International Journal of Computer Applications (0975 - 8887)
Volume 53 - No. 3, September 2012

Towards Arabic Spell-Checker Based on N-Grams Scores

Hasan Muaidi
Computer Science Department
Prince Abdullah bin Ghazi faculty of IT
Albalqa Applied University
Salt - Jordan

Rasha Al-Tarawneh
Computer Science Department
Aqaba University College
Albalqa Applied University
Aqaba - Jordan

ABSTRACT

The main purpose of this paper is to develop a simple and flexible spell-checker for Arabic language. The proposed spell-checker is based on N-Grams scores. For this purpose, eleven matrices are built to present the combination between the Arabic letters word. Each matrix concerns in the connection between a 2-grams letters. Each cell in the generarated matrix is assigned an integer value 2, 1 or 0. The cell is assigned the value 2 in the corresponding matrix; if the word is ended by these two letter and assigned 1 if there is a connection and the word is not over yet, and is assigned 0 otherwise. On the other side searching process for any word that is by extracting each pair of letters in the word then it examines the value for each pair when the corresponding value is zero then the spell checker will consider the test word as wrong; otherwise it will check if it is assign with 1 that indicates that there is a connection it will be continue until reach to the value of 2 to determine that the word is correct. The overall accuracy for the proposed spell-checker is reached to 98.99%.

KEYWORDS

Natural Languages Processing, Arabic Language Processing, Spell-Checker, N-Gram

1. INTRODUCTION

Arabic is an eternal language and it is considered as one of the oldest languages in the world. It is ranked the fifth in the widely used these days. Despite of Arabic language is actually one of the most widely spoken languages in the world it has been relatively little used in speech, writing programs and in data processing.

Unlike English and the Indo-European languages, Arabic text is oriented right to left without the use of capital letters. In addition, Arabic differs from other languages syntactically, morphologically, and semantically that make it one of the most difficult languages for written and spoken language processing [11]. Arabic has been increasingly used in many information retrieval systems and recently on the Internet. Investigation of methods of automatic spell-checker systems for Arabic text is essential to the growth of Arabic texts on the Internet and on other systems. The limitations of the current systems for spell-checker for Arabic languages induced the authors to investigate a simple and flexible approach to build a spell-checker system for Arabic language.

This paper describes a spell-checker problem which belongs to the area of computational linguistics. Many specialists in computer science consider computational linguistics as a part of artificial intelligence (AI)[1]. Computational linguistics might be considered as a synonym of automatic processing of natural language, since the main task of computational linguistics is just the construction of computer programs to process words and texts in natural language [1]. The main objective of the spell-checker system is to flag words in a text that may not be spelled correctly [5]. This flagging process is done at the word level without considering the text context.

The paper starts with a brief summary of Arabic spell-checker related work. In Section 3, some definitions are reviewed in order to clarify the terms used in this paper. The proposed Arabic spell-checker and the corresponding algorithm are presented in Section 4. The experimental results and the causes of the errors are discussed in Section 5. Finally, the conclusion is presented in Section 6.

2. RELATED WORK

The importance of spelling checkers in the field of digital processing of the natural languages is one of the matters that drew attention of the people interested in once solutions since the early beginnings of media. This prompted companies to work on the production of these tools - with hidden source for the most part - with more and more sophisticated functions that meet the needs of computer users in the once field in particular, and overlooks the services given by softwares' distributors who pay millions to get it (the price of the Arab spell-checker exceeds one million U.S. dollars in the global market for the year 2006) [9].

In literature the most known techniques for building a spell-checker are dictionary look up and n-gram. This section presents an outline of how these two techniques are achieved.

2.1 DICTIONARY LOOKUP TECHNIQUE

Dictionary lookup is one of the most important structures in computer science, it is represented as a set of data arranged in rows and columns. To search for one item; every value should be checked in the dictionary sequentially and stop when the value is found.

Dictionary lookup technique is easy to implement but the cost of searching in a large dictionary is often too high, farther more it is difficult to extract and store all the words of a given language in it [7].

2.2 N-GRAM TECHNIQUE

N-gram is n-letter subsequences of n-adjacent letters in a word, if n = 1, 2, 3 reference is made to a unigram, bigram, or trigram, respectively. N-gram analysis is used in spell-checker after compiled a table of n-gram binary values or frequency counts from large corpora, for comparative purposes to check if each n-gram in an input string is likely to be valid in the language [8][10][3].

