Fast and Robust Arabic Error Correction System

Michael N. Nawar
Computer Engineering Department
Cairo University
Giza, Egypt
michael.nawar@eng.cu.edu.eg

Moheb M. Ragheb
Computer Engineering Department
Cairo University
Giza, Egypt
moheb.ragheb@eng.cu.edu.eg

Abstract

In this paper we describe the implementation of an Arabic error correction system developed for the EMNLP2014 shared task on automatic error correction for Arabic text. We proposed a novel algorithm, where we find some correction rules and calculate their probability based on the training data, they we rank the correction rules, then we apply them on the text to maximize the overall F-score for the provided data. The system achieves and F-score of 0.6573 on the test data.

1 Introduction

Traditional techniques in text correction is the generation of a large set of candidates for an incorrect word using different approaches like enumerating all possible candidates in edit distance of one. Then, all the candidates are ranked such that the best candidates are ranked on the top of the list. Finally, the best candidate is chosen to replace incorrect word.

The traditional techniques are slow, since the generation of a large set of candidates is time consuming task. Also, it doesn't take into consideration the overall score of the system. While, in this paper we apply a novel technique in automatic error correction, where we take into consideration the correction rules, not the variants. In the propose technique, we order corrections to be applied on text to maximize the F-score.

This shared task was on automatic Arabic text correction. For this task, the Qatar Arabic Language Bank (QALB) corpus (Mohit et. al, 2014) was provided. The QALB corpus contains a preprocessed input text with some features extracted and the corrected output. The main issue in the shared task, that the tools used for the extraction of the provided features wasn't provided. So, we had a choice, to create an algorithm that can deal with missing features, or to generate our own set of features. Finally, we have chosen to generate our own set of features.

The proposed framework could be described as a probabilistic rule-based framework. During the training of this framework, we extracted some rules and assign a probability to each rule as shown later in section 3. The extracted rules are then sorted based on their probabilities. And during the test, we apply the rules from the highest probability to the lowest probability one by one, on the entire test data till a stopping criteria is satisfied. During the algorithm we have some kind of heuristic to estimate the F-score after each rule is apply. The stopping criteria for the algorithm is that the estimated F-score start to decrease.

This paper is organized as follow, in section 2, an overview of the related work in the field of error correction is discussed. In section 3, the proposed system and its main components are explained. The evaluation process is presented in section 4. Finally, concluding remarks and future work are presented in section 5.

2 Related Work

Most of the work done in the field automatic error correction for text, is made for English language (Kukich, 1992; Golding and Roth, 1999; Carlson and Fette, 2007; Banko and Brill, 2001). Arabic spelling correction has also received considerable interest, Ben Othmane Zribi and Ben Ahmed, (2003) have proposed a new aiming to reduce the number of proposals given by automatic Arabic spelling correction tools, which have reduced the proposals by about 75%. Haddad and Yaseen (2007) took into consideration the complex nature of the Arabic language and the effect of the root-pattern relationship to lo-

Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 143–147,
October 25, 2014, Doha, Qatar.

cate, reduce and rank the most probable correction candidates in Arabic derivative words to improve the process of error detection and correction. Hassan et al. (2008) used a finite state automata to propose candidates corrections, then assign a score to each candidate and choose the best correction in the context. Shaalan et al. (2010) developed an error correction system to Arabic learners. Alkanhal et al. (2012) have developed an error correction system and they emphasized on space insertion and deletion. Zaghouani et al. (2014) provided a large scale dataset for the task of automatic error correction for Arabic text.

3 The Proposed System

The main system idea is explained by the algorithm, in figure 1. The algorithm has two inputs: the set of sentences that need to be modified T[1..n], and the set of correction rules C[1..m] that could be applied to text. The algorithm has one single output: the set of modified sentences T'[1..n]. The algorithm could be divided into two main component: the initialization and the main loop.

