CMUQ@QALB-2014: An SMT-based System for Automatic Arabic Error Correction

Serena Jeblee[1] **, Houda Bouamor**[2]**, Wajdi Zaghouani**[2] **and Kemal Oflazer**[2]
[1]**Carnegie Mellon University**
sjeblee@cs.cmu.edu
[2]**Carnegie Mellon University in Qatar**
{hbouamor,wajdiz}@qatar.cmu.edu, ko@cs.cmu.edu

Abstract

In this paper, we describe the CMUQ system we submitted to The ANLP-QALB 2014 Shared Task on Automatic Text Correction for Arabic. Our system combines rule-based linguistic techniques with statistical language modeling techniques and machine translation-based methods. Our system outperforms the baseline and reaches an F-score of 65.42% on the test set of QALB corpus. This ranks us 3rd in the competition.

1 Introduction

The business of text creation and editing represents a large market where NLP technologies might be applied naturally (Dale, 1997). Today's users of word processors get surprisingly little help in checking spelling, and a small number of them use more sophisticated tools such as grammar checkers, to provide help in ensuring that a text remains grammatically accurate after modification. For instance, in the Arabic version of Microsoft Word, the spelling checker for Arabic, does not give reasonable and natural proposals for many real-word errors and even for simple probable errors (Haddad and Yaseen, 2007).

With the increased usage of computers in the processing of natural languages comes the need for correcting errors introduced at different stages. Natural language errors are not only made by human operators at the input stage but also by NLP systems that produce natural language output. Machine translation (MT), or optical character recognition (OCR), often produce incorrect output riddled with odd lexical choices, grammar errors, or incorrectly recognized characters. Correcting human/machine-produced errors, or post-editing, can be manual or automated. For morphologically and syntactically complex languages, such as Modern Standard Arabic (MSA), correcting texts automatically requires complex human and machine processing which makes generation of correct candidates a challenging task.

For instance, the Automatic Arabic Text Correction Shared Task is an interesting testbed to develop and evaluate spelling correction systems for Arabic trained either on naturally occurring errors in texts written by humans (e.g., non-native speakers), or machines (e.g., MT output). In such tasks, participants are asked to implement a system that takes as input Modern Standard Arabic texts with various spelling errors and automatically correct them. In this paper, we describe the CMUQ system we developed to participate in the The First Shared Task on Automatic Text Correction for Arabic (Mohit et al., 2014). Our system combines rule-based linguistic techniques with statistical language modeling techniques and machine translation-based methods. Our system outperforms the baseline, achieves a better correction quality and reaches an F-score of 62.96% on the development set of QALB corpus (Zaghouani et al., 2014) and 65.42% on the test set.

The remainder of this paper is organized as follows. First, we review the main previous efforts for automatic spelling correction, in Section 2. In Section 3, we describe our system, which consists of several modules. We continue with our experiments on the shared task 2014 dev set (Section 4). Then, we give an analysis of our system output in Section 5. Finally, we conclude and hint towards future improvement of the system, in Section 6.

2 Related Work

Automatic error detection and correction include automatic spelling checking, grammar checking and post-editing. Numerous approaches (both supervised and unsupervised) have been explored to improve the fluency of the text and reduce the percentage of out-of-vocabulary words using NLP tools, resources, and heuristics, e.g., morphological analyzers, language models, and edit-distance measure (Kukich, 1992; Oflazer, 1996; Zribi and Ben Ahmed, 2003; Shaalan et al., 2003; Haddad and Yaseen, 2007; Hassan et al., 2008; Habash, 2008; Shaalan et al., 2010). There has been a lot of work on error correction for English (e.g., (Golding and Roth, 1999)). Other approaches learn models of correction by training on paired examples of errors and their corrections, which is the main goal of this work.