2.3 MOST KNOWN SPELL-CHECKERS FOR ARABIC

2.3.1 Zerrouki-Balla Spell-Checker. Zerrouki and Balla developed a spell-checker for Arabic language [12]. They maintained Aspell and Hunspell spell-checkers. They tried to add infixes and support circumfixes with ignoring diacritics to the open source of Aspell and Hunspell [12].

2.3.2 Shaalan Spell-Checker. Shaalan et. al [6] attempt to develop a spell-checker program to recognize common spelling errors for standard Arabic and Egyptian dialects. They have implemented their checker using SICStus Prolog language. The interface is built using Microsoft Visual Basic. The first step in this checker is to detect an error. According to Shaalan et. al [6] there are two possibilities for causing errors:

(1) The misspelled word is an isolated word (Non-word).
(2) The misspelled word is a valid word. (As writing [نال] instead of [مال]).

2.3.3 Haddad-Yaseen Spell Checker. Haddad and Yaseen [2] proposed a hybrid model for spell-checking and correcting of Arabic words, based on semi-isolated word recognition. In their work, the error of Arabic word is classified into three types; typographic, cognitive and phonetic errors. Each one of these error types is categorized into single error or multi error.

3. DEFINITIONS

In this section some of concepts are defined to clarify the methodology of this research.

Matrix Definition

The matrix is a set of columns *N* and rows *M* defined by *N*×*M*. A rectangular collection of cells is organized by the intersection of each *N* and *M*. Each cell contains exactly one value. The cells in any matrix are addressed by a pairs of letter or integer number depending on how and where they will be used. The first (left) index in the pair specifies the column; while the second (right) index specifies the row.

Tokenizing Process

The main aim of this process is to split the entered text to a set of tokens (words).

Cleaning Process

The cleaning process is executed after tokenizing stage, it concerned with cleaning the input text from not useful data such as punctuation marks and numbers.

4. THE PROPOSED ARABIC SPELL-CHECKER

4.1 ARABIC CORPUS

The corpus which is used in this paper is adapted from Muaidi Phd thesis[1]. This corpus (hereafter refer to as Muaidi Corpus) is implemented and compiled by the first author at De Montfort University in UK [4]. The corpus consists of 101,987 word-types. These words are used to training and testing the proposed spell-checker.

4.2 BUILDING THE MATRICES

In literature, we found that Muaidi [4] has determined the longest valid word in Arabic language has 12 letters. For this reason an eleven matrices (number of matrices = longest word - 1) are built, the dimension of each matrix is 28 × 28 [2]. The first matrix *(M_1)* for the combination of the first and the second letters in a word. The second matrix *(M_2)* for the combination of the second and the third letters in a word. While the eleventh matrix *(M_{11})* for the combination of the eleventh and the twelfth letters in a word. All the matrices are initialized by zeros.

To store a given Arabic word in the corresponding matrices, the 2-Gram set **(S)** for the word is extracted. Each item in **(S)** consists of two letters and the item will assign the value 1 or 2 according to the following rules:

- If the item is the last one in **(S)**, then the value 2 is stored in the corresponding matrix and it indicates that this word is ended by these two letters.
- If the item is not the last one in **(S)**, then the value 1 is stored in the corresponding matrix and it indicates that this word is not ended by these two letters.

The following example clearly shows how to build the matrices from the following tiny corpus:

EXAMPLE 1.

Assume a tiny corpus has 22 Arabic words. The distribution of these words according to their lengths as follows:

- Length=2 : أب، أخ، أم، ثم ، من
- Length=3 : تاج، باب، أخت، متر، حجر
- Length=4 : ناجح ، جهاد تامر، أحمد، أحمر
- Length=5 : أحمدي، ترتيب، حاسوب، أختين
- Length=6 : ياسمين، ناجحون، راسبون

The longest word in the above tiny corpus has 6 letters; so 5 matrices should be generated to represent all the words in the corpus.

- For the word:[جهاد]
  - The 2-Gram set is S = {جهـ، ها، اد}.
  - M_1[ج][هـ]=1, M_2[هـ][ا]=1, M_3[ا][د]=2.
- For the word:[ناجح]
  - The 2-Gram set is S = {نا، اج، جح}.
  - M_1[ن][ا]=1, M_2[ا][ج]=1, M_3[ج][ح]=2.

As example, Figure 1 shows a snapshot for the matrix M_1 for the 1st and the 2nd Letters.

