Arabic spellchecking: a depth-filtered composition metric to achieve fully automatic correction

Hicham Gueddah, Youssef Lachibi
Intelligent Processing and Security of Systems, Faculty of Sciences, Mohammed V University, Rabat, Morocco

Article Info

Article history:

Received Jun 29, 2022
Revised Feb 20, 2023
Accepted Mar 9, 2023

Keywords:

Composition
Correction
Distance
Filtration
Learning
Probability
Weighting

ABSTRACT

Digital environments for human learning have evolved a lot in recent years thanks to incredible advances in information technologies. Computer assistance for text creation and editing tools represent a future market in which natural language processing (NLP) concepts will be used. This is particularly the case of the automatic correction of spelling mistakes used daily by data operators. Unfortunately, these spellcheckers are considered writing aids tools, they are unable to perform this task automatically without user's assistance. In this paper, we suggest a filtered composition metric based on the weighting of two lexical similarity distances in order to reach the auto-correction. The approach developed in this article requires the use of two phases: the first phase of correction involves combining two well-known distances: the edit distance weighted by relative weights of the proximity of the Arabic keyboard and the calligraphical similarity between Arabic alphabet, and combine this measure with the Jaro-Winkler distance to better weight, filter solutions having the same metric. The second phase is considered as a booster of the first phase, this use the probabilistic bigram language model after the recognition of the solutions of error, which may have the same lexical similarity measure in the first correction phase. The evaluation of the experimental results obtained from the test performed by our filtered composition measure on a dataset of errors allowed us to achieve a 96% of auto-correction rate.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Hicham Gueddah
Intelligent Processing and Security of Systems, Faculty of Sciences, Mohammed V University
B.P:8007, Av. Nations Unies, Agdal, Rabat, Morocco
Email: h.gueddah@um5r.ac.ma

1. INTRODUCTION

Over the past several years, theoretical linguistics, computational linguistics and new information and communication technologies have evolved remarkably. As a result of these advances, thousands of electronic documents such as newspapers, emails, blogs and personal and professional documents (thesis, final study projects, and scans) are produced every day. Often we type the text by rushing without revising what has been typed, where the error takes a privileged place in the typed text.

Therefore, the existence and necessity of spelling correction systems in word processing applications that are of paramount importance to improve and assist an effective and unambiguous writing. Given this need, automatic spelling correction applications are currently ubiquitous and are integrated into all computer tools such as word processing, email, social media and information retrieval, search engines, which are frequented used every day by millions of people in the world. This necessity is for more effective writing and to remove the degree of ambiguity in the text because the error is economically expensive.

In the field of natural language processing, the research axis of automatic spell checking and correction remains the most important and the oldest among the other axes of natural language processing (NLP). Research in this area dates back to the 1960s [1], [2] and continues to the present. The first studies in the field of automatic correction had as a first goal to modelize the notion of spelling error, according to Damerau [3] and later Levenshtein [4], an error is considered as a simple or multiple combinations of the elementary editing operations relative to the insertion, deletion and permutation of characters inside a lexicon word. Based on this modeling, several methods and algorithms have been suggested for spelling correction. We distinguish between two large categories of correction methods: combinatorial methods and metric methods. Combinatorial methods [5], [6] consist in generating all the possible sequences from which the erroneous word could be derived by applying elementary editing operations and yielding only the sequences belonging to the lexicon [7], [8].

Metric methods consist in comparing the erroneous word with the entries of a given dictionary while calculating a lexical similarity measure [9]. The solutions to the erroneous form are those with a minimum metric. Metric methods are qualified as the best methods because they yield better results and are implemented in spelling correction systems [10]. Another different method uses the dictionary search and morphological analysis module for the Indonesian language as a spelling correction strategy [11]–[14]. The major limitation raised in metric-based correction methods is that they do not differentiate between several solutions having the same lexical similarity measure. This finally requires these spellcheckers to get the assistance of the user to negotiate the nearest solution to the erroneous word, so that the user can finally select the correct one.