Input: T[1..n], C[1..m]
Output: T'[1..n]
1: T' = T
2: Gold Edits = #Words in Test * # Gold Edits in Train / # Words in Train
3: Correct Edits = 0
4: Performed Edits = 0
5: Precision = 0
6: Recall = 0
7: Old F-score = 0
8: F-score = 0
9: Do
10: T' = T
11: Old F-score = F-score
12: Get next correction "c" with the highest probability "p" from C
13: Apply the correction "c" on T
14: N = number of changes between T and T'
15: Performed Edits = Performed Edits + N
16: Correct Edits = Correct Edits + p * N
17: Precision = Correct Edits / Performed Edits
18: Recall = Correct Edits / Gold Edits
19: F-score = 2*Precision*Recall / (Precision+Recall)
20: while F-score > Old F-score do
21: return T'

Figure 1: Proposed Algorithm

First, the initialization part of the algorithm starts from line 1 to line 8. In the first line, the sentences are copied from T[1..n] to T'[1..n]. In line number 2, the number of errors in the test set T[1..n] is expected using the rate of errors in the train set (#error / #words). In lines 3 to 8, the variables used in the algorithm are initialized to zero.

The main loop of the algorithm starts from line 9 to line 20. In line 9, the loop begins, and the sentences are copied from T[1..n] to T'[1..n] and the F-score is copied to old F-score, in linarae 10 and 11. Then the first not applied correction with the highest probability to be correct is correct is chosen in line 12. In line 13, the correction is applied on the text T[1..n]. Then we calculate the number of changes between T[1..n] and T'[1..n], in line 14. And based on the expected number of changes, we update the expected number of performed edits in line 14. Also, we update the expected number of the correct edits based on the number of change and the probability of a change to be correct in line 15. In lines 17 to 19, we calculate the expected precision, recall and F-score based on the expected gold edits, performed edits, and correct edits calculated at lines 2, 14, and 15. If the F-score is higher than the old F-score, which means that applying the correction c on the text T[1..n] will increase the expected F-score, then go to line 9 and start a new iteration in the loop. And if the F-score is lower than the old F-score, which means that applying the correction c on the text T[1..n] will decrease the expected F-score, then exit the loop and return the modified text T'[1..n].

After we have discussed the main idea of algorithm, in the following subsections we will discuss some of the extracted corrections rules and the calculation of the probability of each rule. These rules and their probabilities are compiled by analyzing the training data.

3.1 Morphological Analyzer Corrections Rules

We used a morphological analyzer, BAMA-v2.0 (Buckwalter Arabic morphological analyzer version 2.0) (Buckwalter, 2010), in the extraction of a correction rule. This rule will be used to solve the errors caused by the exchange between some characters like: ("ا", "A"), ("إ", ">"), ("إ", "<") and ("ه", "h"), ("ة", "p") and ("ي", "y"), ("ى", "Y").

RULE: We analyze a word with the morphological analyzer, if all the solutions of the word have the same form that is different from the

word, then change the word by the solutions form.

For example, the word ("احمد", "AHmd"), when the word is analyzed by the morphological analyzer, there are 20 different solutions, 14 are proper noun ("أحمد", ">Hmd", "Ahmed") and the remaining 6 of them are verb ("أحمد", ">Hmd", "I praise"). Since all the solution of the word ("احمد", "AHmd") have the form ("أحمد", ">Hmd"), then we will change ("احمد", "AHmd") to ("أحمد", ">Hmd"). Another example, the word ("امام", "AmAm"), when the word is analyzed by the morphological analyzer, there are 24 different solutions, 12 of them have the form ("أمام", ">mAm"), and the other 12 have the form ("إمام", "<mAm"), so we leave it unchanged.

To calculate the correctness probability of the rule, we apply the following rule to all the training set, then we calculate the number of correct edits, and the number of performed edits, finally we calculate the probability as the ratio between the correct and the performed edits.

3.2 Colloquial to Arabic Corrections Rules

To convert the colloquial Arabic words to Arabic words, we have compiled some rules as shown below:

RULE: Replace a word or a phrase by a specific word or phrase from a list extracted from the training set provided in Qalb shared task (Mohit et. al, 2014).