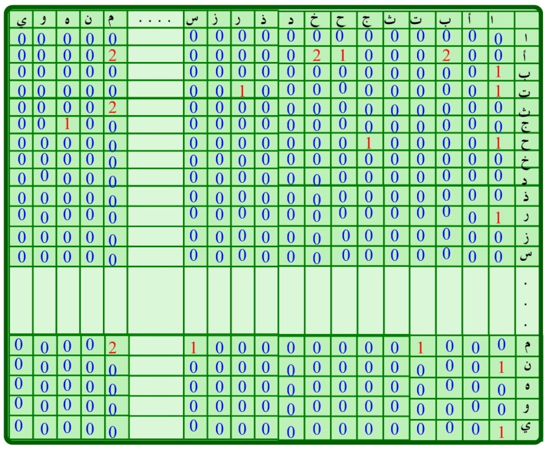
4.3 THE ALGORITHM FOR THE PROPOSED SPELL-CHECKER

The main menu of the proposed spell-checker based on the matrix approach is shown in Figure 2. This interface offers sets of buttons for executing the different implemented algorithms.

The mechanism of the proposed spell-checking of a given Arabic text is as follows:

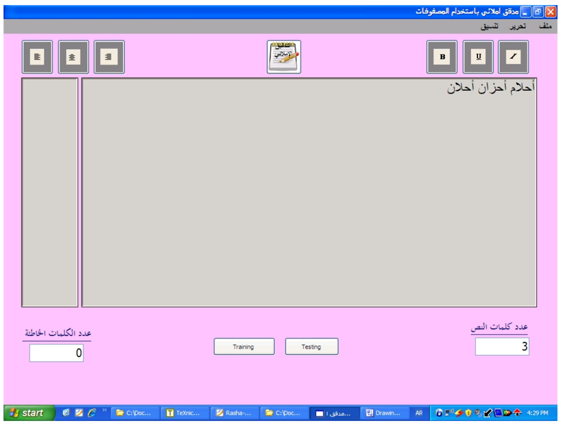
- The user entered the tested text.
- The spell-checker starts the tokenizing processing.
- The cleaning process is executed after tokenizing stage.
- Matrix method deals with each word within the text separately and extracts the 2-Gram set for it. Then it examines the value for each item in the 2-Gram set. When the corresponding value for the item is zero then the spell-checker will consider the tested word as wrong word and colors it by red and then starts checking the next word.

[1]Dr. Hasan Muaidi, AL-Balqa' Applied University.
[2]The Arabic language has 28 letters



ي	و	ه	ن	م	....	س	ز	ر	ذ	د	خ	ح	ج	ث	ت	ب	أ	ا	
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	ا
0	0	0	0	2		0	0	0	0	0	2	1	0	0	0	2	0	0	أ
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	ب
0	0	0	0	0		0	0	1	0	0	0	0	0	0	0	0	0	1	ت
0	0	0	0	2		0	0	0	0	0	0	0	0	0	0	0	0	0	ث
0	0	1	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	ج
0	0	0	0	0		0	0	0	0	0	0	0	1	0	0	0	0	1	ح
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	خ
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	د
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	ذ
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	ر
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	ز
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	س
																			. . .
0	0	0	0	2		1	0	0	0	0	0	0	0	0	1	0	0	0	م
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	ن
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	ه
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	و
0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	ي

Fig. 1. Matrix M_1 for the 1^{st} and the 2^{nd} Letters



احلام أحزان أحلان

عدد كلمات النص 3

عدد الكلمات الخاطئة 0

Fig. 2. Spell-Checker by Matrix Approach

The following algorithm clears the above procedure.

Algorithm 1 Matrix Approach Algorithm

```
1: procedure CHECKMATRIX($Word, Matrix(i)$)                ▷
2:     $Flag \leftarrow True$
3:     $i \leftarrow 1$
4:     while $Flag = True\ and\ i \leq n-1$ do                ▷
5:         if $M_i[L_i, L_{i+1}] = 0$ then
6:             $Flag \leftarrow Fail$
7:             return $Flag$
8:         else
9:             $i \leftarrow i+1$
10:        end if
11:    end while
12:    if $M_{n-1}[L_{n-1}, L_n] = 2$ then
13:        return $Flag$
14:    else
15:        return $Fail$
16:    end if
17: end procedure
```

The drawback for this approach is that it may be failed to detect the misspelling word as shown in the following example:

EXAMPLE 2.

Assume the following two words[أحلام] and [أحزان] are already stored in their corresponding matrices. The indexing values for each two letters of them are as follows:

(1) M_1[ح][أ]=1

(2) M_2[ح][ل]=1

(3) M_3[ل][ا]=1

(4) M_4[ا][م]=2

(5) M_4[ا][ن]=2

- Now, assume the tested word is [أحلان], which is incorrect in the Arabic language.
- The 2-Gram set for this word is S= {أح ، حل ، لا ، ان}.

(1) M_1[ح][أ]=1

(2) M_2[ح][ل]=1

(3) M_3[ل][ا]=1

(4) M_4[ا][ن]=2

- Since no index has zero value, the spell-checker is consider this word as a correct one. The previous Figure 2 shows a snapshot for the above example.

5. EXPERIMENTAL RESULTS

Two stages are presented to test the results and measure the proposed spell-checker performance.

5.1 The Training Stage

As mentioned, the size of Muaidi corpus consists of 101,987 words. These words are considered as a dataset to train and to test the performance of the proposed spell-checker. This dataset is divided into two unequal parts, bulk part (70%) which is used as a training dataset for the training stage and the remaining part (30%) is taken as a testing dataset for the testing stage.

The training dataset consists of 71,390 Arabic words. While the testing dataset consists of 30,597 Arabic words. The evaluation process is done on an Intel core 2 dual processor with a speed 1.80 GHz. The RAM capacity is 1,016 GB and the operating system is WINDOWS XP.

The methodology of evaluation the current research study is organized as follows:

- The training dataset is used to build the matrices.
- To assure the ability of learning, the training dataset is used to test the matrix approach approach.
- Testing dataset (which is unseen data) is used to check the performance of the spell-checker.
- The words which are considered as incorrect words from the previous step are reentered to the system.
- The performance of the spell-checker is recomputed again.
- The implemented evaluation methodology for the proposed spell-checker is based on the ability of it to successfully spelling the correct Arabic words.

The proposed matrix approach is correctly spelled 99.79% of the words. Table 1 summarizes the evaluation of the results in the training dataset. While Figure 3 demonstrates these results in a column format chart. **The Testing Stage**
In testing stage a hidden dataset (testing dataset) is used to indicate the accuracy of the proposed spell-checker system. As mentioned before the testing dataset consists of 30,597 Arabic words. To test the performance of the proposed system, the accuracy is calculated using the success rate measure S_R. This

Table 1. The Evaluation Results in the Training Dataset

Number of tested words	71,390
Number of words that are spelled correctly	71,240
Number of colored words	150
Success rate	99.79%

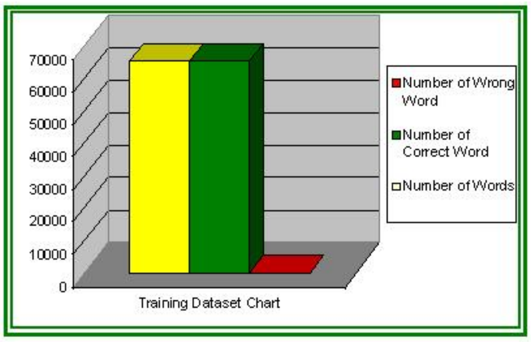


Fig. 3. The Evaluation of the Training Dataset

measure compatible for the proposed spell-checker with checking the error words and colored them. Success rate measure is calculated as shown in Equation 1.

$$S_R = \frac{C_W}{N} 100\% \qquad (1)$$

Where:

- S_R = The success rate.
- C_W = The of words that are spelled correctly.
- N = The size of the testing dataset.

The experiment is performed on the proposed spell-checker system using the testing dataset and the success rate is obtained 40.40%. Table 2 summarize the evaluation of the results in the testing dataset. Figure 4 illustrate these results in a column format.

Table 2. The Evaluation of Results in the Testing Dataset

Number of tested words	30,597
Number of words that are spelled correctly	12,362
Number of colored words	18,235
Success rate	40.40%

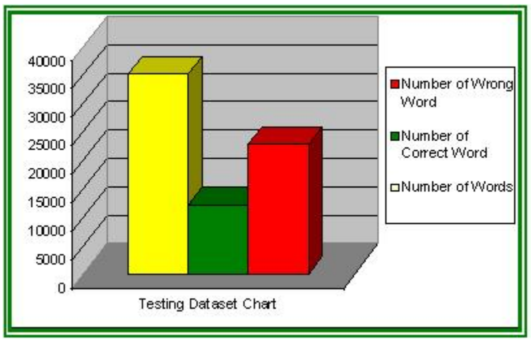


Fig. 4. The Evaluation of Results in Testing Dataset

5.2 Causes of the Errors

The difference for accuracy between the two stages (training and testing) is back to the number of words in the each dataset. When the size of the dataset is increased the accuracy will increased spontaneously; and the system has the ability to recognize a great number of words and automatically the error rate is decreased.

5.3 Discussion of the results

The proposed spell-checker system accuracy reached 99.7% using the training dataset. While the accuracy of the proposed system reached to 40.4% using the testing dataset. This difference back to the variant between build data and test data; which means that the testing data is unseen from the system, so if we use the testing dataset in rebuilding the system and retest them again using this dataset, the accuracy will jump tremendously. It jumps from 40.4% to 97.12%. Table 3 clear this while Figure 5 illustrate these results in a column format.

Table 3. The Evaluation of Results in Testing Dataset

Number of tested words	30,597
Number of words that are spelled correctly	29,715
Number of colored words	879
Success rate	97.12%

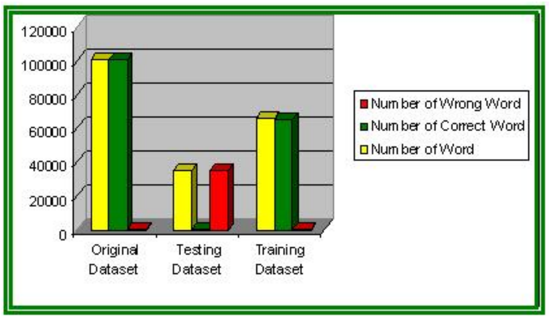


Fig. 5. The Difference Between the Results Before and After Adding Unseen Data

The above discussion is clarified that the accuracy is mainly depend on the number of words in the corpus, so to increase the accuracy of the proposed spell-checker the number of words should be increased. The overall accuracy of reached to 98.99%. Table 4 summarize all these results.

Table 4. The Overall Evaluation of the Results

	Training Data	Testing Before Rebuild	Testing After Rebuild	All Data
Number of tested words	71,390	30,597	30,597	101,987
Number of words that are spelled correctly	71,240	12,362	29,715	100,955
Number of colored words	150	18,235	879	1029
Success rate	99.79%	40.40%	97.12%	98.99%

6. CONCLUSION

This paper addresses the problem of spell-checking for Arabic language. The proposed spell-checker which is presented in this paper is based on N-Grams scores. For this purpose, eleven matrices are built to present the combination between the Arabic letters word. The proposed spell-checker is trained and tested using a dataset extracted from Muaidi Corpus. The accuracy is calculated of the proposed spell-checker. The overall accuracy is reached to 98.99%. The results of these experiments are promising, and represent a good starting point for future research Some of problems faced the spell-checker like it may recognize many of incorrect words and consider them as correct.

7. REFERENCES

[1] Feldman A. Computational linguistics: Models, resources, applications. *Computational Linguistics*, 32(3):443–444, 2006.

[2] Haddad B. and Yaseen M. Detection and correction of non-words in arabic: A hybrid approach. *International Journal of Computer Processing of Oriental Languages*, 30, 2007.

[3] P. Brown, P. deSouza, R. Mercer, V. Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[4] Muaidi H. *Extraction Of Arabic Word Roots: An Approach Based on Computational Model and Multi-Backpropagation Neural Networks.* PhD thesis, De Montfort University - UK, 2008.

[5] Satori H., Harti M., and Chenfour N. Arabic speech recognition system using cmu-sphinx4. *CoRR 0704.2201*, 2007.

[6] Shaalan K., Allam A., and Gomah A. Towards automatic spell checking for arabic. In *Language Engineering*, 2003.

[7] Kukich Karen. Technique for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24:377–439, 1992.

[8] Karttunen. Applications of finite-state transducers in natural language processing. In *CIAA: International Conference on Implementation and Application of Automata, LNCS*, 2000.

[9] Kabbani M. The arabic spell-checker dictionary from ayaspell project. Technical report, Prix special des troisiemes rencontres africaines du Logiciel Libre, 2008.

[10] Suleiman H. Mustafa and Qasem A. Al-Radaideh. Using n-grams for arabic text searching. *JASIST*, 55(11):1002–1007, 2004.

[11] Alqrainy S., Ayesh A., and Muaidi H. Automated tagging system and tagset design for arabic text. *International Journal of Computational Linguistics Research*, 1:55–62, 2010.

[12] Zerrouki T. and Balla A. Implementation of infixes and circumfixes in the spellcheckers. In *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009.