Our research target and main goal in this article is how to improve the correction process in order to reach the stage of achieving a fully automatic spell checker. In other words, is it possible to develop a fully automatic spelling correction system without negotiating solutions with the writer?. So that the first solution suggested in the solutions list is the one desired by the user. To reach this goal, we launched a learning test on a training corpus in order to identify and analyze the nature of the spelling errors committed, and based on this analysis, we integrated these estimated parameters, probabilistic weights related to elementary editing operations, in our learning corpus to improve the spelling correction process [15].

After collecting our corpus of typed texts, we proceeded to identify and analyze the errors committed by the operators and arranging the misspelled words with their corresponding correct words in a database. Generally, the spelling errors committed include: insertion, deletion, and permutation known as elementary editing operations. Table 1 gives the statistics of the error rate according to the type of elementary editing operations calculated from our training test.

Table 1. Error rate of editing operations according to our learning test

Editing operations	Insertion	Deletion	Permutation
Error rate	14.23%	20.77%	65%

– Analysis of permutation errors
  The first question raised according to the Table 1 was to know the reason why data entry operators commit enough errors of the type of permutation. The analysis of the permutation errors led us to conclude that they are mainly caused by two main factors. The first factor is the proximity between Arabic keyboard keys and the second factor is the calligraphic similarity between some Arabic characters [16].
– Analysis of deletion and insertion error
  If the analysis of permutation errors led us easily to find a direct link between the permuted characters in a lexical word, this was not the case for the analysis of deletion and insertion errors. Nevertheless, after consulting the distribution of the keys in the Arabic keyboard, we concluded that there are two different interpretations of the results of analysis of these two types of errors: 85.30% of the erroneous insertion and/or deletion of characters in the lexical word depend on the proximity on the left or on the right of the keys of Arabic keyboard. However, 14.70% of the analyzed characters do not depend on the proximity of the keys of keyboard, such as the insertion or deletion of blank spaces.

As shown in Table 2, we illustrate our interpretations with examples of insertion and deletion errors. We have added two columns to try to demonstrate whether there is a proximity relationship on the keyboard between the inserted or deleted characters (see Arabic keyboard Figure 1). For example, for the first erroneous word and corresponding to the database, we led that the erroneous insertion of the character "غ" is due to the proximity on the left and on the right between the characters "ع" and "ا".

Table 2. Relationship between Arabic keyboard keys and insertion/deletion errors

				Proximity	
Error	Matching correct word	Editing Operation	Character	On the left	On the right
تضاغعفت	تضاعفت	insertion	غ	+(ع / غ)=yes	+(ا / غ)=yes
اسنكر	استنكر	deletion	ت	-(ن / ت)=yes	-(س / ت)=no
الخالطة	المخالطة	deletion	م	-(خ / م)=yes	-(ل / م)=no
تجاوت	تجاوزت	deletion	ز	-(ت / ز)=no	-(و / ز)=yes
اجتهجدت	اجتهدت	insertion	ج	+(د / ج)=yes	+(ه / ج)=no
حدثة	حداثة	deletion	ا	-(ث /ا)=no	-(د /ا)=no

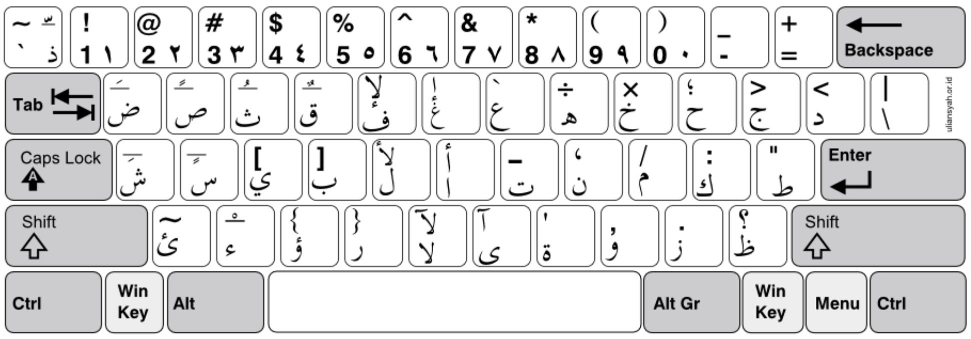

Figure 1. Arabic keyboard

According to these statistics and the different interpretations deduced, we can confirm that the majority of the editing errors made (insertion, deletion, permutation) are caused by the proximity and similarity of the character keys on the Arabic keyboard [17], [18]. In the rest of this paper, we will modelize these interpretations as matrices probabilistic weights. These weights will be assigned for each elementary editing operation for calculating lexical similarity during the spelling correction process [19].

