Two Bigrams Based Language Model for Auto Correction of Arabic OCR Errors

[1]Imad Q. Habeeb, [2]Shahrul A.M. Yusof, [3]Faudziah B. Ahmad
[1, First Author] *Iraqi Commission for Computers and Informatics, emadkassam@yahoo.com*
[*2, Corresponding Author] *Universiti Utara Malaysia, shahrulazmi@uum.edu.my*
[3] *Universiti Utara Malaysia, fudz@uum.edu.my*

Abstract

In Optical character recognition (OCR), the characteristics of Arabic text cause more errors than in English text. In this paper, a two bi-grams based language model that uses Wikipedia's database is presented. The method can perform auto detection and correction of non-word errors in Arabic OCR text, and auto detection of real word errors. The method consists of two parts: extracting the context information from Wikipedia's database, and implement the auto detection and correction of incorrect words. This method can be applied to any language with little modifications. The experimental results show successful extraction of context information from Wikipedia's articles. Furthermore, it also shows that using this method can reduce the error rate of Arabic OCR text.

Keywords: *Auto correction, Arabic OCR errors, Bigram language model, OCR post-processing.*

1. Introduction

An Arabic text over the years is difficult when treated by the process of optical character recognition (OCR), because the letters are connected and their shape is changed depending on their position in the words [1-3]. These characteristics cause some errors during the optical character recognition process, especially when the texts are worn out or their colors have changed [4, 5]. OCR process usually produces two kinds of errors, non-word errors and real word errors. Non-word error occurs when the word produced from the OCR process does not exist in the lexicon. The real word error occurs when the word produced from the OCR process exists in the lexicon but does not match with the source text [6, 7].

The most widely used technique for finding a list of candidates for word errors is the *Levenshtein* distance algorithm [8, 9]. The *Levenshtein* distance algorithm is used to calculate the difference between two strings where every insertion, deletion, or substitution of a single character is considered as a single edit [10]. For example, "*foed*" is a non-word in English, because it does not in the lexicon. It requires one substitution to become "*food*" that is considered a correct word in English [6]. The candidate words obtained from using the edit distance technique, for the previous word "*foed*" are: "*feed*" , "*food*", "*ford*", and "*foe*". All these words have a single edit distance. Now the question is: How to choose the most suitable word? What is the best automated method?

The auto correction process is made up of three sequential steps. These are: (1) identifying the wrong words; (2) generate candidate words from the language resources; and (3) ranking the candidate words according to their importance in the sentences [11]. When the auto correction is implemented, the incorrect word is automatically replaced by the first word in an ordered candidate list [9]. The choice of the appropriate technique is very important because it may replace the wrong word by other found in the lexicon, but it is unsuitable for the sentence, resulting the desired goal of correction is to be unachieved [6].

In this study, a method that combines two bi-grams based language model with the *Levenshtein* distance algorithm [10] will be developed for auto detection and correction of errors in Arabic OCR texts. Wikipedia data is used for the study. The method can be applied to other languages with minimal modifications as the study uses freely available Wikipedia that has more than a hundred languages and continues to grow [12, 13].

2. Related Work of OCR Error Correction

There are three major approaches to OCR post-processing error correction: (1) proofreading-based correction, (2) Lexicon-based correction, and (3) context-based correction[14]. Proofreading-based correction requires human to read and rewrite text produced from the OCR process. This is inefficient as it is time-consuming, especially when the number of words is in thousands [15].

The lexicon-based correction is used to identify the non-words errors. This error happens when the word resulted from the OCR does exist in the lexicon [6]. The drawback of this approach is that it cannot rely on auto correction because candidate words are arranged without any regard for the context within the sentence around the incorrect word. Furthermore, it cannot detect the real word errors [14]. Lastly, the context-based takes into account the words surrounding the wrong word. It is more complex than the previous techniques and can detect real word errors [16].

In correcting OCR errors, several methods have been proposed include: using the similarity in shape among characters in words[11], using grammar rules [17], word count, i.e., the number of occurrences of the word in a large corpus or in the web is used in selecting the right word [18], a combination of a dictionary to correct non-word errors, and a model to correct real word errors [19], a matrix of sequence and count of characters for all words of resulted text of OCR, where the highest count in the matrix is then replaced with the non-word [20]. Others are: confusion sets that use common errors in words[21], Google's spelling and suggestions [14], using of multiple systems and select the best output [22], and probability based language models [8, 23, 24].

