International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 12, No. 1, pp. 127 – 133, 2015

ADAPTATING THE LEVENSHTEIN DISTANCE TO CONTEXTUAL SPELLING CORRECTION

AOURAGH SI LHOUSSAIN

University Sidi Mohamed Ben Abdellah,
Faculty of Sciences, Dhar El Mahraz,
BP :1796, Atlas, Fez, Morocco
aouragh@hotmail.com

GUEDDAH HICHAM

University Mohammed V of Rabat Telecom and,
Embedded Systems, Team SIME Lab,
ENSIAS -Rabat, Morocco.
h.gueddah@um5s.net.ma

YOUSFI ABDELLAH

University Mohammed V of Rabat Faculty of Juridical,
Economic and Social Sciences Souissi, Rabat, Morocco.
yousfi240ma@yahoo.fr

In the last few years, computing environments for human learning have rapidly evolved due to the development of information and communication technologies. However, the use of information technology in automatic correction of spelling errors has become increasingly essential. In this context, we have developed a system for correcting spelling errors in the Arabic language based on language models and Levenshtein algorithm. The metric distance returned by the Levenshtein algorithm is often the same for multiple solutions in correcting a wrong word. To overcome this limitation we have added a weighting based on language models. This combination has helped us to screen and refine the results obtained in advance by the Levenshtein algorithm, and applied to the errors of Arabic words. The results are encouraging and demonstrate the value of this approach.

Keywords: Spelling errors; Contextual correction; Arabic language; Distance; Measurement; Probability; Language model.

1. Introduction

The automatic correction of spelling errors is one of the most important areas in the field of the Natural Language Processing (N.L.P). Research in this field began in the sixties of the previous century [Kukich (1992)]. The correction of spelling errors consists of making comparisons between a wrong word and the words of a lexicon of a given language, and to propose a list of words closest to the wrong word based on the similarity and the distance between words. In the field of automatic correction of spelling errors out of context, several studies have been made, among which we quote:

- Damerau defined a kind of modeling spelling error. He has fixed four types of errors [Damerau (1964)] : insertion, deletion, permutation, replacement.

- Relying on the works of Damerau, Levenshtein [Levenshtein (1966)] considered three editing operations (insertion, deletion, permutation), and defined his method as edit distance that compares two words while calculating the number of editing operations subjected on a word to turn it into another. This distance is also called Levenshtein distance.
- Oflazer has proposed a new approach called "recognition error tolerance"[*] that is based on the use of a dictionary represented as finite state automata. According to this approach, the correction of an erroneous word consits in browsing through a finite state automaton by calculating for each transition a distance called cut-off-edit distance [Oflazer (1996)].
- Relying on the works of Oflazer, Savary [Savary et al. (2000)] has adopted a variant of this method with major modifications because it excludes the calculation of cut-off-edit distance.
- Pollock and Zamora [Pollock et al. (1984)], have defined another way to model the spelling errors, and this is by calculating what they called : Alpha-code. An alternative method of alpha-code was proposed by Ndiay and Faltin [Ndiaye et al. (2004)].

The major drawback of spelling correction systems out of context is that these systems return multiple solutions in cases having the same importance (even Leven-shtein distance for example). To remedy this problem Gueddah and Yousfi [Gueddah et al. (2013)] [Gueddah et al. (2012)] introduced weights of the editing operations errors so as to improve the scheduling of solutions returned by the Levenshtein distance. In this context, there are other works that have been made to improve the process of correcting errors (real-words errors for example), but this time taking into account the context of the word within the sentence based on a voluminous data base, for example: that provided by Google : Google Web 1T 5-gram[†] [Bassil et al. (2012)] [Carlson et al. (2007)]. In this article, we discuss the correction of spelling errors within their contexts and we use n-gram language models to refine the suggested solutions and to better scheduling their positions.

2. N-gram Language Models

The importance of language models is quite clear; they are used in many fields of natural language processing such as the recognition of continuous speech, automatic translation,.... The main goal in these applications is weighting some solutions relative to others.

A model of n-gram language expresses the fact that the probability of appearance of a word after a sequence of words may be given only on the basis of n-1 the last words [Hirst (2008)] . This model is verified :

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

[*] Error-Tolerent Finite-State recognition with Applications to Morphological Analysis and Spelling Correction
[†] http ://catalog.ldc.upenn.edu/LDC2009T25

In practice, the value of n is not more than the ordre of 3.

- If n = 1, the model is called uni-gram model. This type of model does not take into account any history of the word.
- If n = 2, the model is called bi-gram model. This type of such models takes into account the previous word :

$$\Pr(w_i / w_1,..., w_{i-1}) = \Pr(w_i / w_{i-1})$$

- If n = 3, the model is called tri-gram model ([Aouragh et al. (2006)], [Woszc¬zyna (1998)], [Gauvain et al. (2000)], [Bacchiani et al. (2001)]). This type of model takes into consideration the two previous words :

$$\Pr(w_i / w_1,..., w_{i-1}) = \Pr(w_i / w_{i-1}, w_{i-2})$$

For the construction of the n-gram model, we conduct a learning process on a corpus of texts which must encompass all possible successions of words belonging to the vocabulary used. This construction consists in estimating all the probabilities mentioned above. The probability of the appearance of a word, after a given word sequence, is estimated as follows :

$$\Pr(w_i / h_n(w_i)) = O(h_n(w_i), w_i) / O(h_n(w_i))$$

where : $h_n(w_i) = w_{i-1}...w_{i-n+1}$ is the history of the word w_i .
$O(h_n(w_i))$ is the number of occurrences of the context $h_n(w_i)$, and $O(h_n(w_i), w_i)$ that of the word w_i after the same context $h_n(w_i)$.

- Note

The n-gram model is completely defined by the set of parameters noted $\Theta = (A, P)$, with:

- $A = \{a_1,..., a_N\}$ the set of probabilities of initial words.
- $P = (p_{ij})_{1 \le i, j \le N}$ the matrix of probabilities of occurrence of a word w_i after a word w_j .

In general, three estimation methods of model parameters can be used :

- Estimation of Maximum likelihood,
- Estimation by Maximum a posteriori,
- Estimation by maximum of mutual information,

In our study, we have used the maximum of likelihood in order to estimate the parameters of a bi-grams model.

3. Levenshtein algorithm

The Metric method introduced by Levenshtein [Levenshtein (1966)] measures the similarity between two words by calculating an edit distance. The edit distance is defined as the minimum number of basic editing operations needed to transform a wrong word to a dictionary word. Thus, to correct a wrong word, one retains a set of solutions requiring fewer possible editing operations.

The procedure for calculating the Levenshtein distance between two strings $X = x_1 x_2 ... x_m$ of length m and $Y = y_1 y_2 ... y_m$ of length n, is to calculate step by step in a matrix of order $m \times n$ edit distance between different sub-strings of X and Y. The corresponding values are stored in the matrix up to the box (m, n) which expresses the minimum distance between X and Y .

The calculation of the cell (i, j), which corresponds to the edit distance between sub-strings of $X_i^1 = x_1 x_2 ... x_i$ and $Y_j^1 = y_1 y_2 ... y_j$, is given by the following recursive relationship :

$$D(i, j) = Minimum\{D(i-1, j)+1; D(i, j-1)+1; D(i-1, j-1)+\cos t\}$$

With $\cos t = \begin{cases} 0 & if\ x_{i-1} = y_{j-1} \\ 1 & otherwise \end{cases}$

Thus, the following boots: $D(i, \phi) = i$ and $D(\phi, j) = j$, where ϕ is the empty string.
The limitation of such a spelling correction system using the edit distance is that it does not provide a good scheduling suggested for all candidates with the same edit distance solutions.

Example:-

For the wrong word "الحعيم", the algorithm returns the Levenshtein distance for a total closest to the wrong word in the table below words and distance 2 for the solution "الرحيم". But taking into account the context of words within the sentence, the word "الرحيم" is the most likely solution. To overcome this limitation of the Levenshtein distance, we propose to introduce the probabilities of successive words (using language models) in the Levenshtein distance to better schedule proposed by the distance corrections.

Erroneous word	Solutions	Edit distance
بسم الله الرحمان الحعيم	الحكيم	1
	الحليم	1
	النعيم	1
	الرحيم	2

4. Introducing bi-grams language models in levenshtein algorithm

Either w_{err} werr a wrong word in a given context. The proposed approach in this paper, we are only interested in the preceding word of the wrong word werr, noted as v.
By applying the Levenshtein algorithm, we obtain a set of solutions $S = \{w_1, w_2, ..., w_p\}$ and Levenshtein distances between werr and wi of S are given by :

$$D_L(w_{err}, w_i) = d_i$$

To refine the scheduling of these solutions, we introduce the bi-grams model language in the Levenshtein distance, that is characterized in this case by the probability $\Pr(w_i / v)$.

The weighted Levenshtein distance is defined by this probability by :

$$D_{LP}(w_{err}, w_i) = \begin{cases} \dfrac{d_i}{\Pr\left(w_i / v\right)} & if\ \Pr\left(w_i / v\right) \neq 0 \\ \min\limits_{w,u \in Y} \dfrac{d_i}{\Pr\left(w / u\right)} & otherwise \end{cases}$$

Y is the vocabulary of all the words of our spelling correction system.
The best corrections noted wc are given by the following equation :

$$w_c = \min_{w_i \in S} D_{LP}(w_{err}, w_i)$$

This new measure will give new scheduling solutions.

In the next section, we present the results of a comparative study between the two distances for the wrong word shown in the example above.

Erroneous word	Solutions	Edit distance	Adopted Measurement
بسم الله الرحمان الحعيم	الحكيم	1	D_{LP} (الرحيم, الحكيم) $= \frac{1}{0,1}$
	الحليم	1	D_{LP} (الرحيم, الحليم) $= \frac{1}{0,1}$
	النعيم	1	D_{LP} (الرحيم, النعيم) $= \frac{1}{0,01}$
	الرحيم	2	D_{LP} (الرحيم, الرحيم) $= \frac{2}{0,7}$

By introducing the bi-gram probabilities, we note that the الرحيم solution takes the first position according to the number of occurrences of that word before الرحمان in the test corpus.

5. Tests and Results

To evaluate our proposed new contextual correction we misspelled a set of words with different degrees: 1 and 2 (i.e. the edit distance of 1 or 2), and to demonstrate our method, we added a group of related words to this set (المملكة المغربية، المحيط الهندي، مجلس النواب)
The first step of our test is to calculate the Levenshtein distance between the misspelled words and the words of our correction system. The second step is to calculate the bi-gram language model. As the third step, we applied the formula (1) of Section 4.
The results are summarized in the table below. We significantly expressed our satisfaction with the combination of the two concepts of classical spelling correction and the bi-gram language model. We notice that our proposal allows returning the

	Edit distance	Adopted distance
1st position	11,41%	11,41%
2nd position	18,44%	11,41%
3rd position	20,19%	...
4th position	9,66%	...

5th position	12,29%	...
6th position	8,79%	...
7th position	5,29%	...
8th position	1,80%	...
9th position	3,60%	...
10th position	7,90%	...

solutions suggested in first place depending on their context of appearance with a rate of 95.61% against the Levenshtein distance that reaches only 11.41% of the solutions in the first position. As shown in the rest, 4.39% in the second position results from the fact that there are solutions with the same probability and the bi-grams same Levenshtein distance. We would also like to mention that our proposal enables to correct effectively and in first place with a rate of 97.5% of exact solutions for related words, as it returns approximately a rate of 94.59% in the first order solutions desired and selected for other misspelled words.

6. Conclusion

In this article, we have tried to introduce bi-grams language models in the Levenshtein metric correction method. From the results obtained, we see clearly that this combination helps improve satisfactorily scheduling solutions returned by our method.
The major drawback in our approach lies in the fact that our test corpus size is limited. In the next contribution, we will try to extend our method on a very large corpus and to get better estimates for bi-gram language models to enhance and increase the effectiveness of our correction approach.

References

Aouragh, L., Allal J., Yousfi, A. (2006) *A Stochastic Language Model for Automatic Generation of Arabic Sentences*, Georgian Electronic Scientic Journal : Computer Science and Telecommunications No.3(10).

Bacchiani, M., Hirschberg J., Rosenberg A., Whittaker S., Hindle D., Isenhour P., Jones M., Stark L., Zamchick G., (2001) *SCANMail : Audio Navigation in the Voicemail Domain*, In Human Language Technology conference, Sand Diego California (USA).

Bassil, Y. Alwani, (2012) *Context-sensitive Spelling correction Using Google 1T 5-Gram Information*, Vol.5, N°3, Computer and Information Science, pp :37-48.

Carlson, A., Fette, I. (2007) *Memory-Based Contexte-Sensitive Spelling correction at Web Scale*, IEEE Computer Society. ICMLA, pp : 166-171.

Damerau, F.J., (1964) *A technique for computer detection and correction of spelling errors*, Communications of the Association for Computing Machinery, 7(3):171–176,

Gauvain, J.L., Lamel L., Adda G., (2000) *Transcribing broadcast news for audio and video indexing*, CACM, Vol 43, No 2, pp : 64-70.

Gueddah, H., Yousfi, A. (2013) *Impact de la Proximité et de la Similarité inter-caractères Arabe sur la Correction Orthographique*, Proceeding of the 8th International Conference on Intelligent Systems : Theories and Applications, pp : 244-246, SITA, EMI.

Gueddah, H., Yousfi, A.,Belkasmi, M. (2012) *Introduction of the weight edition errors in the Levenshtein distance*, International Journal of Advanced Research in Articial Intelligence, Vol.1 Issue 5, pp : 30-32,Aout-2012.

Hirst, G. (2008) *An Evaluation of the Contextual Spelling Checker of Microsoft Office Word 2007*, Department of Computer Science University of Toronto Toronto, Canada .

Kukich, K., (1992) *Techniques for Automatically Correcting Words in Text*, ACM Computing Surveys, Volume 24, No.4, pp, 377-439, December.

Levenshtein, V., (1966) Binary codes capable of correcting deletions, insertions and reversals, Soviet Physics Doklady, 10(8): 707-710.

Ndiaye, M., Faltin, A. V. (2004) *Correcteur Orthographique Adapté à Apprentissage du Français*, Revue Bulag n°29, pp, 117-134.

Oflazer, K., (1996) Error-tolerant Finite-state Recognition with Applications to Morphological Analysis and Spelling Correction, Computational Linguistics archive, Volume 22 Issue 1,pp,73-89, March.

Pollock, J. and Zamora A., (1984) *Automatic Spelling Correction in Scientific and Scholarly Text*, Communications of the ACM, Vol. 27, No. 4, pp. 358-368.

Savary, A., (2000) Recensement *et description des mots composés méthodes et applications*, 2000, version 1, pp, 149-158.

Walker, P. R. and Mishra, O. P. (1999). *Framework for CBR based Health Management System*. Ann. of Computing., 41 : 27–42.

Woszczyna, M., (1998) *Fast speaker independant large vocabulary continuous speech recognition*, thèse de l'université de Karlsruhe (Allemagne), 156 pages,