2. THE PROPOSED METHOD

In this article, we propose to assign probabilistic weights, which are related to the proximity between the keys of the keyboard and to the calligraphic similarity between the Arabic characters. These probabilistic weights will be assigned to the different editing operations during the calculation of lexical similarity between the erroneous word and arabic dictionary entries for a spelling correction method based on edit distance. The analysis of permutation errors allowed us to conclude that permutation can be caused by two factors: the proximity between the keys of the Arabic keyboard or the degree of calligraphic similarity between some arabic characters. For example the calligraphic similarity is very high between: (ض,ص), (ق,ف), (غ,ع), (ت,ث), (خ,ج,ح), (ش,س), (ط,ظ), (د,ذ), (ر,ز), (ى,ي), (ك,ل).

Subsequently, We are modeling these factors as a proximity matrix and a calligraphic similarity matrix between Arabic alphabets in order to introduce them into the Levenshtein algorithm [4]. Then, these will be tested to examine whether this weighting edit distance [20] will help improve the correction rate so as to better refine the scheduling of the closest solutions to the detected erroneous word in order to achieve an effective auto-correction [21], [22].

2.1. Definition of the weighted edit distance

The lexical similarity calculation between two sequences $X = x_1x_2\ldots x_m$ of length m and $Y = y_1y_2\ldots y_n$ of length n, is done using a new measure,called weighted edit distance and noted Ed_{wei}. The calculation is given by the following recurring relationship: We note $Ed_{wei}(i, j)$ the weighted edit distance between the two substrings $\underset{1<i\leqslant m}{X_1^i} = x_1 x_2 \ldots x_i$ and $\underset{1<j\leqslant n}{Y_1^j} = y_1 y_2 \ldots y_j$, $Ed_{wei}(i,j) = Ed_{wei}(X_1^i, Y_1^j)$

$$Ed_{wei}(i,j) = Minimum \begin{cases} Ed_{wei}(i-1,j) - Cost_{del}(x_{i-1}), \\ Ed_{wei}(i,j-1) - Cost_{ins}(y_{j-1}), \\ Ed_{wei}(i-1,j-1) - Cost_{per} \end{cases} \tag{1}$$

With

$$Cost_{per} = \begin{cases} 0 & \text{if} \quad x_{i-1} = y_{j-1} \\ 1 - Permut(x_{i-1}, y_{j-1}) & \text{otherwise} \end{cases} \tag{2}$$

As a result:

- $Cost_{del}(x_{i-1})$= the cost of deleting the character x_{i-1} given by the $Proxim(x_{i-1}/y_j)$
- $Cost_{ins}(y_{j-1})$= the cost of insertion the character (y_{j-1}) given by the $Proxim(x_i/y_{j-1})$
- $Permut(x_{i-1}, y_{j-1})$: cost of permutation between characters x_{i-1} and y_{j-1}.
- $Proxim(x_i/x_j)$: cost relied to the proximity between the keyboard keys characters x_i and x_j.

To test and evaluate the interest of our new weighting edit distance, we launched a test on a dataset of errors of different editing operations kinds. We randomly select 547 errors extracted from our learning corpus. Figure 2 summarizes the the scheduling rates of the different ranks of exact solutions according to the erroneous word/correct word database.

