Efficient Weighted Edit Distance and N-gram Language Models to Improve Spelling Correction of Segmentation Errors

Hicham GUEDDAH
Intelligent Processing and Security of Systems Team- F.S.R,
E.N.S, Mohammed V University in Rabat,
B.P:8007, Avenue des Nations Unies, Agdal, Rabat, Morocco.
https://www.researchgate.net/profile/Hicham-Gueddah

***Abstract*—In most research that has dealt with the correction of spelling errors, the errors are caused by the misuse of space (deletion or insertion of space) are not tackled. Forgetting to deal with this type of errors in the texts poses a problem of understanding and ambiguity of the meaning of the sentence containing these errors. In this article, we propose a new approach to correct errors due to the insertion of space in a word, and at the same time correct other types of editing errors. This approach is based on the edit distance and uses bi-grams language models to correct words in context. The test conducted on hundreds of erroneous words (by insertion of space and/or by simple editing errors) made it possible to assess the relevance and validity of the methods developed to correct this type of error. The approaches proposed in this article provide a very important clarification and reminder by comparing them to those of other existing approaches.**

***Keywords*—Spelling correction; error; natural language; insertion; space; distance; language models; probability**

I. Introduction

For several years now, the language industry and new information and communication technologies have been evolving. Alongside this progress, thousands of electronic documents such as newspapers, emails, blogs, dissertations and theses are produced on a daily basis. Therefore, the existence and need for spelling correction systems in NLP applications is of paramount importance to improve and help with sound and unambiguous writing.

Automatic spelling error correction is currently ubiquitous and integrated into all computer tools such as word processing, email, social media, search engines, which are frequented every day by millions of people around the world.

For a long time, by analyzing the strategy of correction systems integrated into large word processing software such as Microsoft's WinWord, OpenOffice Writer, or those embedded in web textures (email, search engine), we have pointed out that these remain ineffective for correcting certain types of spelling errors. They're committed when typing Arabic text, for example, which we cite as errors resulting from insertion and/or untimely deletion of the space character in a lexical form.

Their strategies consist only in proposing solutions separately to segments divided by the insertion of space.

Automatic correction of spelling errors has been the topic of much research since the 1960's [1]. Despite the monopolization of this axis by the major computer production industries,it remains a promising domain of research [2][3]. The principle is to propose the most similar solutions to a word detected out of vocabulary based on the lexical similarity inter words.

Among the work in the subject area of spelling correction, we mainly include:

Damerau's analysis of typographical errors [4]. He indicated that about 80-95% of mistakes in English texts are unique errors that are induced by poor insertion, deletion, permutation of a single character or the transposition of two adjacent character. This analysis has been the cornerstone of the concept of error as a simple or multiple combinations of operations, called elementary editing operations (insertion, deletion, transposition, and permutation).

Based on Damerau's work, Levenshtein [5] considered only three editing operations (insertion, deletion, permutation), and subsequently defined a metric that allows us to compare two words while calculating the number of editing operations undergone on one word to turn it into some other word. This distance is also called edit distance which remains, despite the technique, the most widely used in spelling correction and which has also been the themes of several adaptations and weighting [6]. Then a series of similar works were carried out. We can put them into different categories:

- Metric-based correction approaches such as Jaro distance [7], Jaro-Winkler distance [8], Jaccard distance [9], distance of Stoilos [10].
- Probabilistic correction approaches such as n-grams decomposition [11], the correction method based on the noisy channel model [12], Alpha-code methods [13], or those based on probabilistic automatons [14], [15]
- Since 2012 Gueddah, Yousfi and Nejja have carried out a series of work on spelling correction for the Arabic language, with the aim of improving the scheduling rate of solutions returned by the classical edit distance [16] [17] [18], or integrating the morphological analysis into the spelling correction phase [19] [20], or integrating context into spelling correction [21] [22].

II. Related Work

Based on our bibliographical research on spelling correction, we noticed that the bulk of this work did not adequately address the errors due to insertion or deletion of space in the seized texts. The number of works dealing with this category of error is very negligible compared to the number of works in the field of spelling correction.

The work that dealt with errors due to the insertion or deletion of space is of two types:

- Census-type studies: Mitton [3], and Kukich [1] noted that more than 14% of spelling mistakes in seized text are mainly related to the omission of the space between words. Conforming to a statistical study of errors made in Urdu typed text, Tahira [23] found out that space errors are a fairly significant percentage of errors, more than 75% of errors are neighborhood errors, 32% of which are linked to the insertion of space in words.
- Research work that provides hints for dealing with errors due to the insertion/deletion of space in words, either at the level of detection of this type of error or in the actual correction. This work includes, for example, the work that relies on the generation of all possible partitions of the erroneous word and testing whether or not the segments exist in the dictionary [24][25][26].
- The study presented by Alkanhal and al [27], according to the latter, the correction of space insertion errors uses two procedures, the initiative is to merge the different neighboring words from the wrong word. The outcome of this procedure is a list of the various possible combinations of this merger. This list may contain valid fragments and other erroneous fragments that will subsequently be handled on to the second correcting procedure.

In this article, we suggest a new metric approach that uses bi-gram language models to correct spelling errors due to the insertion of spaces (also called segmentation errors), into a correct or incorrect word taking into account the context in which this type of error was induced.

III. Errors Due to Space Insertion and Deletion

A. *Defining Space Errors*

Although the emergence of new generations of near-present correctors in most word processing editor, emails, blogs, social media, smartphones, these remain ineffective and unsuccessful [28] with regard to correcting errors due to the insertion or deletion of space in or between words. These forms of mistakes can be induced in several situations:

- Because one writes by accelerating without regard to who has been seized, the editor may unintentionally insert one or more spaces within a word in the belief that he inserted it to separate between two words. Therefore, this type of error leads to the appearance of two or more segments of words that may be lexicon or wrong words.
- Moreover, this kind of error can be due to poor optical text recognition (OCR), which can additionally insert the space character inside the word, or as a result of a file type conversion, such as converting Word documents to Pdf or vice versa.

However, the problem of actual word error is more complex. Generally, such an error disrupts the syntax and then rectify it.

Example:

Instead of typing the word ”misspelled”, you add a space in that word and you get both sequences ”missp” and ”elled”. Instead of typing the two words ”to get vaccinated”, you remove the space between these two words, and you get the only word ”toget vaccinated”.

There are cases where this type of error is combined with other types of editing errors, i.e. in the same word w, we have errors due to editing operations more than a space insertion(after inserting the space we have both words w_1 and w_2).

In this case, there are four cases:

- Both segments w_1 and w_2 are not changed. In this case the solution of the correction is simply w_1-w_2-w, for example ”vacci nating instead of vaccinating”.
- The first segment w_1 has been modified, and w_2 not, for example: ”vaxi nating instead of vaccinating”.
- w_2 has been modified and w_1 not, example of ”vacci nathing instead of vaccinating”.
- Both segments w_1 and w_2 are modified, for example: ”vaxin ating instead of vaccinating”.

In the last three cases we must stick the two segments w_1 and w_2 (w_1-w_2) and then correct the new merged word.

In the recently published work Yousfi and al.[30], the authors used and adapted the Levenshtein’s algorithm to detect and correct errors due to space deletion between words in the case of Arabic texts.

In this paper, we will propose a new approach to correct space insertion errors, and to integrate it with other approach that which corrects deletion errors and other editing errors, in a single one that corrects these three types of error at the same time.

B. *Introducing the Approach to Correcting Deletion Space Errors*

Either w_{err} a erroneous word of length n, and w_i a word of the lexicon of p_i length. Among the errors in the wrong word w_{err} perhaps the space that is removed between several words more other types of editing errors (insertion, deletion, and permutation).

The approach is done in two stages:

- The detection of the position in the word w_{err} where space will be inserted.
- The correction of the two sequences obtained after the insertion of the space.

We note by: $D_L(w_{err}, w_i) = D_L(n, p_i)$, Levenshtein's distance between two words w_{err} and w_i. For the detection of the position where space will be inserted (noted $pos.space_i$), the authors gave the following rule:

$$pos.space_i = \underset{j=1,...,n}{\arg\min}\, DL(j, p_i) \tag{1}$$

After the insertion of space, we get the two words w_{1space_i}, w_{2space_i}, and we move on to the second phase.

The second phase verifies whether the two words exist in the lexicon or not, otherwise we move on to the correction. To correct errors due to the deletion of spaces between words and other types of editing errors at the same time, the authors defined a new distance (noted D_{LS} based on the Levenshtein distance (edit distance):

$$\begin{aligned} D_{LS}(w_{err}, w_i) = Min[D_L(w_{err}, w_i); D_L(w_{err}, w_{1space_i}) \\ +space + w_{2space_i}] \\ where\, D_L\, is\, the\, edit\, distance \end{aligned} \tag{2}$$

The scheduling of solutions is based on this new D_{LS} distance.

In the sequel, we will present the new approach we propose in this paper to correct errors in the insertion of space inside word, and then we will show how to integrate it with the previous approach in a single one. Then all types of spelling errors can be corrected (editing errors, errors due to deletion and insertion of space).

IV. The Approach to Correcting Space Insertion Errors

This approach is based on the edit distance and on the bi-grams language models. In the rest of this paragraph, we will give a little reminder on these two concepts.

A. The Edit Distance

The metric method introduced by Levenshtein [5] measures the similarity between two words by calculating an edit distance. The edit distance is defined as the minimum number of basic editing operations required to turn an erroneous word into another word in the dictionary. Thus, to correct a wrong word, we retain a set of solutions requiring as few editing operations as possible.

The procedure for calculating the distance of Levenshtein between two strings $X = x_1x_2 \ldots x_m$ of length m and $Y = y_1y_2 \ldots y_n$ of length n, consists of calculating from near to near in a matrix of order $(m * n)$ the edit distance between the different sub-chains of X and Y.

The calculation of the case (i,j), which corresponds to the editing distance between the sub-chains $X_1^i = x_1x_2 \ldots x_i$ and $Y_1^j = y_1y_2 \ldots y_j$, is given by the following recurring relationship:

$$D(i,j) = Minimum \begin{cases} D(i-1,j)+1, \\ D(i,j-1)+1, \\ D(i-1,j-1)+cost \end{cases} \tag{3}$$

with

$$cost = \begin{cases} 0 & \text{if} \quad x_{i-1} = y_{j-1} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

The limitation of such a spelling correction system using the edit distance is that it does not allow a good scheduling of suggested solutions for a set of candidates with the same edit distance.

B. N-gram Language Models

The importance of language models is quite clear. They are used in several areas of NLP, such as continuous speech recognition, machine translation, etc. The main goal in these different applications is to have some solutions weighed against others.

A n-gram language model shows the fact that the probability of a word appearing after a sequence of words can be given only on the basis of the last $n-1$ words [29].
This model verifies:

$$Pr(w_i/w_1, w_2..., w_{i-1}) = Pr(w_i/w_{i-n+1}, .., w_{i-1}) \tag{5}$$

In practice, the value of n does not exceed order 3.

- If $n = 1$, the model is called a uni-gram model. This type of model does not take into account any history of the word.
- If $n = 2$, the model is called a bi-gram model. This type of model only takes into account the previous word: $Pr(w_i/w_1, w_2..., w_{i-1}) = Pr(w_i/w_{i-1})$
- If $n = 3$, the model is called a tri-gram model. This type of model takes into account only the previous two words: $Pr(w_i/w_1, w_2..., w_{i-1}) = Pr(w_i/w_{i-1}, w_{i-2})$

For the construction of n-gram language models, learning is done on a corpus of texts that must encompass all possible successions of words belonging to the vocabulary of the language used. This construction consists of estimating all the probabilities already mentioned.

V. Processing Space Insertion Errors

The processing of this type of error normally goes through the following two steps: the detection that one has an error due to the insertion of space into a correct or erroneous word, and the phase of correction.

A. Detection of Errors Due to the Space Insertion

Here we cite methods that are not 100% correct to detect errors in inserting space into a correct or erroneous word. We have cases where the probability of having a space insertion error is very high.

Among these cases we cite:

- If two successive words w_1 and w_2 are erroneous, then the probability is very high to have inserted a space that gave us these two wrong words. In this case, we concatenate the two words, and we treat $w_1 - w_2$ as a single word that must be submitted to the correction procedure. However, this is not always the case as we can have two successive erroneous words without having a space insertion error (”The wolrd vaxinated against Covid” instead of ”The world vaccinated against Covid”

- If we have a word consisting of a single character, then this may be due to a bad insertion of space into a word (”v accination” instead of ”vaccination”.
- If we have some kind of erroneous word and the following word is correct, it may be due to a space insertion into a word (”vacci nation” instead of ” vaccination”).

So from what we have presented, it is very difficult to fix cases and say that they are the only ones that exist for space insertion errors.

There are even cases where you insert a space into a word and you get two sequences that are both correct (”foot ball” instead of ”football”).

For this, we will treat the problem of space insertion according to two methods:

- **Method 1**: The space insertion error is only addressed if we have two successive words that are wrong, working with an H1-rated hypothesis that will be defined afterwards. If we have a single erroneous word, we correct it in a simple way.
- **Method 2**: Once we have a wrong word and regardless of the next word, plus the simple correction of that wrong word, we add the corrections of the error due to the wrong insertion of space between that wrong word and the two neighboring words.

B. Correcting Space Insertion Errors using Method 1

This method uses the following hypothesis:

$\boxed{\text{If we have a correct word, then we should not change it.}}$

Be a text $T = w'_1 w'_2 \ldots w'_n$ consists of a set of words typed in that order, and suppose that we have two successive erroneous words w'_i and w'_{i+1}. To also take into account space insertion errors, and use the H1 hypothesis, all the corrections proposed for these errors consist of two sets:

- All simple corrections of the two words w'_i and w'_{i-1} (assuming we don’t have errors in inserting or deleting space). This set is given by the following rule:

$$S_i^1 = \operatorname*{arg\,min}_{w_k \in Lexique} \frac{D_L(w'_i, w_k)}{Pr(w_k / w'_{i-1})} \tag{6}$$

and

$$S_{i+1}^1 = \operatorname*{arg\,min}_{w_k \in Lexique, w_i \in S_i^1} \frac{D_L(w'_{i+1}, w_k)}{Pr(w_k / w_i)} \tag{7}$$

- we Concatenate w'_i and w'_{i+1} (w'_i+w'_{i+1}), i.e. assume that we have inserted a space in the word $w'_i + w'_{i+1}$ which produced the two wrong words w'_i and w'_{i+1}. For correction, we check if $w'_i + w'_{i+1}$ is in the lexicon of the system. If so, we keep the word $w'_i + w'_{i+1}$ as an ideal solution with edit distance equal to zero; otherwise we make the correction with the edit distance weighted by the bi-gram language model:

$$S_i^2 = \operatorname*{arg\,min}_{w_k \in Lexique} \frac{D_L(w'_i + w'_{i+1}, w_k)}{Pr(w_k / w'_{i-2})} \tag{8}$$

To take into account space insertion errors during the correction operation, we propose the distance rated D_{AL} which is defined as a follow-up:

$$D_{AL}(w'_i, w_k) = Min[\frac{D_L(w'_i, w_k)}{Pr(w_k / w'_{i-1})} + \frac{D_L(w'_{i+1}, w_k)}{Pr(w_k / w'_i)}, \frac{D_L(w'_i + w'_{i+1}, w_k)}{Pr(w_k / w'_{i-2})}] \tag{9}$$

The best corrections are those that check :

$$\min_{w_k \in Lexique} D_{AL}(w'_i, w_k) \tag{10}$$

C. Correcting Space Insertion Errors using Method 2

For this method, we do not use the H1 hypothesis; and in this case, we can modify words that were correct. The list of solutions or corrections for the erroneous word w'_i proposed is of three types:

- The set of all simple corrections of the word w'_i (assuming we have no errors in the insertion or deletion of space). This set is given by the rule given in equation number (6)
- we concatenate w'_{i-1}, correct word, and w'_i ($w'_{i-1} + w_i$) and check if $w'_{i-1} + w'_i$ in the lexicon of the system. If so, we keep the word $w'_{i-1} + w'_i$ as an ideal solution with edit distance equal to zero, otherwise the correction is made with the edit distance weighted by the bi-gram language model :

$$S_i^2 = \operatorname*{arg\,min}_{w_k \in Lexique} \frac{D_L(w'_{i-1} + w'_i, w_k)}{Pr(w_k / w'_{i-2})} \tag{11}$$

- We concatenate w'_i and w'_{i+1} (w'_i+w'_{i+1}) and we check if w'_i+w'_{i+1} is in the lexicon of the system. If so, we keep the word w'_i+w'_{i+1} as an ideal solution with distance equal to zero; otherwise the correction is made with the edit distance weighted by the bi-gram language model:

$$S_i^3 = \operatorname*{arg\,min}_{w_k \in Lexique} \frac{D_L(w'_i + w'_{i+1}, w_k)}{Pr(w_k / w'_{i-1})} \tag{12}$$

To take into account space insertion errors during the correction operation, we apply the D_{AL} distance, which this time is defined as follows:

$$D_{AL}(w'_i, w_k) = Min[\frac{D_L(w'_i, w_k)}{Pr(w_k / w'_{i-1})}, \frac{D_L(w'_{i-1} + w'_i, w_k)}{Pr(w_k / w'_{i-2})}, \frac{D_L(w'_i + w'_{i+1}, w_k)}{Pr(w_k / w'_{i-1})}] \tag{13}$$

The best corrections of the wrong word w'_i are given by the following formula:

$$\underset{w_k \in Lexique}{\arg\min} \ D_{AL}(w'_i, w_k) = \underset{w_k \in S_i^1 \cup S_i^2 \cup S_i^3}{\arg\min} [\frac{D_L(w'_i, w_k)}{Pr(w_k/w'_{i-1})}, \frac{D_L(w'_{i-1} + w'_i, w_k)}{Pr(w_k/w'_{i-2})}, \frac{D_L(w'_i + w'_{i+1}, w_k)}{Pr(w_k/w'_{i-1})}] \quad (14)$$

Its solutions belong to one of the sets of S_i^1, S_i^2 and S_i^3 :

1) If a solution belongs to S_i^1, we correct w'_i by this solution, and we go to w'_{i+1} to check whether or not this word exists in the lexicon; otherwise we repeat our correction approach quoted here on w'_{i+1}
2) If a solution belongs to S_i^2, we removew'_{i-1}, we correct w'_i by this solution, and switchto w'_{i+1}. We check w'_{i+1} whether we exist in the lexicon or not, or we apply our correction approach to w'_{i+1}.
3) If the solution belongs to S_i^3, we correct w'_i with this solution, we delete w'_{i+1}, and we move to w'_{i+2}. We check whether w'_{i+2} exists in the lexicon or not, otherwise we apply our correction approach to w'_{i+2}.

When the correction is completed, the new sentences are scheduled with the original sentence, $w'_1 w'_2 \ldots w_n$ by the edit distance taking the space as a character: $D_L(w'_1 w'_2 \ldots w'_n, w'_1 w'_2 \ldots, w'_{i-2}, w_{i-1} \ldots, w_k) \quad k \leq n$

Example:
"The world vaccinated against Covid"
We have two successive erroneous words, so we do the processing according to the two methods.

Method 1:

- The corrections of the two erroneous sequences "vacc" and "inated" are "vacc" ={ vaccine, vice, Vicki, vaccines..}, and "inated"={noted, anted, named, mated, hated...}. The best corrections are distance 1, so the sum 1+1=2.