For Arabic, this issue was studied in various directions and in different research work. In 2003, Shaalan et al. (2003) presented work on the specification and classification of spelling errors in Arabic. Later on, Haddad and Yaseen (2007) presented a hybrid approach using morphological features and rules to fine

tune the word recognition and non-word correction method. In order to build an Arabic spelling checker, Attia et al. (2012) developed semi-automatically, a dictionary of 9 million fully inflected Arabic words using a morphological transducer and a large corpus. They then created an error model by analyzing error types and by creating an edit distance ranker. Finally, they analyzed the level of noise in different sources of data and selected the optimal subset to train their system. Alkanhal et al. (2012) presented a stochastic approach for spelling correction of Arabic text. They used a context-based system to automatically correct misspelled words. First of all, a list is generated with possible alternatives for each misspelled word using the Damerau-Levenshtein edit distance, then the right alternative for each misspelled word is selected stochastically using a lattice search, and an n-gram method. Shaalan et al. (2012) trained a Noisy Channel Model on word-based unigrams to detect and correct spelling errors. Dahlmeier and Ng (2012a) built specialized decoders for English grammatical error correction. More recently, (Pasha et al., 2014) created MADAMIRA, a system for morphological analysis and disambiguation of Arabic, this system can be used to improve the accuracy of spelling checking system especially with Hamza spelling correction.

In contrast to the approaches described above, we use a machine translation (MT) based method to train an error correction system. To the best of our knowledge, this is the first error correction system for Arabic using an MT approach.

3 Our System

Our system is a pipeline that consists of several different modules. The baseline system uses a spelling checking module, and the final system uses a phrase-based statistical machine translation system. To preproces the text, we use the provided output of MADAMIRA (Pasha et al., 2014) and a rule-based correction. We then do a rule-based post-processing to fix the punctuation.

3.1 Baseline Systems

For the baseline system, we try a common spelling checking approach. We first pre-process the data using the features from MADAMIRA (see Feature 14 Replacement), then we use a noisy channel model for spelling checking.

Feature 14 Replacement

The first step in the pipeline is to extract MADAMIRA's 14th feature from the *.column file* and replace each word in the input text with this form. MADAMIRA uses morphological disambiguation and SVM analysis to select the most likely fully diacritized Arabic word for the input word. The 14th feature represents the undiacritized form of the most likely word. This step corrects many Hamza placement or omission errors, which makes a good base for other correction modules.

Spelling Correction

The spelling checker is based on a noisy channel model - we use a word list and language model to determine the most probable correct Arabic word that could have generated the incorrect form that we have in the text. For detecting spelling errors we use the AraComLex word list for spelling checking (Attia et al., 2012), which contains about 9 million Arabic words.[1] We look up the word from the input sentence in this list, and attempt to correct those that are not found in the list. We also train a mapping of incorrect words and possible corrections from the edits in the training data. If the word is in this map, the list of possible corrections from the training data becomes the candidate list. If the word is not in the trained map, the candidate list is created by generating a list of words with common insertions, substitutions, and deletions, according to the list in (Attia et al., 2012). Each candidate is generated by performing these edits and has a weight according to the edit distance weights in the list. We then prune the candidate list by keeping only the lowest weight words, and removing candidates that are not found in the word list. The resulting sentence is scored with a 3-gram language model built with KenLM (Heafield et al., 2013) on the correct side of the training data. The top one sentence is then kept and considerd as the "corrected" one.

This module handles spelling errors of individual words; it does not handle split/merge errors or word reordering. The spelling checker sometimes attempts to correct words that were already correct, because the list does not contain named entities or transliterations, and it does not contain all possible correct Arabic words. Because the spelling checker module decreased the overall performance, it is not included in our final system.

3.2 Final System

Feature 14 Replacement

The first step in our final system is Feature 14 Replacement, as described above.

Rule-based Clitic Correction

With the resulting data, we apply a set of rules to reattach clitics that may have been split apart from the base word. After examining the train dataset, we realized that 95% of word merging cases involve "و" attachment. When found by themselves, the clitics are attached to either the previous word or next word, based on whether they generally appear as prefixes or suffixes. The clitics handled by this module are specified in Table 2.

We also remove extra characters by replacing a sequence of 3 or more of the same character with a single

[1] http://sourceforge.net/projects/arabic-wordlist/

	Dev					
	Exact Match			No Punct		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Feature 14	0.7746	0.3210	0.4539	0.8100	0.5190	0.6326
Feature 14 + Spelling checker (baseline)	0.4241	0.3458	0.3810	0.4057	0.4765	0.4382
Feature 14 + Clitic Rules	0.7884	0.3642	0.4983	0.8149	0.5894	0.6841
Feature 14 + Phrase-based MT	0.7296	0.5043	0.5964	0.7797	0.6397	0.7028
Feature 14 + Clitic Rules + Phrase-based MT	0.7571	0.5389	**0.6296**	0.8220	0.6850	**0.7473**
	Test					
Feature 14 + Clitic Rules + Phrase-based MT	0.7797	0.5635	**0.6542**	0.7438	0.6855	**0.7135**

Table 1: System results on the dev set (upper part) and on the test set (lower part).

Attach clitic to...	**Clitics**
Beginning of next word	{و ,ال ,ب ,ف ,س}
End of previous word	{ك ,ها ,نا ,ني ,ي ,كم ,ا}

Table 2: Clitics handled by the rule-based module.

instance of that character (e.g. !!!!!!! would be replaced with !).

Statistical Phrase-based Model

We use the Moses toolkit (Koehn et al., 2007) to create a statistical phrase-based machine translation model built on the best pre-processed data, as described above. We treat this last step as a translation problem, where the source language is pre-processed incorrect Arabic text, and the reference is correct Arabic. Feature 14 extraction, rule-based correction, and character de-duplication are applied to both the train and dev sets. All but the last 1,000 sentences of the train data are used at the training set for the phrase-based model, the last 1,000 sentences of the train data are used as a tuning set, and the dev set is used for testing and evaluation. We use fast_align, the aligner included with the cdec decoder (Dyer et al., 2010) as the word aligner with grow-diag as the symmetrization heuristic (Och and Ney, 2003), and build a 5-gram language model from the correct Arabic training data with KenLM (Heafield et al., 2013). The system is evaluated with BLEU (Papineni et al., 2002) and then scored for precision, recall, and F1 measure against the dev set reference.

We tested several different reordering window sizes since this is not a standard translation task, so we may want shorter distance reordering. Although 7 is the default size, we tested 7, 5, 4, 3, and 0, and found that a window of size 4 produces the best result according to BLEU score and F1 measure.

4 Experiments and Results

We train and evaluate our system with the training and development datasets provided for the shared task and the m2Scorer (Dahlmeier and Ng, 2012b). These datasets are extracted from the QALB corpus of human-edited Arabic text produced by native speakers, non-native speakers and machines (Zaghouani et al., 2014).

We conducted a small scale statistical study on the 950K tokens training set used to build our system. We realized that 306K tokens are affected by a correction action which could be a word edit, insertion, deletion, split or merge. 169K tokens were edited to correct the spelling errors and 99K tokens were inserted (mostly punctuation marks). Furthermore, there is a total of 6,7K non necessary tokens deleted and 10.6K attached tokens split and 18.2 tokens merged. Finally, there are only 427 tokens moved in the sentence and 1563 multiple correction action.

We experiment with different configurations and reach the sweet spot of performance when combining the different modules.

4.1 Results

To evaluate the performance of our system on the development data, we compare its output to the reference (gold annotation). We then compute the usual measures of precision, recall and f-measure. Results for various system configurations on the dev and test sets are given in Table 1. Using the baseline system consisting in replacing words by their non diacritized form (Feature 14), we could correct 51.9% of the errors occurring in the dev set, when punctuation is not considered. This result drops when we consider the punctuation errors which seem to be more complex to correct: Only 32.1% of the errors are corrected in the dev set. It is important to notice that adding the clitic rules to the Feature 14 baseline yields an improvement of + 5.15 in F-measure. We reach the best F-measure value when using the phrase-based MT system after pre-processing the data and applying the Feature 14 and clitic rules. Using this combination we were able to correct 68.5% of the errors (excluding punctuation) on the development set with a precision of 82.2% and 74.38% on the test set. When we consider the punctuation, 53.89% of the errors of different types were corrected on the dev set and 56.35% on the test set with a precision of 75.71% and 77.97%, respectively.

5 Error Analysis and Discussion

When building error correction systems, minimizing the number of cases where correct words are marked as incorrect is often regarded as more important than covering a high number of errors. Therefore, a higher precision is often preferred over higher recall. In order to understand what was affecting the performance, we took a closer look at our system output and translation tables to present some samples of errors that our system makes on development set.

5.1 Out-of-vocabulary Words

This category includes words that are not seen by our system during the training which is a common problem in machine translation systems. In our system, most of out-of-vocabulary words were directly transferred unchanged from source to target. For example the word افلمسؤولية was not corrected to المسؤولية.

5.2 Unnecessary Edits

In some cases, our system made some superfluous edits such as adding the definite article in cases where it is not required such as :

Source	أطياف المدينة
Hypothesis	**الأطياف** المدينة
Reference	(unchanged) أطياف المدينة

Table 3: An example of an unnecessary addition of the definite article.

5.3 Number Normalization

We observed that in some cases, the system did not normalize the numbers such as in the following case which requires some knowledge of the real context to understand that these numbers require normalization.

Source	450000 ميغاوات
Hypothesis	450**000** ميغاوات
Reference	450 ميغاوات

Table 4: An example of number normalization.

5.4 Hamza Spelling

Even though our system corrected most of the Hamza spelling errors, we noticed that in certain cases they were not corrected, especially when the words without the Hamza were valid entries in the dictionary. These cases are not always easy to handle since only context and semantic rules can handle them.

5.5 Grammatical Errors

In our error analysis we encountered many cases of uncorrected grammatical errors. The most frequent type

Source	واد الوطنية
Hypothesis	**واد** الوطنية
Reference	وأد الوطنية

Table 5: A sentence where the Hamza was not added above the Alif in the first word because both versions are valid dictionary entries.

is the case endings correction such as correcting the verbs in jussive mode when there is a prohibition particle (negative imperative) like the (لا) in the following examples :

Source	لا يضربوا على أياديهم
Hypothesis	لا **يضربوا** على أياديهم
Reference	لا يضربون على أياديهم

Table 6: An example of a grammatical error.

5.6 Unnecessary Word Deletion

According to the QALB annotation guidelines, extra words causing semantic ambiguity in the sentence should be deleted. The decision to delete a given word is usually based on the meaning and the understanding of the human annotator, unfortunately this kind of errors is very hard to process and our system was not able to delete most of the unnecessary words.

Source	هل سنشهد وضعا أيديهما مشينا آخر
Hypothesis	هل سنشهد وضعا **أيديهما** مشينا آخر
Reference	هل سنشهد وضعا مشينا آخر

Table 7: An example of word deletion.

5.7 Adding Extra Words

Our analysis revealed cases of extra words introduced to some sentences, despite the fact that the words added are coherent with the context and could even improve the overall readability of the sentence, they are uncredited correction since they are not included in the gold standard. For example :

Source	ضرب سمعة الجيش السوري
Hypothesis	ضرب سمعة الجيش السوري **الحر**
Reference	ضرب سمعة الجيش السوري

Table 8: An example of the addition of extra words.

5.8 Merge and Split Errors

In this category, we show some sample errors of necessary word splits and merge not done by our system. The

word خصوصابعد should have been split as خصوصا بعد and the word لا بد should have been merged to appear as one word as in لابد.

5.9 Dialectal Correction Errors

Dialectal words are usually converted to their Modern Standard Arabic (MSA) equivalent in the QALB corpus, since dialectal words are rare, our system is unable to detect and translate the dialectal words to the MSA as in the expression مب زين that is translated in the gold standard to غير زين.

6 Conclusion

We presented our CMUQ system for automatic Arabic text correction. Our system combines rule-based linguistic techniques with statistical language modeling techniques and a phrase-based machine translation method. We experiment with different configurations. Our experiments have shown that the system we submitted outperforms the baseline and we reach an F-score of 74.73% on the development set from the QALB corpus when punctuation is excluded, and 65.42% on the test set when we consider the punctuation errors . This placed us in the 3rd rank. We believe that our system could be improved in numerous ways. In the future, we plan to finalize a current module that we are developing to deal with merge and split errors in a more specific way. We also want to focus in a deeper way on the word movement as well as punctuation problems, which can produce a more accurate system. We will focus as well on learning further error correction models from Arabic Wikipedia revision history, as it contains natural rewritings including spelling corrections and other local text transformations.

Acknowledgements

This publication was made possible by grants NPRP-09-1140-1-177 and NPRP-4-1058- 1-168 from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

Mohamed I. Alkanhal, Mohamed Al-Badrashiny, Mansour M. Alghamdi, and Abdulaziz O. Al-Qabbany. 2012. Automatic Stochastic Arabic Spelling Correction With Emphasis on Space Insertions and Deletions. *IEEE Transactions on Audio, Speech & Language Processing*, 20(7):2111–2122.

Mohammed Attia, Pavel Pecina, Younes Samih, Khaled Shaalan, and Josef van Genabith. 2012. Improved Spelling Error Detection and Correction for Arabic. In *Proceedings of COLING 2012: Posters*, pages 103–112, Mumbai, India.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A Beam-Search Decoder for Grammatical Error Correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578, Jeju Island, Korea.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better Evaluation for Grammatical Error Correction. In *NAACL HLT '12 Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572.

Robert Dale. 1997. Computer Assistance in Text Creation and Editing. In *Survey of the state of the art in Human Language Technology*, chapter 7, pages 235–237. Cambridge University Press.

Chris Dyer, Jonathan Weese, Hendra Setiawan, Adam Lopez, Ferhan Ture, Vladimir Eidelman, Juri Ganitkevitch, Phil Blunsom, and Philip Resnik. 2010. cdec: A Decoder, Alignment, and Learning Framework for Finite-state and Context-free Translation Models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden.

A. R. Golding and D. Roth. 1999. A Winnow Based Approach to Context-Sensitive Spelling Correction. *Machine Learning*, 34(1-3):107–130.

Nizar Habash. 2008. Four Techniques for Online Handling of Out-of-Vocabulary Words in Arabic-English Statistical Machine Translation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 57–60, Columbus, Ohio.

Bassam Haddad and Mustafa Yaseen. 2007. Detection and Correction of Non-words in Arabic: a Hybrid Approach. *International Journal of Computer Processing of Oriental Languages*, 20(04):237–257.

Ahmed Hassan, Sara Noeman, and Hany Hassan. 2008. Language Independent Text Correction using Finite State Automata. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP 2008)*, pages 913–918, Hyderabad, India.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable Modfied Kneser-Ney Language Model Estimation. In *In Proceedings of the Association for Computational Linguistics*, Sofia, Bulgaria.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Christopher Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys (CSUR)*, 24(4):377–439.

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

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. In *Computational Linguistics*, page 1951.

Kemal Oflazer. 1996. Error-Tolerant Finite-State Recognition with Applications to Morphological Analysis and Spelling Correction. *Computational Linguistics*, 22(1):73–89.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the Association for Computational Linguistics*, Philadelphia, Pennsylvania.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, 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'14)*, pages 1094–1101, Reykjavik, Iceland.

Khaled Shaalan, Amin Allam, and Abdallah Gomah. 2003. Towards Automatic Spell Checking for Arabic. In *Proceedings of the 4th Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE)*, Cairo, Egypt.

Khaled Shaalan, Rana Aref, and Aly Fahmy. 2010. An Approach for Analyzing and Correcting Spelling Errors for Non-native Arabic Learners. In *Proceedings of The 7th International Conference on Informatics and Systems, INFOS2010, the special track on Natural Language Processing and Knowledge Mining*, pages 28–30, Cairo, Egypt.

Khaled Shaalan, Mohammed Attia, Pavel Pecina, Younes Samih, and Josef van Genabith. 2012. Arabic Word Generation and Modelling for Spell Checking. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 719–725, Istanbul, Turkey.

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.

Chiraz 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*, pages 770–777, Oxford, UK.