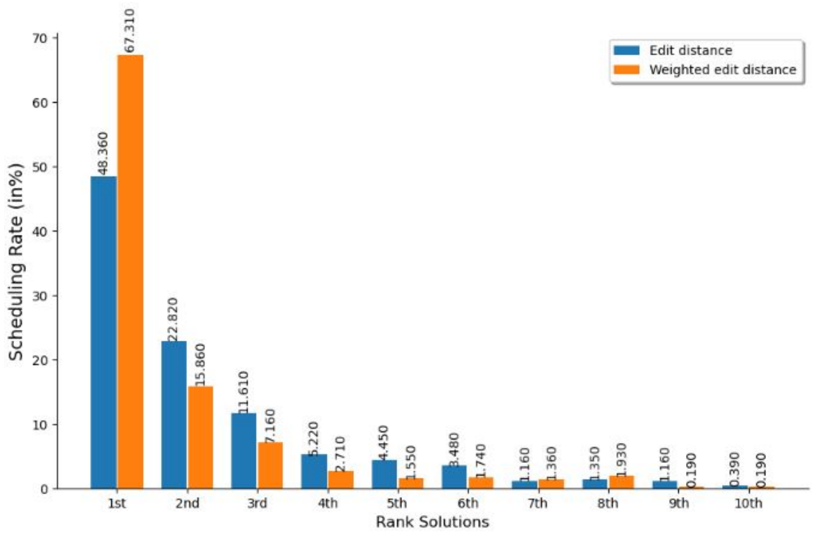

Figure 2. Comparison of correction rates between edit distance and weighted edit distance

According to these results, we found that our weighting has remarkably improve the rate of scheduling with a score of around 67.50% to achieve autocorrection. In fact, even if we integrate probabilistic weights to improve the correction rate, we have not succeeded in reaching our objective of auto-correction. Our second step is to look for another filter to increase the correction rate.

After a deep analysis of the nature of spelling errors committed in typing Arabic texts (learning test), we discovered that the majority of elementary editing operation errors are committed either at the beginning of the word or in the middle and are rarely committed at the end of the word. This valuable information discovered greatly helped us to choose another composition with our previously defined weighting edit distance.

Among the distances used in the similarity measure, we find that of Jaro-Winkler [23], [24] which is more adapted to our new situation as well as the n-gram language models [25], [26]. This composition will give very good results and can be combined with our weighting to further optimize the weighting, and finally achieving our end objective. The choice of the Jaro-Winkler distance is justified because the majority of editing errors are made in the beginning of the lexical word.

2.2. The filtered composition approach

To summarize all that has been deduced in this paper, we define in this section our filtered composition (composition of the three metrics: weighted edit distance, Jaro-Winkler distance and the bi-gram probabilistic language models). More formally, the similarity measure, called D_{cf}, between X and Y is obtained by the following relationship:

$$D_{cf}(X,Y) = \frac{Ed_{wei}(X,Y) \times (1 - J_{wink}(X,Y)}{Pr(X/Y)} \tag{3}$$

Where:

– $Ed_{wei}(X,Y)$: the weighted edit distance between strings X and Y
– $J_{wink}(X,Y)$: the Jaro-winkler distance between strings X and Y
– $Pr(X/Y)$: bi-gram language model probability of a word X appearing after a word Y estimated on a corpus of correct forms. Let's w_{err} an erroneous word, D_{ict} a dictionary of a given language and S_{ols} a set of the proposals closest solutions of the error w_{err} where $S_{ols}=\{ \underset{1<i\leqslant 8}{s_i}$ with $s_i= \underset{w_j \in D_{ict}}{\operatorname{argmin}} D_{edit}(w_j, w_{err})\}$, D_{edit} is the edit distance.

According to the new measure, filtered composition, the best corrections are those that check:

$$\min_{w_k \in D_{ict}} D_{cf}(s_i, w_k) \tag{4}$$

As shown in Figure 3, our spelling correction procedure consists of:

– Check if W_{err} is in the Lexicon
– Otherwise, calculate the edit distance with the words of the lexicon, and return only 10 solutions having a minimum distance.
– The list of solutions will then be passed as parameters to our filtered composition to weight solutions relative to others.
– Finally return another list generated by our probabilistic lexical measure and test if the word corresponding to the error is in the first position of the list.