A language model based on web corpora has been used in recent researches for correcting OCR errors, it gathers context information of sentences that are stored in the web corpora [24, 25], the context information gathered from texts is used to select the best candidate for wrong-word errors [18].

Existing language models, which are using one n-gram to predict the next word [26] will not use in this study, instead of that, a language model that uses two bi-grams is proposed to improve the accuracy in the selection from the candidate words. In addition to that, the candidate's list will not be generated from millions of words from the lexicon, but it will be generated from a specific set of words to improve processing time. The next section elaborates the method.

2. Proposed Method

The method consists of two stages. First, explains how to design and fill of the language model based on Wikipedia’s database. Second stage explains how to detect and correct of Arabic OCR errors.

2.1. Designing and filling of a language model

In order to use a language model for Arabic language, it needs a free large web corpus. The study used a Wikipedia database, which is available freely and can be downloaded as one file in the xml format[1]. The database has been chosen for several reasons : (1) It is the largest free source of Arabic web corpus, (2) It contains more than 232,917 Arabic articles in different categories, and (3) There is a rapid growth in Arabic articles and (4) the database is constantly updated[27, 28].

English language and many other languages that use Latin characters, do not suffer from any problem to extract of text from Wikipedia’s database because there are many programs that can be used to extract text directly from a Wikipedia dump file. However, these programs are designed to eliminate any non-Latin letter[12], and therefore, cannot be used to extract Arabic texts. The use of an indirect way in the extraction of Arabic texts by downloading pages and articles from Wikipedia’s site and then extracting text from them takes a long time[29]. For these reasons; a new method is developed that can be used to extract only Arabic text from a Wikipedia database as shown in Fig. 1. In this method, the reading of Wikipedia xml file will be in parts because it is large.

[1] http://dumps.wikimedia.org/arwiki/

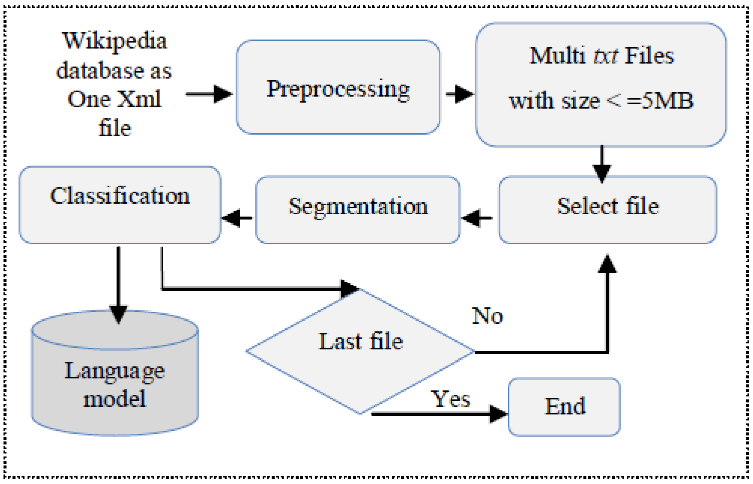


Figure 1. Filling of language model database.

In preprocessing stage, address of images, navigation, and layout are ignored and only text from Wikipedia xml file is extracted. Then each text with size less or equal to 5MB is stored as a separate plain text file. The text is split into small pieces for two reasons: First, there are thousands of operations that will take place on each file and these processes consume time. Therefore, when the file is a large and the process has an error, the work will be repeated from scratch, and this takes additional time. However, in case of file is small, the program will have to re-process only the specific file and thus, takes a shorter time. The second reason is, a large file is not recommended to put in the main memory at once because this may cause the computer to be hanged or stopped. All files are placed in a specific folder by the program automatically. In the segmentation stage, the program extracts only text "T" from input file and then split the text "T" into an array of words "W", so that it passed to the classification stage.

In the classification stage, every word W[i] (W[i] € lexicon) will have variables stored within the language model database. These variables are: (1) frequency of word W[i] in the web corpus, (2) frequency of the word W[i-1] when come before word W[i] in the web corpus, (3) frequency of the word W[i+1] when come after word W[i] in the web corpus, (4) all words that came after word W[i], i.e., any word W[i+1] be on the right of word W[i] and (5) all words that came before word W[i], i.e., any word W[i-1] on the left of word W[i]. Variables one, two, and three are stored because they help in calculating of probability of the two bi-grams model. In addition to that, variables four and five are stored because they help in calculating *Levenshtein* distance algorithm for specific set of words.

To illustrate more why these variables are stored, let take variables one, two, and three, the probability of one bigram language model for any candidate W[i] to come instead of a wrong word can be calculated from (1)[30]:

$$P(\mathrm{W[i]} \mid \mathrm{W[i-1]}) = \frac{f(\mathrm{W[i-1]}\ \ \mathrm{W[i]})}{f(\mathrm{W[i-1]})} \qquad (1)$$

Where P(W[i] | W[i-1] is the probability of candidate W[i] to come after the word W[i-1], while f (W[i-1] W[i]) is frequency of candidate W[i] to come after a word W[i-1] in corpus, and f (W[i-1]) is frequency of the word W[i-1] in same corpus.

Equation (1) shows that bi-gram language model is used to predict the next word based only on the previous word, and this will ignore the impact of the next word to predict the previous word. Furthermore, in the absence of the valid word W[i-1], before the wrong word, then it cannot be used (1) to predict the best candidate word for the incorrect word. Based on these, in this study, two bi-grams are used in one language model so that the selection of the appropriate word from the candidate's list depends on the probability offered by the two. On the other hand, if one of them has the probability of zero, then it can rely upon the second and vice versa.

Now the probability of two bi-grams model for word W[i] to come after the word W[i-1] and before the word W[i+1] is:

$$P(\mathrm{W[i]}) = \left(\frac{f(\mathrm{W[i-1]\ W[i]})}{f(\mathrm{W[i-1]})} + \frac{f(\mathrm{W[i]\ W[i+1]})}{f(\mathrm{W[i+1]})}\right)/2 \qquad (2)$$

Where P(W[i]) is the probability of two bigrams language model for word W[i] to come instead of the wrong word;
while f(W[i] W[i+1]) is frequency of candidate W[i] to come before the word W[i+1] in corpus, and f (W[i+1]) is frequency of the word W[i+1] in same corpus. Equation (2) is performed if: (f (W[i-1] W[i])≠0) and (f (W[i] W[i+1]) ≠0). However, in case of
(f (W[i-1] W[i])=0) and (f (W[i] W[i+1]) ≠0), then the equation will be as follows:

$$P(\mathrm{W[i]}) = \frac{f(\mathrm{W[i]\ W[i+1]})}{f(\mathrm{W[i+1]})} \qquad (3)$$

And if: (f (W[i-1] W[i])≠0) and (f (W[i] W[i+1]) =0), then the equation will be as follows:

$$P(\mathrm{W[i]}) = \frac{f(\mathrm{W[i-1]\ W[i]})}{f(\mathrm{W[i-1]})} \qquad (4)$$

In case of (f (W[i-1] W[i])=0) and (f (W[i]W[i+1]=0), then the choice of the best candidate for incorrect word is based only on least edit distance between them.

At this point, the explanation should justify why two bi-grams are used instead of one and why variables one, two, and three are stored. Now Let return to the variables four and five, which are mentioned in beginning of classification stage and illustrate why they are stored; the generating of candidates list from millions of words from the lexicon is not used in this study directly, for two reasons: (1) It reduces the accuracy because of the huge number of candidates, which may be reaching thousands, and (2) It takes long time to process. Instead of that, the list will be generated by using the *Levenshtein* distance algorithm, but the process is performed by comparing the incorrect word with all words belong to valid word that comes after or before it. As an example, to generate of candidates list for incorrect word W[i], it needs of comparing W[i] with all words that can come after the valid word W[i-1] and with all words that can come before the valid word W[i+1].

To clarify more, let take the following questions: is it possible, the main verb comes after the same verb? Or the word "school" precedes the word "school"? This is impossible. For these reasons, the comparison will not use all the words from the lexicon in generating of a candidate's list, but the comparison will use all the words that could come after or before the valid word, and these words are extracted from the huge amount of data automatically, which is here Wikipedia's database.

