A Lightweight System for Correction of Arabic Derived Words

Mohammed Nejja and Abdellah Yousfi

Abstract In this paper, we address the lexicon insufficient problem used in the automatic spell checker. In order to solve that deficiency, we developed an approach that aims to correct the derived words, considering that the most Arabic words are derived ones by adjusting the Levenshtein algorithm to our need. This method is based on a corpus constituted of surface patterns and roots characterized by a scaled down size compared to conventional approaches The proposed method reduced the execution time while maintaining the highest correction rate.

Keyword ANLP (Arabic Natural Language Processing) · Misspelled word · Spell checker tool · Edit distance · Surface patterns

1 Introduction

According to Lunsford and Lunsford [1], the spelling mistakes are considered among the most common mistakes when writing a text. In fact, approximately 6.5 % of all errors detected in a US national sample of college composition essays were identified as misspellings.

Arabic is one of the languages in which the spelling mistakes are frequently detected. In fact, by comparing the rate of committing a misspelling in Arabic, French and English, the Arabic language has been identified as having the highest rate. This result is due to the fact that Arabic words are much closer to each other with an average number of related forms of 26.5 while English words and French words have an average number of related forms of 3 and 3.5 respectively [2].

M. Nejja (✉)
TSE Team, ENSIAS Mohammed V University in Rabat, Morocco
e-mail: mohammed.nejja@gmail.com

A. Yousfi
ERADIASS Team, FSJES Mohammed V University in Rabat, Morocco
e-mail: yousfi240ma@yahoo.fr

A. El Oualkadi et al. (eds.), *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*, Lecture Notes in Electrical Engineering 380, DOI 10.1007/978-3-319-30301-7_14

Automatic spelling correction is an active area where several studies have been developed to provide effective solutions for the gaps in this research area. As an example, Kukich [3] and Mitton [4] have presented a classical approach based on research in dictionaries. This approach aims at assessing whether the input string appears or not in the list of valid words. In case the string is missing from the dictionary then it is called erroneous chain. A second approach based on the calculation of the edit distance was presented by Drameau [5] and Levenshtein [6]. The main objective of this approach is to calculate the minimum number of operations required to move from a string to another. Pollock and Zamora [7] have also developed another approach that associate each dictionary word to a skeleton key. This technique is used to correct character duplication errors, deletion and insertion of some occurrences of character as well as accent mistakes.

For Arabic language, several correction techniques and studies have emerged and are available for exploitation, namely:

- Gueddah [8] suggested a new approach so as to improve planning solutions of an erroneous word in Arabic documents by integrating frequency editing errors matrices in the Levenshtein algorithm.
- A new approach has been advised by Bakkali [9] based on the use of a dictionary of the stems of Buckwalter to integrate morphological analysis in the Levenshtein algorithm.

In this paper, we present an improved extension of the approach proposed by Nejja and Yousfi [10]. The approach exposed in a previous work is based on using surface patterns to overcome the lexicon insufficient problem. Thereby, this approach aims at improving the identification accuracy of the surface pattern nearest to the wrong word so as to properly classify surface patterns with the same edit distance.

2 Correction by the Levenshtein Distance

The Levenshtein algorithm (also called Edit-Distance) calculates the least number of edit operations that are necessary to modify one string to obtain another string.

Elementary editing operations considered by Levenshtein are:

- Permutation (لعب [laåiba: to play] → لعت[laåita])
- Insertion (سمع [samiåa: to hear] → شسمع[šasamiåa])
- Deletion (جمع [jamaåa: to collect] → جع[jaå])