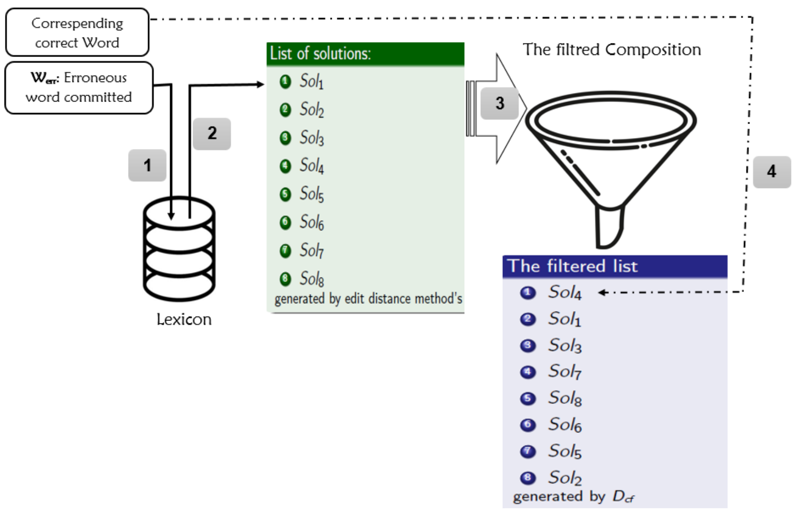

Figure 3. General pattern of our spelling correction sytem

Example: In our training corpus, the data entry operator commited the following error, “السيق”. The corresponding correct word is “السيف” in our learning dataset. The following example in Table 3, shows how our new measure floated the corresponding word to the commited error to the first position by the other suggested solutions.

Table 3. Improving the scheduling rank of the solutions

Rank	$Solts$	E.D	$Solts$	Ed_{wei}	$Solts$	Jw	$Solts$	D_{cf}
1	الريق	1	السبق	0.27	السيَاق	0.96	السيف	0.0217
2	السَاق	1	السيف	0.27	السيف	0.92	السَاق	0.0225
3	السبق	1	الشيق	0.28	السين	0.92	السبق	0.0251
4	السجق	1	السين	0.51	السَاق	0.90	الشيق	0.0262
5	السيَاق	1	السَاق	0.66	السبق	0.90	السين	0.0409
6	السيف	1	الريق	0.696	السجق	0.906	السَاق	0.0513
7	السين	1	السيَاق	0.69	الريق	0.893	الريق	0.0631
8	الشيق	1	السجق	0.84	الشيق	0.89	السجق	0.0749

3. RESULTS AND DISCUSSION

In order to apply and evaluate the relevance of our new approach in terms of correction rate (accuracy rate) and in order to validate our design choices, we launched a spelling correction test using our new measure: the filtered composition. These tests were done on a set of 547 erroneous words extracted from our error corpus. This test set of errors are of different kinds of elementary editing operations.

In this test, we evaluate more precisely the performance rate of our new measure in order to see if this measure will correct exactly the errors by ranking the corresponding correct words in the first position of the suggested solutions list. At the end of this test, we compared the accuracy rates of the different distances mentioned in this study. According to the obtained results in Figure 4, we can confirm that our filtered composition achieved a rather high accuracy rate compared to the other measurements cited in this article.

Our proposed similarity measure, filtered composition, achieved an precision rate up to 96%, which means that 96% of the returned solutions, ranked in the first position of solution lists, are the correct words that correspond to the erroneous words in our training corpus. While for the other measures studied in this article their precision rates do not exceed 68%, respectively: 67.70%, 67.31% and 48.36% for Jaro Winkler, Weighted edit distance and edit distance.

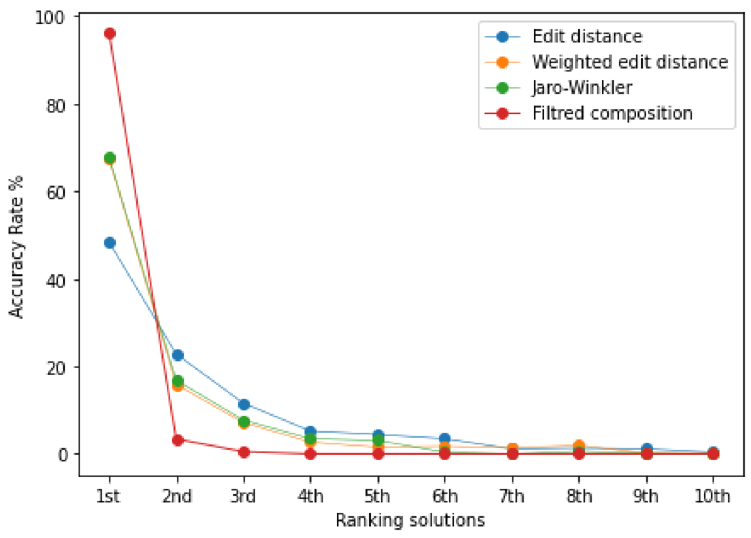

Figure 4. Accuracy rate of the different measurements

However, our D_{cf} measure ranked 3.41% of the solutions (correct words) in the second position of the list. This is can be interpreted by the fact that some solutions have the same weighted edit distance metric and the same probabilistic bigram model langauge. As an example, the two solutions "الهَادي" and "الهندي" for the erroneous word "الهتدي", have the same probabilities of appearance in front of the word "المحيط", and the same measures returned by Jaro-Winkler and weighted edit distance.

4. CONCLUSION

In this paper, we presented a new approach to the spelling correction of errors in Arabic texts. This approach is based on a filtered composition of robust lexical similarity distances and probabilistic language models. From a training phase, we were able to assign probabilistic weights to the different elementary editing operations in the edit distance, the fact that the data operators committed editing errors either at the beginning or in the middle guided us to introduce the Jaro-Winkler distance and finally the use of the bi-gram language model to better weigh, refine and filter some solutions against to the others.

The experimental results carried out on a corpus of errors made it possible to achieve our objective in this work: it is to reach a highly effective auto-correction. The attribution of probabilistic weights to the different editing operations during the calculation of the editing distance allowed us to reach a very high rate of auto-correction, which validates the choices of our conception. The obtained results are very encouraging and show the interest to embed our new measure in a spelling correction system.

REFERENCES

[1] K. Kukich, "Techniques for automatically correcting words in text," *ACM Computing Surveys*, vol. 24, no. 4, pp. 377–439, Dec. 1992, doi: 10.1145/146370.146380.

[2] R. Mitton, "Fifty years of spellchecking," *Writing Systems Research*, vol. 2, no. 1, pp. 1–7, Jan. 2010, doi: 10.1093/wsr/wsq004.

[3] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, Mar. 1964, doi: 10.1145/363958.363994.

[4] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, 1966.

[5] R. Mitton, "Spellchecking by computer," in *English Spelling and the Computer*, vol. 20, no. 1, 1996, pp. 4–11.

[6] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, 2000, pp. 286–293, doi: 10.3115/1075218.1075255.

[7] K. Oflazer, "Error-tolerant finite state recognition with applications to morphological analysis and spelling correction," *Computational Linguistics*, vol. 22, no. 1, pp. 69–89, Apr. 1995.

[8] L. Karttunen and K. Oflazer, "Introduction to the special issue on finite-state methods in NLP," *Computational Linguistics*, vol. 26, no. 1, pp. 1–2, Mar. 2000, doi: 10.1162/089120100561593.

[9] R. C. Angell, G. E. Freund, and P. Willett, "Automatic spelling correction using a trigram similarity measure," *Information Processing and Management*, vol. 19, no. 4, pp. 255–261, Jan. 1983, doi: 10.1016/0306-4573(83)90022-5.

[10] L. Salifou and H. Naroua, "Design and implementation of a spell checker for Hausa language (in French)," *Association pour le Traitement Automatique des Langues*, pp. 147–158, 2014.

[11] M. Y. Soleh and A. Purwarianti, "A non word error spell checker for Indonesian using morphologically analyzer and HMM," in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, Jul. 2011, pp. 1–6, doi: 10.1109/ICEEI.2011.6021514.