Incorrect words in any texts can be added to the database inadvertently because of the automatic way of storing words. For this, the program performs a procedure to delete any word from the database if they do not exist in the lexicon (this study uses a lexicon with 2158342 Arabic words), and all letters, numbers and symbols used in ASCII table for Arabic and English were added and treated like words. This is because any word, symbol, number or letter that is not in the lexicon will be treated as an error by the program that uses this method for correction of Arabic OCR text.

In this study, smoothing method will not be applied in the Language Model, and this will give a probability of zero for cases where the words that are not contained in the lexicon and for unknown words, i.e., the words that are present in the lexicon, but are not found in web corpus. During training of the language model, zero is assigned to their probability, and the choice of the best candidate for incorrect word is based only on least edit distance between them.

2.2. Detection and correction of Arabic OCR errors

The process of detection and correction of incorrect words in Arabic OCR text is shown in Fig. 2. In this process, there are differences in detecting and correction of errors in OCR based on the type of the error, while the non-word errors will be detected and corrected automatically, the real word errors will not be corrected automatically, but will be identified and the task of correcting the errors is passed to the user. This is because non-word errors are easily identified through the lexicon in which any word outside of the lexicon is wrong. On the other hand, real word errors cannot be determined by the lexicon because they already exist in the lexicon, but are determined by the language model which does not guarantee 100% that they are wrong. Real errors are detected and passed the correction tasks to the user to correct. This is better than the user is reading the text word by word to identify real word errors, because this process is a tiring if mistakes are in thousands.

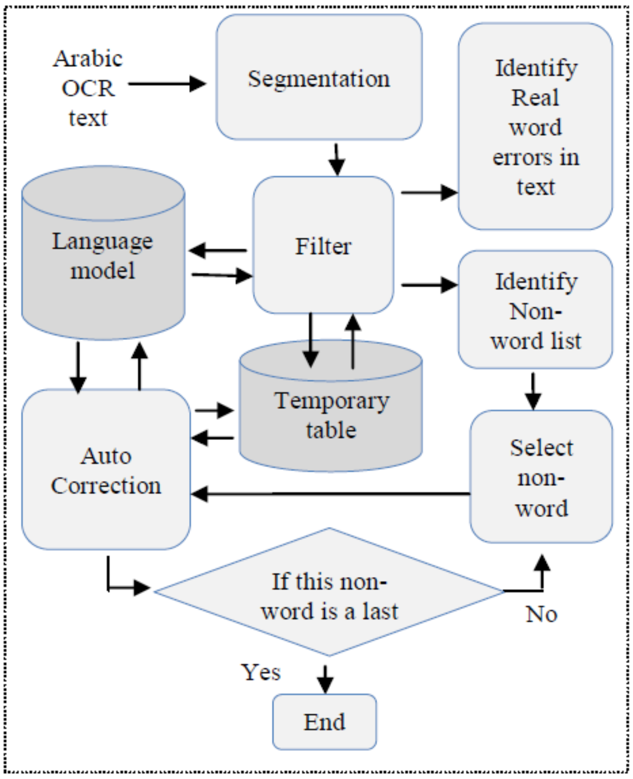


Figure 2. Auto detection and correction process.

Initially, Arabic OCR text is read and stored in memory. In the segmentation stage, the text is split into an array of words "W" using the space between the words as a divider. In filter stage, real word errors are determined by language model and then identified within the input text to make it easier for the user to see them. On the other hand, the non-word errors are identified by filtering all words in array "W" of Arabic OCR text with lexicon words, where any word from array "W" not exists in the lexicon will identify as non-word. Furthermore, a temporary table called "*Temporary*" is created in memory, which has the same properties as the original table in the database. The purpose is to reduce dependence on the database to speed up the correction process. The table "*Temporary*" will fill with records from the database, where each record is referring only to one valid word from Arabic OCR text. At this point, all valid words from Arabic OCR text will be found in the table "*Temporary*", then each non-word will pass through the auto correction stage, and this process will continue for all non-words in the Arabic OCR text. Correction stage involves three levels as shown in Fig. 3. If first level fails, then the operation proceeds to the second and if that fails, then the operation proceeds to the third and if that fails, then the word will be indicated as incorrect word in the Arabic OCR text without correction.

