Automatic Correction of Arabic Text: a Cascaded Approach

Hamdy Mubarak, Kareem Darwish
Qatar Computing Research Institute
Qatar Foundation
{hmubarak,kdarwish}@qf.org.qa

Abstract

This paper describes the error correction model that we used for the Automatic Correction of Arabic Text shared task. We employed two correction models, namely a character-level model and a case-specific model, and two punctuation recovery models, namely a simple statistical model and a CRF model. Our results on the development set suggest that using a cascaded correction model yields the best results.

1 Introduction

In This paper, we describe our system for automatic Arabic error correction shared task (QALB-2014 Shared Task on Automatic Correction of Arabic) as part of the Arabic NLP workshop (Mohit et al., 2014). Our system is composed of two main steps. The first involves correcting word level errors, and the second pertains to performing punctuation recovery. For word level correction, we used two approaches, namely: 1) a statistical character level transformation model that is aided by a language model (LM) to handle letter insertions, deletions, and substitutions and word merges; and 2) a case-specific system that is aided by a LM to handle specific error types such as dialectal word substitutions and word splits. For punctuation recovery, we used two approaches, namely a simple statistical word-based system, and a conditional random fields (CRF) sequence labeler (Lafferty et al., 2001) that attempts to recover punctuation based on POS and word sequences. We performed all experiments on the QALB dataset (Zaghouani et al., 2014).

2 Word Error Correction

In this section we describe two approaches for word correction. The first approach involves using a character level model, and the second handles specific correction cases.

2.1 Character-level Correction Model

For the character level model, we treated correction as a Transliteration Mining (TM) task. In TM, a sequence in a source alphabet is used to find the most similar sequence in a lexicon that is written in a target alphabet. TM has been fairly well studied with multiple evaluation campaigns such as the Named Entities Workshop (NEWS) (Zhang et al., 2011; Zhang et al., 2012). In our work, we adopted a TM system to find corrections appearing in a large Arabic corpus. The system involved learning character (or character-sequence) level mappings between erroneous words and their correct counterparts. Given the character mappings between the erroneous and correct words, we used a generative model that attempts to generate all possible mappings of a source word while restricting the output to words in the target language (El-Kahki et al., 2011; Noeman and Madkour, 2010). Specifically, we used the baseline system of El-Kahky et al. (2011). To train character-level mappings, we extracted all the parallel word-pairs in the original (uncorrected) and corrected versions in the training set. If a word in the original version of the training set was actually correct, the word would be mapped to itself. We then aligned the parallel word pairs at character level using GIZA++ (Och and Ney, 2003), and symmetrized the alignments using grow-diag-

final-and heuristic (Koehn et al., 2007). In all, we aligned a little over one million word pairs. As in the baseline of El-Kahki et al. (2011), given a possibly misspelled word w_{org}, we produced all its possible segmentations along with their associated mappings that we learned during alignment. Valid target sequences were retained and sorted by the product of the constituent mapping probabilities. The top n (we picked n = 10) candidates, $w_{trg_{1..n}}$ with the highest probability were generated. Using Bayes rule, we computed:

$$\underset{w_{trg_{i \in 1..n}}}{argmax}\, p(w_{trg_i}|w_{org}) = p(w_{org}|w_{trg_i})p(w_{trg_i}) \quad (1)$$

where $p(w_{org}|w_{trg_i})$ is the posterior probability of mapping, which is computed as the product of the mappings required to generate w_{org} from w_{trg_i}, and $p(w_{trg_i})$ is the prior probability of the word. Then we used a trigram LM to pick the most likely candidate in context. We used a linear combination of the the character-level transformation probability and the LM probability using the following formula: $score = \lambda log(Prob_{LM}) + (1 - \lambda) log(Prob_{char})$ We built the lexicon from a set of 234,638 Aljazeera articles[1] that span 10 years and all of Arabic Wikipedia. We also built a trigram language model on the same corpus. The combined corpus contains 576 million tokens including 1.6 million unique ones. Spelling mistakes in Aljazeera articles (Mubarak et al., 2010) and Wikipedia were infrequent.

We varied the value of λ between 0 and 1 with increments of 0.1 and found that the values 0.6 and 0.7 yielded the best results. This indicates that LM probability is more important than character-mapping probability.

2.2 Case-specific Correction

In this method we attempted to address specific types of errors that are potentially difficult for the character-based model to handle. Some of these errors include dialectal words and words that were erroneously split. Before applying any correction, we consulted a bigram LM that was trained the aforementioned set of Aljazeera articles. The following cases are handled (in order):

• Switching from English punctuations to Arabic ones, namely changing: "?" → "؟" and "," → "،".

• Handling common dialectal words and common word-level mistakes. An example dialectal word is اللي (Ally)[2] (meaning "this" or "that") which could be mapped to الذي (Al∗y) , التي (Alty) or الذين (Al∗yn). An example of a common mistake is انشاء الله (An$A' Allh) (meaning "God willing") which is corrected to إن شاء الله (>n $A' Allh). The sentence is scored with and without the word replacement, and the replacement is done if it yields higher LM probability.

• Handling errors pertaining to the different forms of *alef*, *alef maqsoura* and *ya*, and *ta marbouta* and *ha* (Nizar Habash, 2010). We reimplemented the baseline system in (Moussa et al., 2012) where words are normalized and the different possible denormalized forms are scored in context using the LM. We also added the following cases, namely attempting to replace: ؤ (&) with ؤو (&w) or ئو (}w); and ئ (}) with يء (y') or vice versa (ex: مرؤس (mr&s) → مرؤوس (mr&ws)).

• Handling merges and splits. Often words are concatenated erroneously. Thus, we attempted to split all words that were at least 5 letters long after letters that don't change their shapes when they are connected to the letters following them, namely different alef forms, د (d), ذ (*), ر (r), ز (z), و (w), ة (p), and ى (Y) (ex: ياربنا (yArbnA) → يا ربنا (yA rbnA)). If the bigram was observed in the LM and the LM score was higher (in context) than when they were concatenated, then the word was split. Conversely, some words were split in the middle. We attempted to merge every two words in sequence. If the LM score was higher (in context) after the merge, then the two words would be merged (ex:

[1] http://www.aljazeera.net

[2] Buckwalter transiteration

انتصار ات (AntSAr At) → انتصارات (AntSArAt)).

• Removing repeated letters. Often people repeat letters, particularly long vowels, for emphasis as in أخيييييرااا (>xyyyyyrAAA) (meaning "at last"). We corrected for elongation in a manner similar to that of Darwish et al. (Darwish et al., 2012). When a long vowel are repeated, we replaced it with a either the vowel (ex. أخيرا) (>xyrA) or the vowel with one repetition (ex. أخييرا) (>xyyrA) and scored using the LM. If a repeated *alef* appeared in the beginning of the word, we attempted to replace it with alef lam (ex. االحضارة (AAHDArp) → الحضارة (AlHDArp) (meaning "civilization")). A trailing alef-hamza-alef sequence was replaced by alef-hamza (ex. سماء ا (smA'A) → سماء (smA') (meaning "sky")).

• Correcting out-of-vocabulary words. For words that were not observed in the LM, we attempted the following corrections: 1) replacing phonetically or visually confusable letters, namely ض (D) and ظ (Z), د (d) and ذ (*), and ذ (*) and ز; (z) (ex: ظابط (ZAbT) → ضابط (DAbT)) 2) removing the letters ب (b) and د (d) that are added to verbs in present tense in some dialects (ex: بيكتب (byktb) → يكتب (yktb)); 3) replacing the letters ح (H) and ه (h), which are added in some dialects to indicate future tense, with س (s) (ex: حيشرب (Hy$rb) → سيشرب (sy$rb)); and 4) replacing a leading هال (hAl) with either هذا ال (h*A Al) or هذه ال (h*h Al) (ex. هالكتاب (hAlktAb) → هذا الكتاب (h*A AlktAb)) and the leading عال (EAl) with على ال (ElY Al) (ex. عالأرض (EAl>rD) → على الأرض (ElY Al>rD)). After replacement, the LM was always consulted.

2.3 Correction Results

Table 1 reports on the results of performing both correction methods on the development set. Also, since the case-specific corrections handle cases that were not handled by the character-level model, we attempted to cascade both methods together. It seems that when applying the character-level model first followed by the case-specific correction yielded the best results.

Method	F-measure
Character-level	0.574
Case-specific	0.587
Character-level → Case-specific	0.615
Case-specific → Character-level	0.603

Table 1: The correction results using the character-level model, case-specific correction, or their cascades.

3 Punctuation Recovery

In this section, we describe two methods for punctuation recovery. The first is a simple word-based model and the other is a CRF based model.

