CUFE@QALB-2015 Shared Task: Arabic Error Correction System

Michael Nawar Ibrahim
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 WANLP-2015 shared task on automatic error correction for Arabic text. We proposed improvements to a previous statistical rule based system, where we use the words patterns to improve the error correction, also we have used a statistical system the syntactic error correction rules. The system achieves an F-score of 0.7287 on the Alj-test-2015 dataset, and an F-score of 0.3569 on the L2-test-2015 dataset.

1 Introduction

This paper presents improvements to a previously developed rule-based probabilistic system (Nawar and Ragheb, 2014). We first make use of a unique Arabic feature, which is the word pattern to extract more rules for the system. Also, we have proposed a probabilistic Arabic grammar analyzer instead of a simple rule-based one proposed in the previous work.

This shared task was on automatic Arabic text correction. For this task, the Qatar Arabic Language Bank (QALB) corpus (Rozovskaya et. al, 2015) was provided. It is an extension of the first QALB shared task (Mohit et al., 2014) that took place last year. QALB-2014 addressed errors in comments written to Aljazeera articles by native Arabic speakers (Zaghouani et al., 2014). This year's competition includes two tracks, and, in addition to errors produced by native speakers, also includes correction of texts written by learners of Arabic as a foreign language (L2) (Zaghouani et al., 2015). The native track includes Alj-train-2014, Alj-dev-2014, Alj-test-2014 texts from QALB-2014. The L2 track includes L2-train-2015 and L2-dev-2015. This data was released for the development of the systems. The systems were scored on blind test sets Alj-test-2015 and L2-test-2015.

The proposed framework could be described as a probabilistic rule-based framework. During the training of this framework, we extracted error correction rules and compute 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 improvements in the correction rules are discussed in section 4. The evaluation process is presented in section 5. Finally, concluding remarks and future work are presented in section 6.

2 Related Work

During the last two decades, there was an increasing interest in the problem of error correction, and most of the work done in that field, is made for English language (Kukich, 1992; Golding and Roth, 1999; Carlson and Fette, 2007; Banko and Brill, 2001). Recently, Arabic spelling correction has also received considerable interest. Ben Othmane Zribi and Ben Ahmed, (2003) have reduced the number of alternatives to a wrong word by about 75%. Haddad and Yaseen (2007) used a unique Arabic language feature, word root-pattern relationship, to locate, reduce and rank the most probable correction

Proceedings of the Second Workshop on Arabic Natural Language Processing, pages 133–137,
Beijing, China, July 26-31, 2015.

candidates in Arabic derivative words to improve the process of error detection and correction. Hassan et al. (2008) proposed an error correction system that use a finite state automata to propose candidate corrections for wrong words, then assign a score to each candidate and choose the best correction based on the context. Shaalan et al. (2010) developed an error correction system to Arabic learners. Alkanhal et al. (2012) have developed an error correction system that emphasizes on space insertion and deletion errors.

Last year, in the QALB 2014 shared task, multiple systems for text error correction were proposed. Jeblee et al. (2014) proposed a pipeline consisting of rules, corrections proposed by MADAMIRA (Pasha et al., 2014), a language model for spelling mistakes, and a statistical machine-translation system. Rozovskaya et al. (2014) used multiple approaches to correct the wrong word including: corrections proposed by MADAMIRA, a Maximum Likelihood model trained on the training data, regular expressions; a decision-tree classifier for punctuation errors trained on the training data, an SVM character-level error correction model, a Naïve Bayes classifier trained on the training data and the Arabic Gigaword corpus, and finally, they analyzed the results to find the best combination of correction technique that produce the best result.

3 The Proposed System

This paper is an extension to the work by Nawar and Ragheb (2014). 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.

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 variable Pattern[1..n] holds the patterns of the words in the sentences T[1..n]. For example, the pattern of ("كاتب","kAtb", "writer") is ("فاعل","fAEl) and the pattern of ("مكتب","mktb", "office") is ("مفعل","mfEl"). For the extraction of the word pattern, we assign for each stem in the stem table of the morphological analyzer (BAMA v2) an appropriate pattern, then we assign the word a pattern based on its stem pattern, prefix and suffix. For example, we assign for the stem ("ستخدم", "stxdm") the pattern ("ستفعل", "stfEl"), and when we analyze the word ("مستخدمين", "mstxdmyn", "users") – that have a prefix ("م","m") and a suffix ("ين", "yn") – we assign to it the pattern ("مستفعلين", "mstfElyn"), and when we analyze the word ("يستخدمون", "ystxdmwn", "they use")–that have a prefix ("ي","y") and a suffix ("ون", "wn") – we assign to it the pattern ("يستفعلون", "ystfElwn"). We don't assign a pattern to a word if the word is Arabized (nouns borrowed from foreign languages) like ("كمبيوتر", "kmbywtr", "computer") or ("أمريكا", ">mrykA", "America"), or if the word is fixed (words used by Arabs, and do not obey the Arabic derivation rules) like ("هذا", "h*A", "this") or ("كل", "kl", "every").