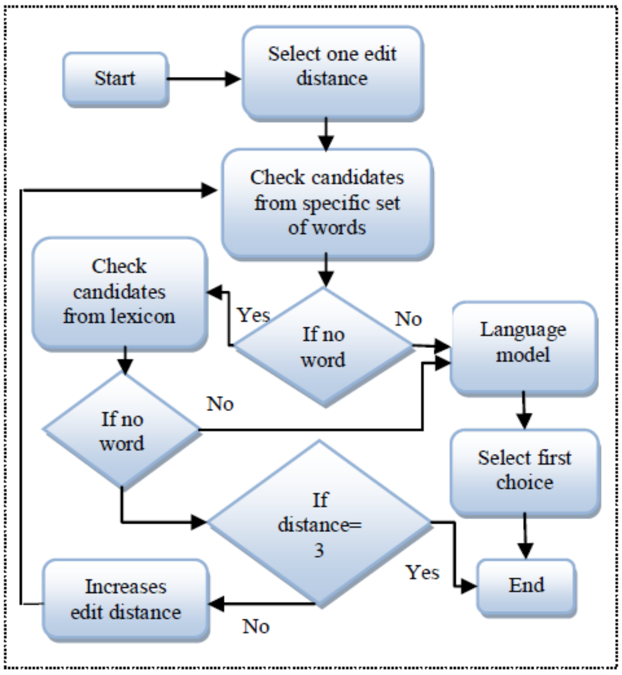


Figure 3. Procedure for generating of candidates list for one non-word.

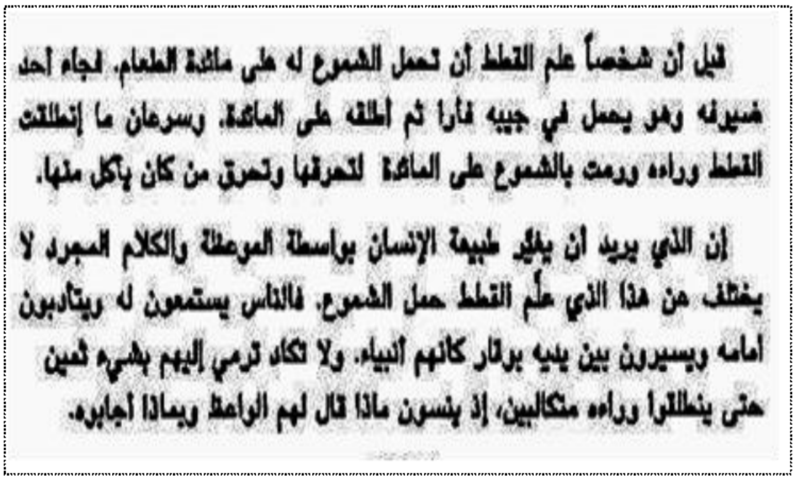
Fig. 3 shows that the first level is performed by searching on single edit distance between the non-word with all words belonging to the correct word that preceded it, or with all words belonging to the correct word that followed. If the result is one candidate word, then the non-word will be replaced by candidate word. If else, then two cases are presented: First, if there is more than one candidate words, then the results are filtered with language model. Second, if no candidate word, then the searching on single edit distance is performed again, but between the non-word and all words in the lexicon, if there are a candidate's words, then the results are filtered with language model. If else, the operation is moved to next level.

At the second and third levels, the previous steps are all returned except that for second level; the searching is on two edit distance and for third level; the searching is on three edit distance. If results of three levels are null, then no auto correction is performed for this non-word. Furthermore, for each level, there are cycles of correction, as an example: after changing a non-word W[i] by a valid word *"X"* depending on the valid word W[i-1], the non-word W[i+m] is corrected based on the word "*X*" and so on.

3. Experimental Result and Evaluation

To implement the method, a prototype is designed using VB.NET under MS Visual Studio.net 2010. The database is designed using MS SQL server 2008. All Arabic Wikipedia articles are downloaded as one dump *xml* file. The result of split of a dump *xml* file into small pieces is 4149 files; their format is *txt*. The sizes of resulted files lie between 0.5 to 5MB. To evaluate the method on incorrect word errors, a Library from Google, which depends on the algorithm of edit distance[2], is used to generate lists of candidates for any incorrect word. Furthermore, *OmniPage18* program with support for Arabic is used to generate the OCR text file. *OmniPage18* program has been subjected on pages of an old book that have poor quality; the resolution of scanning is 300 pixel / inch. Title of the book is "*The analysis of the Arabic human culture*" by *"Ali Alordy"*. Number of words in the pages is 9856. Fig. 4 shows a sample from page from this book.

[2] http://code.google.com/p/google-diff-match-patch/



قيل أن شخصاً علم القطط أن تحمل الشموع له على مائدة الطعام، فجاء أحد ضيوفه وهو يحمل في جيبه فأرا ثم أطلقه على المائدة، وسرعان ما إنطلقت القطط وراءه ورمت بالشموع على المائدة لتحرقها وتحرق من كان يأكل منها.

إن الذي يريد أن يغير طبيعة الإنسان بواسطة الموعظة والكلام المجرد لا يختلف عن هذا الذي علّم القطط حمل الشموع. فالناس يستمعون له ويتأدبون أمامه ويسيرون بين يديه بوقار كأنهم أنبياء، ولا تكاد ترمي إليهم بشيء ثمين حتى ينطلقوا وراءه متكالبين، إذ ينسون ماذا قال لهم الواعظ وبماذا أجابوه.

Figure 4. Sample of page has poor quality.

The test results are shown in Table 1. Row one in Table 1 shows that the total number of words in the OCR text files is 9856 words. Row two shows that the number of both non-word errors and real-word errors resulted from using *OmniPage18* program is 1039 words, where non-word errors are 822 words as shown in row three. Real word errors are 217 words in row four.

Auto detection of non-word errors by using this method is performed and the result is 804 words as shown in row five, with a detection ratio of 97.81% in row six. The detection ratio is calculated through ((804 * 100) / 822). The auto correction of non-word errors using this method performs a valid correction for 772 non-words from 822 (row seven). The error rate improvement for non-word errors is 93.91% (row eight). The improvement ratio is calculated through ((772 * 100) / 822).

Furthermore, the auto detection of real word errors by using this method is performed and the result is 165 real words from 217 as shown in row nine, with a detection ratio of 76.03% in row ten, the detection ratio is calculated through ((165 * 100) / 217). There was no auto correction for real word errors, but they will be identified to the user in input text so that he can correct them. The reason is described in section III. The results in Table 1 show that usage of this method will result in the decrease of error rate in OCR text file.

Table 1. Results of Proposed Method

Words Classification	
Number of words that resulted from using *OmniPage18* program.	9856
Non-word & real word errors that resulted from using *OmniPage18* program.	7.81%
Non-word errors that resulted from using *OmniPage18* program.	822
Real word errors that resulted from using *OmniPage18* program.	217
Non-word errors, that detect by using this method.	804
Detection ratio of non-word errors using this method.	97.81%
Non-word errors that corrected automatically using this method.	772
Error rate improvement of auto correction of non word errors using this method.	93.91%
Real word errors, that detect by using this method.	165
Detection ratio of real word errors using this method.	76.03%

4. Conclusion and Future Work

This paper presents a new method to perform auto detection and correction. The method produced a high accuracy for incorrect words in Arabic OCR text. The method can be used for any language with little modifications. It consists of two parts: extract the context information from huge volume of data, and implementation of automatic correction of errors resulting from the OCR. Data used to extract context information, relied on hundreds of thousands of Arabic articles in the Wikipedia site.

The experimental results show success in the extraction of context information from Wikipedia's articles. It also shows that using the method can reduce considerable error rate. Further research can be done to develop methods that can correct real word errors in Arabic OCR text, develop algorithms to improve images with low resolution, and develop a language model for a grammar in Arabic language is as well needed. Another potential project includes producing e non-static language resources automatically for Arabic Language such as words list of names, verbs, prepositions and adjectives. In addition to that, it can use this method to correct errors in automatic speech recognition of Arabic language.