- The corrections of the word "vaccinated", after deleting space between sequences "vacc"and "inated" we obtain this word which is a lexicon entry.
- Applying the distance D_{AL} the min between 0 and 2 is 0 and like that, the best correction is "vaccinated" ("The world vaccinated against Covid")

Method 2:

- The first wrong word is "vacc", so the three sets without taking into account the language models in the formulas, S1, S2, and S3 are:
  - S1- corrections of the wrong word "vacc" ={ vaccine, vice, Vicki, vaccines..}
  - S2- corrections of the wrong word "worldvacc"={∅}, in reality no suggestion
  - S3- the corrections of the word "vaccinated", this word is a lexicon entry.

TABLE I. Recall and accuracy of different methods

	Recall	Accuracy
Method 1	82%	91%
Method 2	76%	88%

Words in S1 have a distance greater than or equal to 1, and words in S2 have a distance greater than or equal to 2. So the words that check the rule in equation (14) is "vaccinated" that is, the solution belongs to S3, so we correct "vaccinated" and we delete "inated", So the new sentence after the correction is: "The world vaccinated against Covid".

VI. Tests and Results

The corpus on which we conducted our test is composed of 100 paragraphs taken from the Wikipedia site, and in each paragraph, we randomly created space insertion errors in words in those paragraphs in addition with other types of editing errors.

In total, we have 1000 errors due to the insertion of space into words that are correct or erroneous. These errors are created according to the four types of space insertion errors cited in subsection (3.1.)

In order to evaluate the different methods used for spelling correction, we use the classic evaluation measurements by calculating recall and accuracy. The results obtained are cited according to these four types of error, and are as mentioned the Table I. In order to test the robustness of our approach with these two methods, we compared it to a widely recognized commercial spell checker.

The first results approved that our approach achieves a very high correction rate compared to this corrector: 89.5 % of the suggested correction for errors due to the insertion against only 19.12 % for the commercial corrector. this can be interpreted by the fact that this corrector does not take into account the existence of this type of error due to the insertion.

Conclusion

According to the results obtained, we can say that our new proposed approach is an effective method to correct errors due to space insertion comparing with other commercial spell checker. In other hand, our approach present an acceptable and better complexity level compared with other approach based only on n-gram language models.

References

[1] Kukich K.," Techniques for automatically correcting words in text ", ACM Computing Surveys (CSUR),Volume 24, Issue 4, pp: 377-439, 1992.

[2] Mitton R., "Spelling Checkers, Spelling Correctors, and the Misspellings of Poor Spellers ",in Information Processing & Management, Volum23, issue 5, pp: 495-505, 1987.

[3] Mitton R., " Spellchecking by computer ", in Journal of the Simplified Spelling Society, Volum 20, Issue 1, pp :4–11, 1996.

[4] Damerau F.J.," A Technique for computer detection and correction of spelling errors", Communications of the ACM, Volum 7, Issue 3, pp: 171-176, March 1964.

[5] Levenshtein V.I," Binary codes capable of correcting deletions, insertions, and reversals ", Soviet Physics Doklady, pp: 707-710, 1966.

[6] HORST B., "A Fast Algorithm for Finding the Nearest Neighbor of a Word in a Dictionary ". IAM-93-025, 1993

[7] Jaro M. A.," Advances in record-linkage methodology as applied to matching the census of Tampa", Journal of the American Statistical Association, pp : 414-420, Florida, 1985.

[8] Winkler W. E.," The State of Record Linkage and Current Research Problems ", Statistical Society of Canada, Proceedings of the Section on Survey Methods, pp: 73-79, 1999.

[9] Jaccard P.," The Distribution of the flora in the alpine zone", New Phytologist, Volume 11, pp: 37-50, 1912.

[10] Stoilos G., Stamou G., Kollias S., "A string Metric for Ontology Alignment ", International Semantic Web Conference, pp: 624-37, 2005

[11] Angell Richard C., Freund George E. et Willett P.," Automatic spelling correction using a trigram similarity measure", Information Processing and Management, Volume 19, Num. 4, pp: 255-261, 1983.

[12] Kernighan M.D, Church K.W. and Gale W.A.," A Spelling correction program based on a noisy channel model ", In Proceeding of the 13th Conference on Computational Linguistics, pp : 205-210, 1990.