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: Pattern[1..n] = Extract Patterns of T
10: Do
11: T' = T
12: Old F-score = F-score
13: Get next correction "c" with the highest probability "p" from C
14: Apply the correction "c" on T
15: Update Patterns based on "c"
16: N = number of changes between T and T'
17: Performed Edits = Performed Edits + N
18: Correct Edits = Correct Edits + p * N
19: Precision = Correct Edits / Performed Edits
20: Recall = Correct Edits / Gold Edits
21: F-score = 2*Precision*Recall / (Precision+Recall)
22: while F-score > Old F-score do
23: return T'

Figure 1: Proposed Algorithm

The main loop of the algorithm starts from line 10 to line 22. In line 10, 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 lines 11 and 12. Then the first not applied correction with the highest probability to be correct is correct is chosen in line 13. In line 14, the correction is applied on the text T[1..n], and in line 15 the Patterns[1..n] is updated based on the corrections performed. Then we calculate the number of changes between T[1..n] and T'[1..n], in line 16. And based on the expected number of changes, we update the expected number of performed edits in line 16. 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 17. In lines 19 to 21, we calculate the expected precision, recall and F-score based on the expected gold edits, performed edits, and correct edits calculated at lines 2, 16, and 17. 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 10 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].

To calculate the correctness probability of a rule, we apply the rule to 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. For example, let's consider the rule to be a simple edit rule as shown below:

RULE: Replace the word **W1** by the word **W2**.

W1	W2	Correct Edits	Performed Edits	p
امريكا AmrykA	أمريكا >mrykA	785	786	0.99
اميركا AmyrkA	أمريكا mrykA	25	54	0.46
اميركا AmyrkA	أميركا >myrkA	29	54	0.54
لان lAn	لأن l>n	690	702	0.98
اسرائيل AsrA}yl	إسرائيل <srA}yl	1087	1088	0.99
ان An	إن <n	1507	9359	0.16

Table 1: Examples of Correction Rules Precisions

The calculation of the correctness probability is the same when applied to more complex rules. One naïve method to generate rules, is to extract all edit rules from the training set and, calculate their probabilities, and finally adding them to the rules file. The algorithm will deal with multiple edit rules with the same first word (W1) by ignoring the rules with smaller probability. For example, ("اميركا", "AmyrkA") if it is going to be modified, it will always be edited to ("أميركا", ">myrkA").

4 Correction Rules

After we have discussed the main idea of algorithm, in the following subsections we will discuss some of the extracted corrections rules based on the word pattern and the syntactic error correction rules. These rules and their probabilities are compiled by analyzing the training data.

4.1 Patterns Corrections Rules

We have modified the morphological analyzer, BAMA-v2.0 (Buckwalter Arabic morphological analyzer version 2.0) (Buckwalter, 2010), to be able to assign an appropriate pattern to each word.

These patterns will be used to make rules based on the words patterns. For example, removing the unnecessary determinant ("ال", "Al"), or adding necessary determinant ("ال", "Al") which are common errors in the second language text. Another example, these patterns could be used to correct errors based on type mismatch between masculine or feminine words. Also, it could be used to correct errors on count mismatch is plural or dual or singular words. Finally, simple punctuation rules could be put based on the words patterns.

4.2 Syntactic Errors Corrections

The syntactic errors are the most difficult error to correct. For this task we apply a statistical grammatical analyzer to assign simple grammatical tag to the words. And based on these tags, we apply different correction rules. For example, nouns are genitive if they occur after a preposition ("حرف جر"," Hrf jr"), or if they are possessives ("مضاف إليه", "mDAf <lyh") or if they are adjectives ("نعت", "nEt") of genitive nouns, or if they are conjunction ("معطوف", "mETwf") with genitive noun, or if they are appositions ("بدل", "bdl") to genitive noun. And based on these facts the following simple rule could be applied.