The Levenshtein algorithm uses the matrix of (N + 1) * (P + 1) (where N and P are the lengths of the strings to compare T, S) that allows calculating recursively the distance between the strings T, S. The matrix can be filled from the upper left to the lower right corner. Each jump horizontally or vertically corresponds to an insert or a delete, respectively. The cost is normally set to 1 for each of the operations. The diagonal jump can cost either one, if the two characters in the row and column do

not match or 0, if they do. The calculation of the cell M[N, P] equals to the minimum between the elementary operations:

$$\mathrm{M}(i,j) = \mathrm{Min}\begin{cases} \mathrm{M}(i-1,j)+1 \\ \mathrm{M}(i,j-1)+1 \\ \mathrm{M}(\mathrm{i}-1,\mathrm{j}-1)+\mathrm{Cost}(\mathrm{i}-1,\mathrm{j}-1) \end{cases} \quad (1)$$

where

$$\mathrm{Cost}(\mathrm{i},\mathrm{j}) = \begin{cases} 0 & \text{if } \mathrm{T}(i) = \mathrm{S}(j) \\ 1 & \text{if } \mathrm{T}(\mathrm{i}) \neq \mathrm{S}(j) \end{cases} \quad (2)$$

3 The Surface Pattern

Arabic pattern essentially aims at identifying the structure of most of the words. The patterns allow producing Stems from a root or conversely extracting the root of a word. The Patterns are variations of the word فعل [faåala] which are obtained by using diacritics or adding of affixes.

The surface pattern [11] is a way to present the morphological variations of words that are not submitted by the classical scheme. Example: The conjugation of the verb رَعَى [raåa] to the active participle in the 1st person singular is رَاعٍ [râåin]; therefore, the surface pattern of the root رَعَى [raåa] is فَعَى [faåa] and فَاعٍ [fâåin] is the surface pattern of رَاعٍ [râåin]. The surface pattern of آجِرٌ [ajiron] is آفِعٌ [afiåon] and of آجِرَاتٌ ['ajirâton'] is آفِعَاتٌ [afiåâton] [11].

4 The Morphological Correction by Surface Patterns in the Levenshtein Algorithm

An automatic spelling correction system is a tool that allows analyzing and eventually correcting spelling mistakes. To this end, that system uses a dictionary to compare the text's word to the dictionary's words.

However, the dictionary's size is considered as a major concern in the automatic spelling correction. In order to have an efficient automatic spelling correction system, those dictionaries need to contain all the words of the processed language as well as linguistic information of each word.

Some techniques are expected to use modules to calculate edition distances. Other techniques are meant for exploiting the morphological analysis. The objective of these techniques is to remedy the deficiency of the used dictionary.

In the same context, and in order to solve that deficiency, we developed an approach that aims to correct the derived words, considering that the most Arabic words are derived ones.

This approach consists in finding first the surface pattern that is the nearest lexically to the misspelled word. Then, the word is corrected through this surface pattern and using an assigned method. To identify the nearest surface scheme to the misspelled word, we adopted the Levenshtein algorithm to the Arabic language, and extended it in a way to select the nearest surface scheme to the word input.

We note by:

- $A = \{A_1, A_2, \ldots, A_n\}$: all patterns.
- W_{err}: erroneous word
- $\beta = \{'ل','ع','ف'\}$: the basic letter.

Therefore, the Levenshtein algorithm adapted to extract the correct surface patterns is defined (for all W_{err}, A_n) by:

$$M(k,p) = \text{Min}\begin{cases} M(k-1,p)+1 \\ M(k,p-1)+1 \\ M(k-1,p-1)+\text{Cost}(k-1,t-1) \end{cases} \tag{3}$$

where

$$\text{Cost}(k,p) = \begin{cases} 1 & \text{if } A(k) \sim = B(p) \text{ and } A(k) \notin \beta \\ 0 & \text{if } A(k) = B(p) \text{ or } A(k) \in \beta \end{cases} \tag{4}$$

We denote by U(k) the letter of the word U at position k.

4.1 Approach 1

This approach consists in finding the nearest surface pattern lexically to the misspelled word using the formula 3, then correcting the word through the identified surface pattern. For example, for the misspelled word شتلعيون [šatalåayûna] the nearest surface pattern is ستفعلون [satafåalûna] so the corrected word is ستلعيون [satalåayûna]. As soon as we have the corrected word, we extract the potential root based on the letters $\beta = \{'ل', 'ع', 'ف'\}$ of the identified surface pattern. For our example ستلعيون [satalåayûna], the potential root is لعي [laåaya]. We then compare that potential root with the roots in our base to get the nearest one in such a way that the root's size is equal to the size of β (the surface pattern تفعلل [tafaålala] has a root's size equal to 4 because the size of $\beta = \{'ل','ل','ع','ف'\}$ is 4). For our example, the correct root is لعب [laåiba]. At the end, we gather that information to construct the correct word, which is in our example ستلعبون [satalåabûna].

4.2 Approach 2

This new approach was developed to remedy the gap of the previous one. Indeed, the previous approach showed an issue when the deleted characters were characters of the root word. For example, if we use the previous approach with the misspelled word سضيرب [saDayribo] and the surface pattern سيفعل [sayafåalo] the character ض [D] will be deleted. Due to this gap, we improved the first approach in such a way that the deleted characters belong to both the surface pattern and the misspelled word and take up the same position in both of them. For the word شتتبون [šatatibûna] and the surface pattern ستفعلون [satafåalûna] the characters ون [ûna] will be deleted. That way, we make sure that we only deleted the characters that belong to the affixes of the concerned surface pattern.

That approach has proven its efficiency. Besides, and considering that the obtained results were satisfactory, we changed the formula (4) to improve the classifying of the selected surface pattern [12].

$$M(k,p) = \text{Min}\begin{cases} M(k-1,p)+1 \\ M(k,p-1)+1 \\ M(k-1,p-1)+\text{cost}(k-1,t-1) \end{cases} \tag{5}$$

where

$$\text{cost}(k,p) = \begin{cases} 1\ if\ \ A(k) \neq B(p)\ and\ \ A(k) \notin \{ل\ ع\ ف\} \\ 0\ \ if\ \ \ A(k) \in \{ل\ ع\ ف\} \\ 0{,}2\ \ if\ \ A(k) = B(p)\ and\ A(k) \notin \{ل\ ع\ ف\}\ and\ p = k \\ 0{,}5\ if\ A(k) = B(p)\ and\ \ A(k)\ \notin \{ل\ ع\ ف\}\ and\ p \neq k \end{cases} \tag{6}$$

The rectifications we introduced to the formula helped us improve the precision of identifying the surface pattern having the same Edit-Distance while displaying first the most adaptable solution (Fig. 1).

Fig. 1 An example of the result provided by our improvement

ستتبو

Edit Distance	Surface Patterns
1	ستفعلون
1	سيفعلون
2	سنفعل
2	نفعلوا

Befor the improvement

ستتبو

Edit Distance	Surface Patterns
1,9	ستفعلون
2,2	سيفعلون
2,2	سنفعل
2,2	نفعلوا

After the improvement

5 Test and Result

In order to evaluate the automatic spelling correction system efficiently, the ranking of the correct word in comparison with other candidates should be identified. And to achieve this, we have chosen, in a first place, to display the first 10 solutions for each erroneous word in order to obtain satisfactory results.

To test our method, we have performed a comparison between our approach and that of Levenshtein. This approach has been evaluated on 10000 erroneous words. For this reason, we considered:

- A training corpus containing 290 words for our approach (40 of surface patterns and 250 of root).
- A training corpus containing 10,000 words for the Levenshtein algorithm.
- Ever since the tests have been done, we have obtained a set of results:
- The characteristics of the used machine are:
- **System**: *Win XP.*
- **Memory**: *1G.*
- **Processor**: *Intel® Pentium® Dual CPU* 1.46 GHz. (Table 1).

We notice that our approach has reduced, considerably, the execution time which may be due to the lexicon size adopted by our method. In fact, the lexicon size used in this study is reduced compared to conventional approaches that are based on the edit distance.