[12] A. H. Aliwy and B. Al-Sadawi, "Corpus-based technique for improving Arabic OCR system," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 21, no. 1, pp. 233–241, Jan. 2021, doi: 10.11591/ijeecs.v21.i1.pp233-241.

[13] B. Hamza, Y. Abdellah, G. Hicham, and B. Mostafa, "For an independent spell-checking system from the Arabic language vocabulary," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, 2014, doi: 10.14569/IJACSA.2014.050115.

[14] G. Hicham, Y. Abdallah, and B. Mostapha, "Introduction of the weight edition errors in the Levenshtein distance," *International Journal of Advanced Research in Artificial Intelligence*, vol. 1, no. 5, 2012, doi: 10.14569/IJARAI.2012.010506.

[15] A. B. Kiros and P. U. Aray, "Tigrigna language spellchecker and correction system for mobile phone devices," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2307–2314, Jun. 2021, doi: 10.11591/ijece.v11i3.pp2307-2314.

[16] G. Hisham and Y. Abdullah, "Impact of proximity and similarity between arabic characters on spelling correction (in French)," in *Proceeding of the 8th International Conference on Intelligent Systems: Theories and Applications*, 2013, pp. 24–246.

[17] C. B. O. Zribi and M. Ben Ahmed, "The context at the service of the correction of faulty Arabic spellings (in French)," in *Proceedings of The 10th Conference on Natural Language Processing*, 2003, pp. 409–414.

[18] C. B. O. Zribi and M. Ben Ahmed, "Efficient automatic correction of misspelled arabic words based on contextual information," in *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)*, vol. 2773, 2003, pp. 770–777.

[19] Y. Zhang, P. He, W. Xiang, and M. Li, "Discriminative reranking for spelling correction," in *Proceedings of the 20th*

Pacific Asia Conference on Language, Information and Computation, 2006, pp. 64–71.

[20] M. Mohri, "Edit-distance of weighted automata: general definitions and algorithms," *International Journal of Foundations of Computer Science*, vol. 14, no. 6, pp. 957–982, Dec. 2003, doi: 10.1142/S0129054103002114.

[21] H. Gueddah, M. Nejja, S. Iazzi, A. Yousfi, and S. L. Aouragh, "Improving SpellChecking: an effective Ad-Hoc probabilistic lexical measure for general typos," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 27, no. 1, pp. 521–527, Jul. 2022, doi: 10.11591/ijeecs.v27.i1.pp521-527.

[22] R. Mitton, "Ordering the suggestions of a spellchecker without using context," *Natural Language Engineering*, vol. 15, no. 2, pp. 173–192, Apr. 2009, doi: 10.1017/S1351324908004804.

[23] W. E. Winkler, "The state of record linkage and current research problems," in *Statistical Research Division*, 1999, pp. 1–15.

[24] M. A. Jaro, "Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 414–420, Jun. 1989, doi: 10.1080/01621459.1989.10478785.

[25] 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*, Dec. 2013, pp. 258–263, doi: 10.1109/NOORIC.2013.59.

[26] 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, Aug. 2019, doi: 10.1109/TASLP.2019.2918404.

BIOGRAPHIES OF AUTHORS

Hicham Gueddah is qualified professor in computer science at Department of Computer Science-ENS, Mohammed V University in Rabat- Morocco, joining the Intelligent Processing and Systems Security team- FSR, Mohammed V University since 2018. His research area and topic is Natural Language Processing (NLP), Text Mining and Deep Learning, especially the research axis of automatic correction. He has published several works in automatic correction. Further info on his homepage: https://www.researchgate.net/profile/Hicham-Gueddah. He can be contacted at email: h.gueddah@um5r.ac.ma.

Youssef Lachibi is a Ph.D. Student researcher in computer Science and Intelligent Artificial at the Intelligent Processing and Security of Systems Team-FSR at Mohammed V University in Rabat. Currently his working on processing medical images also include natural language processing all this using deep learning and machine learning. He can be contacted at email: youssef_lachibi@um5.ac.ma.