[13] Pollock J. J. and Zamora A.," Automatic spelling correction in scientific and scholarly text", Communications of the ACM, Volume 27, Num.4, pp: 358-68, 1984.

[14] Oflazer K.," Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction", Computational Linguistics, Volume 22, Num. 1, pp: 73-98, 1996.

[15] Savary A.," Recensement et description des mots composés - méthodes et applications", Thèse de doctorat en Informatique Fondamentale, Université de Paris 7, pp : 149-158, 2000.

[16] Gueddah H., Yousfi A. and Belkasmi M.," Introduction of the weight edition errors in the Levenshtein distance", International Journal of Advanced Research in Artificial Intelligence, Volum 1, Issue 5, pp : 30-32, 2012.

[17] Gueddah H. et Yousfi A.," Impact de la proximité et de la similarité inter-caractère Arabe sur la correction orthographique ", Proceeding of the 8th International Conference on Intelligent Systems : Theories and Applications- SITA 13, pp : 244-246, EMI-Rabat 2013.

[18] Nejja M., Yousfi A., "A lightweight system for correction of Arabic derived words", in Mediterranean Conference on Information & Communication Technologies, Volum 1, pp: 131-138, Saïdia 2015.

[19] Bakkali H., Gueddah H., Yousfi A. and Belkasmi M.," For an Independent SpellChecking System from the Arabic Language Vocabulary ", Proceeding of International Journal of Advanced Computer Science and Applications, Volume 5 Issue 1, pp : 114-116, Janvier 2014.

[20] Nejja M., Yousfi A., " Contexts impact on the automatic spelling correction", in International Journal of Artificial Intelligence and Soft Computing archive, Volum 6, Issue 1, pp: 56-74, 2017.

[21] Gueddah H., Aouragh L., et Yousfi A.," Adaptation de la distance de Levenshtein Pour la correction orthographique contextuelle ", Proceeding du 9éme Conférence Internationale sur l'Intelligence Artificielle : Théories et Applications, SITA 14, pp : 242-245, INPT- Rabat 2014.

[22] Nejja M., Yousfi A., " Contexts impact on the automatic spelling correction", in International Journal of Artificial Intelligence and Soft Computing archive, Volum 6, Issue 1, pp: 56-74, 2017.

[23] Naseem T., Hussain S., "A novel approach for ranking spelling error corrections in Urdu ", Language Resources and Evaluation, Volum 41, Issue 2, pp: 117–28. DOI 10.1007/s10579-007-9028-6, Springer Science+Business Media B.V, 2007.

[24] Naseem T., Hussain S., "A Hybrid Approach for Urdu Spell Checking ", MSc thesis, National University of Computer & Emerging Sciences, Pakistan, 2004

[25] Attia M., Pecina P., Samih Y., Shaalan K., et Genabith J.V., "Arabic Spelling Error Detection and Correction ", in: Natural Language Engineering, Volum 22, Issue 5, pp: 751-773 Cambridge University Press, 2016.

[26] Maha M. A., William J. T., "Automatic Correction of Arabic Dyslexic Text". Computers 2019, Volum 8, Issue 1, 2019.

[27] Alkanhal M. I., Al-Badrashiny M. A., Alghamdi M. M., and Al-Qabbany A. O., "Automatic Stochastic Arabic Spelling Correction With Emphasis on Space Insertions and Deletions ", in IEEE Transactions on Audio, Speech, and Language Processing Volum 20, Issue 7, pp: 2111–2122, 2012.

[28] Alexis Amid Neme, " Why Microsoft Arabic Spell checker is ineffective ", Linguistica Communication, http://www.al-erfan.com/, 2014, Arabic Language in Information Technology, 16, pp.55, hal01081965.

[29] G. Hirst, "An Evaluation of the Contextual Spelling Checker of Microsoft Office Word 2007 ", Department of Computer Science university of Toronto Toronto, Canada 2008.

[30] Yousfi A., Aouragh L., Gueddah H and Nejja M, " Spelling correction for the Arabic language: space deletion errors ", in Procedia Computer Science, pages: 568-574, Volum 177, 2020,ISSN 1877-0509.