Thereby, thanks to the improvement contributed to approach 2, we were able to increase classification accuracy of selected words while keeping a high rate of correction.

Table 1 Comparative table between our approach and method of Levenshtein

10,000 W_{err}	Method of Levenshtein	Approach 1	Approach 2
Nbr of solution in position 1 (%)	48.13	50.1	54.32
Nbr of solution in position 2 (%)	13.04	15.48	16.81
Nbr of solution in position 3 (%)	6.59	4.38	7.98
Nbr of solution in other position (%)	2.16	7.84	9.17
Correction rate (%)	69.72	77.8	84.84
Time/nbr W_{err} (ms)	0.14128	0.03178	0.03188

6 Conclusion

The automatic spell checker is a system that allows correcting spelling mistakes committed in a text, by using a set of methods often based on dictionaries. In fact, if the word processed by the system belongs to the adopted dictionary, it will be accepted as a word of the language; otherwise, the correction system will report it as a misspelled word and will suggest a set of similar words in it.

Inadequate vocabularies used in dictionaries are the major problem that hinders the most existing spell-checkers, consequently that requires a very large size to form a dictionary which can contain all possible terminologies.

Today, many works in ANLP were been done in order to focus on the development of independent methods of vocabulary's correction, by including morphological analysis, syntax, context, etc.

To remedy this problem, we were interested in this article in reducing the size of lexicon used. Thus, our proposed method deals with a particular case of derived words correction. This is due to the fact that most Arabic words are derived ones. Therefore, we have started by describing the way in which the Levenshtein algorithm was adapted to extract the nearest surface pattern of the erroneous word pattern. Then we have proposed a new solution for word's correction.

Thanks to our new approach, we were able to reduce the size of the dictionary, which reflects positively on the performance of our system while maintaining a higher coverage.

Among the performance criteria such spelling correction systems, we include the number of candidates proposed for a misspelled word. It is in this context that our next work is progressing. Actually, we aim to extend this study in order to reduce the number of words proposed candidates that have the same frequency of occurrence for a misspelled word.

References

1. Lunsford, A.A., Lunsford, K.J.: Mistakes are a fact of life: a national comparative study. Coll. Compos. Commun. **59**(4), 781–806 (2008)
2. Ben Othmane Zribi, C., Zribi, A.: Algorithmes pour la correction orthographique en arabe. TALN, 12–17 (1999)
3. Kukich, K.: Techniques for automatically correcting words in text. ACM Comput. Surv. **24**, 377–439 (1992)
4. Mitton, R.: English Spelling and the Computer. Longman Group (1996)
5. Damerau, F.J.: A technique for computer detection and correction of spelling errors. Commun. Assoc. Comput. Mach. (1964)
6. Levenshtein, V.: Binary codes capable of correcting deletions, insertions and reversals. SOL Phys Dokl, 707–710 (1966)
7. Pollock, J.J., Zamora, A.: Automatic spelling correction in scientific and scholarly text. Commun. ACM **27**(4), 358–368 (1984)
8. Gueddah, H., Yousfi, A., Belkasmi, M. :Introduction of the weight edition errors in the Levenshtein distance. Int. J. Adv. Res. Artif. Int., 30–32 (2012)

9. Bakkali, H., Yousfi, A., Gueddah, H., Belkasmi, M.: For an independent spell-checking system from the Arabic language vocabulary. Int. J. Adv. Comput. Sci. Appl. **5** (2014)
10. Nejja, M., Yousfi, A.: Correction of the Arabic derived words using surface patterns. In: Workshop on Codes, Cryptography and Communication Systems (2014)
11. Yousfi, A.: The morphological analysis of Arabic verbs by using the surface patterns. Int. J. Comput. Sci. Issues, **7** (2010)
12. Nejja, M., Yousfi, A.: Classification of the solutions proposed in the correction of the Arabic words derived using the use of surface patterns. In: Workshop on Software Engineering and Systems Architecture (2014)