3.1 Simple Statistical Model

In this approach, we identified words that were preceded or followed by punctuations in the training set. If a word was preceded or followed by a particular punctuation mark more than 40% of the time, then we automatically placed the punctuation before or after the word in the dev set. Also, if a sentence did not have a period at the end of it, we added a period.

3.2 CRF Model

In this approach we trained a CRF sequence labeling to attempt to recover punctuation. CRF combines state and transition level features making it a possibly better choice than an HMM or a simple classifier. We used the CRF++ implementation[3] of the sequence labeler. We trained the labeler on the training part of the QALB dataset. We used the following features:

Word features: the current word, the previous and next words, and the two previous and two next words.

Part-of-speech (POS) tags: the POS of the current

[3] http://crfpp.googlecode.com/svn/trunk/doc/index.html

Method	Precision	Recall	F-measure
Stat model	0.306	0.153	0.204
CRF model	0.373	0.141	0.204

Table 2: The punctuation recovery results using the simple statistical model and the CRF model.

Method	F-measure
Stat model (before correction)	0.593
Stat model (after correction)	0.614
CRF model (before correction)	0.607
CRF model (after correction)	0.615

Table 3: Cascaded correction (Character-level → Case-specific) combined with punctuation recovery.

word and the POS of the two previous and two following words.

3.3 Punctuation Recovery Results

Table 2 reports on the results of using the two different methods for punctuation recovery. Note that no other correction is applied.

4 Combining Correction with Punctuation Recovery

Given that cascading both correction models yielded the best results, we attempted to combine the cascaded correction model with the two punctuation recovery methods. We tried to put punctuation recovery before and after correction. Table 3 summarizes the results. As the results suggest, combining correction with punctuation recovery had a negative effect on overall F-measure. This requires further investigation.

5 Official Shared Task Experiments and Results

For the official submissions to the shared task, we submitted 3 runs as follows:

1. QCRI-1: character-level correction, then case-based correction.
2. QCRI-2: case-based correction, then statistical punctuation recovery
3. QCRI-3: exactly like 2, but preceded also by statistical punctuation recovery

Run	Precision	Recall	F-measure
QCRI-1	0.717	0.5686	0.6343
QCRI-2	0.6286	0.6032	0.6157
QCRI-3	0.6066	0.5928	0.5996

Table 4: Official Results.

Table 4 reports on the officially submitted results against the test set. It seems that our attempts to add punctuation recovery worsened results.

6 Conclusion

In this paper, we presented automatic approaches for correcting Arabic text and punctuation recovery. Our results on the development set shows that using a cascaded approach that involves a character-level model and another model that handles specific errors yields the best results. Incorporating punctuation recovery did not improve correction.

References

Kareem Darwish, Walid Magdy, and Ahmed Mourad. 2012. Language processing for arabic microblog retrieval. Proceedings of the 21st ACM international conference on Information and knowledge management. ACM, 2012.

Ali El-Kahky, Kareem Darwish, Ahmed Saad Aldein, Mohamed Abd El-Wahab, Ahmed Hefny, and Waleed Ammar. 2001. Improved transliteration mining using graph reinforcement. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1384-1393, 2011.

Nizar Habash. 2010. Introduction to Arabic natural language processing. Synthesis Lectures on Human Language Technologies 3.1 (2010): 1-187

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst, Moses: Open Source Toolkit for Statistical Machine Translation, Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session, Prague, Czech Republic, June 2007.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data, In Proc. of ICML, pp.282-289, 2001.

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.

Mohammed Moussa, Mohamed Waleed Fakhr, and Kareem Darwish. 2012. Statistical denormalization for Arabic Text. In Empirical Methods in Natural Language Processing, pp. 228. 2012.

Hamdy Mubarak, Ahmed Metwali, Mostafa Ramadan. 2010. Spelling Mistakes in Arabic Newspapers. Arabic Language and Scientific Researches conference, Faculty of Arts, Ain Shams University, Cairo, Egypt

Sara Noeman and Amgad Madkour. 2010. Language Independent Transliteration Mining System Using Finite State Automata Framework. ACL NEWS workshop 2010.

Franz J. Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, Vol. 1(29), 2003.

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 (LREC14), Reykjavik, Iceland.

Min Zhang, A Kumaran, Haizhou Li. 2011. Whitepaper of NEWS 2012 Shared Task on Machine Transliteration. IJCNLP-2011 NEWS workshop.

Min Zhang, Haizhou Li, Ming Liu, A Kumaran. 2012. Whitepaper of NEWS 2012 Shared Task on Machine Transliteration. ACL-2012 NEWS workshop.