From example replace the word ("احنا", "AHnA", "we") by the word ("نحن", "nHn", "we").

RULE: Replace a word or phrase with a specific word or phrase based on its context.

RULE: Replace a word or phrase with a specific pattern to another word or phrase.

From example replace the word ("بيلعب", "bylEb", "is playing") by the word ("يلعب", "ylEb", "is playing").

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data.

3.3 The Single Character Spelling Errors Correction

The single character spelling errors are divided into four main subcategories: replace character by another character, insert character, delete character, and transpose two adjacent characters. For these four errors, we have conducted four types of rules.

RULE 1: We analyze a word with the morphological analyzer, if it is outside the corpus, and it not defined in the correct words in qalb corpus (the words that don't change) try to change one character by a specific character, if the new word is recognized by the morphological analyzer or it is inside the corpus, then change the word and keep the new solution.

For example, if we have a word ("بعظ", "bEZ") and a rule that change the character ('ظ', 'Z') to ('ض', 'D'). And the word ("بعض", "bED") is recognized by the morphological analyzer, then we change the word ("بعظ", "bEZ") to ("بعض", "bED"). Another example, if we have the word ("بعظ", "bEZ") and a rule that change the character ('ع', 'E') to ('غ', 'g'). And the word ("بغظ", "bgZ") is not recognized by the morphological analyzer and it is outside the Qalb corpus, then we don't change the word.

RULE 2: We analyze a word with the morphological analyzer, if it is outside the corpus, and it not defined in the correct words in qalb corpus (the words that don't change) try to insert one specific character between a pair of specific characters, if the new word is recognized by the morphological analyzer or it is inside the corpus, then change the word and keep the new solution.

RULE 3: We analyze a word with the morphological analyzer, if it is outside the corpus, and it not defined in the correct words in qalb corpus (the words that don't change) try to delete one specific character from a triplet of specific characters, if the new word is recognized by the morphological analyzer or it is inside the corpus, then change the word and keep the new solution.

RULE 4: We analyze a word with the morphological analyzer, if it is outside the corpus, and it not defined in the correct words in Qalb corpus (the words that don't change) try to replace a pair of characters to the transpose of the pair of characters, if the new word is recognized by the morphological analyzer or it is inside the corpus, then change the word and keep the new solution.

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data, and it differs from one character to another (i.e. the two examples in rule 1, will have different correctness probabilities based on the training data).

3.4 The Space Insertion Errors Correction

The space insertion error correction is the process of splitting an incorrect word to multiple correct word.

RULE: If there is a character concatenated after taa marbouta ('ة', 'p'), insert a space between them.

RULE: If the word starts with negation particle, split negation particle from it.

RULE: If the word starts with vocative particle, split vocative particle from it.

RULE: If the word starts with vocative particle, split vocative particle from it.

RULE: We analyze a word with the morphological analyzer, if it is outside the corpus, and it not defined in the correct words in Qalb corpus (the words that don't change) try to find the long substring from the word, that keep another substring, where both of them are recognized by the morphological analyzer.

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data.

3.5 The Space Deletion Errors Correction

The space deletion errors correction is the process of merging multiple tokens into one correct word.

RULE: Merge conjunction particles, with their succeeding token.

RULE: If two out of corpus tokens could be merged to an inside the corpus word, then merge them.

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data.

3.6 Punctuation Errors Corrections

The punctuation errors are hard to correct because they depends on the meaning of the sentence, and require almost full understanding of the sentence. However, we have conducted some rules for the punctuation, for example:

RULE: If the sentence doesn't end with a punctuation point from (".", "!", "؟"), then add a point at the end of the sentence.

RULE: Insert a punctuation mark before a certain word.

For example, insert a semicolon before the word ("لأنه", "l>nH", "because he").

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data.