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

For the construction of this grammatical system, we used the data provided by Ibrahim 2015. To assign an appropriate grammatical tag to the tokens, the classifier training and testing could be characterized as follow:

Input: A sequence of transliterated Arabic tokens processed from left-to-right with break markers for word boundaries.

Context: A window of -3/+3 tokens centered at the focus token.

Features: The 7 tokens themselves, POS tag decisions for tokens within context, the base phrase chunk for tokens within the context, the root of the words within the context,, the pattern of the words within the context, whether the word is definite or not, whether the word is feminine or not, and whether the word is plural or dual or singular.

Classifier: CRF suite classifier.

5 Results and discussion

For the evaluation of the system, we used the M2 scorer by Dahlmeier and Ng (2012). When we evaluated the system with the Alj-dev-2014 dataset, we have reached an F-score of 0.6872; and F-score of 0.6668 on Alj-test-2014 dataset and an F-score of 07287 when evaluated on Alj-test-2015 dataset. For the second language, the system achieved an F-score of 0.5673 on L2-dev-2015 dataset, and an F-score of 0.3569 on the L2-test-2015 dataset.

The proposed algorithm is very fast compared to traditional error correction algorithm, since that the algorithm ranks the rules during the training time, and applies one rule at the time until the expected F-score decreases. But as a direct result to the design of the algorithm, and its concern in maximizing the overall F-score of the test set, the algorithm may apply the rule with the highest probability till it saturates, i.e. it applies the rule to the first few errors and stops if this is going to decrease the expected value for the F-score.

Also, one can notice, that this algorithm may apply correction rules with probability less than 0.5 (which means that applying this rule is expected to cause more errors than correcting wrong word), it all depends on the value of the precision and the recall. Although that seems to be a little bit not logical but this could be justified by its ability to maximize the F-score. This is not an issue from the algorithm, this problem arises from the properties of the F-score. This shows the problem in using the F-score for evaluating the text error correction systems, and it opens the doors for researchers to find a new metric to measure the performance of the text error correction systems.

Another problem in the F-score as an evaluation metric for the error correction systems is that if a word contains more than one error, if you correct one of these errors and not the others the entire word is considered wrong. An example of a word that contains one syntax error and anther syntactic error is: ("العراقون", "AlErAqwn") in the context ("مع العراقون", "mE AlErAqwn"). The word should be corrected to ("العراقيين", "AlErAqyyn"), but if it is corrected to ("العراقيون", "AlErAqywn") which means the syntax error is handled and the syntactic is not, the entire word will be considered as wrong.

6 Conclusions and Future Work

In this paper we have improved a previously proposed system for text correction for Arabic. The proposed algorithm has has the potential to be further improved. As a future work, the punctuation error correction might need to be further improved. And finally, the rules used in the framework could be extended by further analysis of the training data. As a future work, we can merge the proposed algorithm with other error correction technique, and use it as an acceptance-rejection scheme for the other error correction algorithm. Another future work, is to propose another evaluation metric for the text error correction systems.

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).

Arfath Pasha and Mohamed Al-Badrashiny and Ahmed El Kholy and Ramy Eskander and Mona Diab and Nizar Habash and Manoj Pooleery and Owen Rambow and Ryan Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC)

Alla Rozovskaya, Nizar Habash, Ramy Eskander, Noura Farra, and Wael Salloum. 2014. The Columbia System in the QALB-2014 Shared Task on Arabic Error Correction. In Proceedings of EMNLP Workshop on Arabic Natural Language Processing: QALB Shared Task.

Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The Second QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of ACL workshop on Arabic Natural Language Processing*. Beijing, China.

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).*Ibrahim, Michael Nawar. 2015. Statistical Arabic Grammar Analyzer. *Computational Linguistics and Intelligent Text Processing. Springer International Publishing. 187-200.*

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.

Nawar, Michael N., and Moheb M. Ragheb. 2014. Fast and robust arabic error correction system. *ANLP 2014. 143.*

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.

Wajdi Zaghouani, Nizar Habash, Houda Bouamor, Alla Rozovskaya, Behrang Mohit, Abeer Heider, and Kemal Oflazer. 2015. Correction Annotation for Non-Native Arabic Texts: Guidelines and Corpus. In *Proceedings of the Ninth Linguistic Annotation Workshop*, Denver, Colorado, USA. *Association for Computational Linguistics*. *129-139.*