5. References

[1] Aljarrah, I., et al., *Automated System for Arabic Optical Character Recognition with Lookup Dictionary.* Journal of Emerging Technologies in Web Intelligence, 2012. **4**(4): p. 362-370.
[2] Labidi, M., M. Khemakhem, and M. Jemni, *Grid'5000 Based Large Scale OCR Using the DTW Algorithm: Case of the Arabic Cursive Writing.* Recent Advances in Document Recognition and Understanding, 2011: p. 73.
[3] Oujaoura, M., et al., *Zernike moments and neural networks for recognition of isolated Arabic characters.* International Journal of Computer Engineering Science, 2012. **2**: p. 17-25.
[4] Al-Rashaideh, H., *Preprocessing phase for Arabic Word Handwritten Recognition.* Информационные процессы, 2006. **6**(1).
[5] Khorsheed, M.S., *Off-line Arabic character recognition–a review.* Pattern analysis & applications, 2002. **5**(1): p. 31-45.
[6] Daðason, J.F., *Post-Correction of Icelandic OCR Text*, 2012.
[7] Kukich, K., *Techniques for automatically correcting words in text.* ACM Computing Surveys (CSUR), 1992. **24**(4): p. 377-439.

[8] Magdy, W. and K. Darwish, *Effect of OCR error correction on Arabic retrieval.* Information Retrieval, 2008. **11**(5): p. 405-425.
[9] Naseem, T., *A Hybrid Approach for Urdu Spell Checking*, 2004, National University.
[10] Navarro, G., *A guided tour to approximate string matching.* ACM Computing Surveys (CSUR), 2001. **33**(1): p. 31-88.
[11] Naseem, T. and S. Hussain, *A novel approach for ranking spelling error corrections for Urdu.* Language Resources and Evaluation, 2007. **41**(2): p. 117-128.
[12] Vrandecić, D., P. Sorg, and R. Studer. *Language resources extracted from Wikipedia*. in *Proceedings of the sixth international conference on Knowledge capture*. 2011. ACM.
[13] Cucerzan, S. *Large-Scale Named Entity Disambiguation Based on Wikipedia Data*. in *EMNLP-CoNLL*. 2007.
[14] Bassil, Y. and M. Alwani, *Ocr post-processing error correction algorithm using google online spelling suggestion.* arXiv preprint arXiv:1204.0191, 2012.
[15] Lee, Y.-S. and H.-H. Chen, *Analysis of error count distributions for improving the post-processing performance of OCCR.* Communication of Chinese and Oriental Languages Information Processing Society, 1996. **6**(2): p. 81-86.
[16] Shaalan, K., et al., *Arabic Word Generation and Modelling for Spell Checking.* Language Resources and Evaluation (LREC). Istanbul, Turkey. Pages, 2012: p. 719-725.
[17] Guyon, I. and F. Pereira. *Design of a linguistic postprocessor using variable memory length Markov models*. in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. 1995. IEEE.
[18] Lapata, M. and F. Keller. *The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks*. in *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2004.
[19] Mays, E., F.J. Damerau, and R.L. Mercer, *Context based spelling correction.* Information Processing & Management, 1991. **27**(5): p. 517-522.
[20] Liu, L.-M., et al. *Adaptive post-processing of OCR text via knowledge acquisition*. in *Proceedings of the 19th annual conference on Computer Science*. 1991. ACM.
[21] Tong, X., et al. *OCR Correction and Query Expansion for Retrieval on OCR Data -- CLARIT TREC-5 Confusion Track Report*. in *TREC*. 1996.
[22] Lund, W.B., D.D. Walker, and E.K. Ringger. *Progressive alignment and discriminative error correction for multiple OCR engines*. in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. 2011. IEEE.
[23] Brill, E. 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. Association for Computational Linguistics.
[24] Choudhury, M., et al., *Investigation and modeling of the structure of texting language.* International Journal of Document Analysis and Recognition (IJDAR), 2007. **10**(3-4): p. 157-174.
[25] Lapata, M. and F. Keller, *Web-based models for natural language processing.* ACM Transactions on Speech and Language Processing (TSLP), 2005. **2**(1): p. 3.
[26] Jurafsky, D., et al., *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Vol. 2. 2000: MIT Press.
[27] Attia, M., et al., *An automatically built Named Entity lexicon for Arabic.* 2010.
[28] Remy, M., *Wikipedia: The free encyclopedia.* Reference Reviews, 2002. **16**(6): p. 5-5.
[29] Alkhalifa, M. and H. Rodríguez. *Automatically extending NE coverage of Arabic WordNet using Wikipedia*. in *Proc. Of the 3rd International Conference on Arabic Language Processing CITALA2009, Rabat, Morocco*. 2009.
[30] Jurafsky, D. and H. James, *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech.* 2009.