corpus%28windows-1256%29/>
- [34] "NEMLAR Written Corpus." [Online]. Available: <http://metashare.elda.org/repository/browse/nemlar-written-corpus/baf9e9a0de6711e2b1e400259011f6eaa11bfe7fed041b9b094afac6a99ab58/>
- [35] T. Zerrouki and A. Balla, "Tashkeela: Novel corpus of Arabic vocalized texts, data for auto-diacritization systems," *Data in brief*, vol. 11, pp. 147–151, 2017.
- [36] "Arabic Corpus." [Online]. Available: <https://sourceforge.net/projects/arabiccorpus/files/>
- [37] "arabiclearncorpus." [Online]. Available: <https://www.arabiclearncorpus.com/>
- [38] "Abu El-Khair Corpus." [Online]. Available: <http://www.abuelkhair.net/index.php/en/arabic/abu-el-khair-corpus>
- [39] "Software Architecture For Arabic." [Online]. Available: <http://arabic.emi.ac.ma/safar>
- [40] K. Bouzoubaa, Y. Jaafar, D. Namly, R. Tachicart, R. Tajmout, H. Khamar, H. Jaafar, L. Aouragh, and A. Youfi, "A description and demonstration of SAFAR framework," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2021, pp. 127–134.
- [41] "TEI (Text Encoding Initiative)." [Online]. Available: <https://tei-c.org/>
- [42] V. V. Bhaire, A. A. Jadhav, P. A. Pashte, and P. Magdum, "Spell checker," *International Journal of Scientific and Research Publications*, vol. 5, no. 4, pp. 5–7, 2015.
- [43] B. Haddad and M. Yaseen, "Detection and Correction of Non-Words in Arabic: A Hybrid Approach," *International Journal of Computer Processing of Oriental Languages*, vol. 20, no. 04, pp. 237–257, 2007.
- [44] "Alpha Level (Significance Level): What is it? - Statistics How To." [Online]. Available: <https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/what-is-an-alpha-level>
- [45] C. Kwak and A. Clayton-Matthews, "Multinomial logistic regression," *Nursing research*, vol. 51, no. 6, pp. 404–410, 2002.
- [46] "Wald test." [Online]. Available: <https://www.statisticshowto.com/wald-test/>
- [47] "Sahehly." [Online]. Available: <https://sahehly.com>



Ahmed Abdalrhman Saty is currently a research fellow in computer science at the Sudan University of Science and Technology. He worked as a teacher at Alzaeem Azhary University for two years. He works in the Ministry of Finance of Sudan in the IT department. He is a developer (Java and Dotnet Framework)



Si Lhoussain Aouragh     is a full professor at the National High School of Computer Science and Systems Analysis (ENSIAS) at Mohamed V University in Rabat. He is a member of the information, communication, embedded systems and natural language processing team at ENSIAS. He is a member of the Arabic language engineering and learning modeling (ALELM) research group at the Mohammadia School of Engineers. He is president of the association of Arabic language engineering in Morocco.

His main research interests include computational linguistics, artificial intelligence, machine learning, natural language processing, and data science.



Karim Bouzoubaa is a full professor at the Mohammadia School of Engineers at the Mohammed V University in Rabat. He is the founder and leader of the Arabic Language Engineering and Learning Modeling (ALELM) research group. He is also the co-founder and past president of the Arabic Language Engineering Society in Morocco. His main research interests include Artificial Intelligence, Multi-Agent Systems, Machine and Deep Learning, Natural Language Processing, Data Science and Computational

Linguistics.