3.7 Syntactic Errors Corrections

The syntactic errors is one of the most difficult error to correct. For this task we apply a simple kind of a grammatical analyzer to assign simple grammatical tag to some words. One simple grammatical system, is the one to determine genitive noun. Nouns are genitive mainly if they occur after a preposition, or if they are possessives (definite noun after indefinite noun) or if they are adjectives of genitive nouns, or if they are conjunction with genitive noun.

RULE: Plural and Dual genitive nouns that end with ("ون", "wn") or ("ان", "An") should end with ("ين", "yn").

The correctness probability of each rule is the ratio between the correct and the performed edits when this rule is applied on the train data.

3.8 Additional Corrections Rules

Finally, we generated some rules that present the data on a correct format as the training data and we will assign their correctness probability manually to be equal to 1.

RULE: Remove kashida (tatweel) from text.

RULE: Replace "*" if between parenthesis by the Arabic character ('ذ', '*').

RULE: If a character is repeated consecutively more than twice inside a word, remove the extra characters except if the word consists of only one char like ("هههههه", "hhhhh").

RULE: Write a comma between two numbers.

4 Evaluation of the System

For the evaluation of the system, we used the M2 scorer by Dahlmeier and Ng (2012). When we evaluated the system with the development dataset, we have reached an F-score of 0.6817; and when the system is evaluated the test dataset, we have reached and F-score of 0.6573.

The proposed algorithm is very fast compared to traditional error correction algorithm. In traditional error correction algorithm, you generate all possible variants of an incorrect word, then you rank the solutions and choose the best solution. But, in the proposed algorithm, you rank the rules during the training time, and you apply one rule at the time until you find an appropriate solution of an incorrect word.

For example, let's consider single character replace spelling error, if the incorrect word length is five characters, so you need to make ((28-1)*5) iterations to generate all possible variants of a word, while in the proposed algorithm you generate one variant at the time, and you might stop after that.

5 Conclusion

In this paper we have presented a novel and fast algorithm for the automatic text correction for Arabic. The proposed algorithm has a good F-score, and the system has the potential to be further improved. As a future work, the punctua-

tion error correction might need to be further improved. And the expected number of gold edits, could be improved or calculated on the sentence level. And finally, the rules used in the framework could be extended by further analysis of the training data.

References

Mohamed I. Alkanhal, Mohammed A. Al-Badrashiny, Mansour M. Alghamdi, and Abdulaziz O. AlQabbany. 2012. Automatic Stochastic Arabic Spelling Correction with Emphasis on Space Insertions and Deletions. *IEEE Transactions on Audio, Speech & Language Processing*, 20:2111–2122.

Michele Banko and Eric Brill, 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France.

Chiraz Ben Othmane Zribi and Mohammed Ben Ahmed. 2003. Efficient Automatic Correction of Misspelled Arabic Words Based on Contextual Information. In *Proceedings of the Knowledge-Based Intelligent Information and Engineering Systems Conference*, Oxford, UK.

Tim Buckwalter. 2010. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2004L02. ISBN 1-58563-324-0.

Andrew Carlson and Ian Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA).*

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceeding of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.*

Andrew R. Golding and Dan Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Bassam Haddad and Mustafa Yaseen. 2007. Detection and Correction of Non-Words in Arabic: A Hybrid Approach. *International Journal of Computer Processing Of Languages (IJCPOL).*

Ahmed Hassan, Sara Noeman, and Hany Hassan. 2008. Language Independent Text Correction using Finite State Automata. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP 2008).*

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4).

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid, 2014. The First shared Task on Automatic Text Correction for Arabic. In *Proceedings of EMNLP workshop on Arabic Natural Language Processing*. Doha, Qatar.

Khaled Shaalan, Rana Aref, and Aly Fahmy. 2010. An approach for analyzing and correcting spelling errors for non-native Arabic learners. In *Proceedings of Informatics and Systems (INFOS).*

Wajdi Zaghouani, Behrang Mohit, Nizar Habash, Ossama Obeid, Nadi Tomeh, Alla Rozovskaya, Noura Farra, Sarah Alkuhlani, and Kemal Oflazer. 2014. Large Scale Arabic Error Annotation: Guidelines and Framework. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland.