GWU-HASP-2015@QALB-2015 Shared Task: Priming Spelling Candidates with Probability[1]

Mohammed Attia, Mohamed Al-Badrashiny, Mona Diab
Department of Computer Science
The George Washington University
{Mohattia;badrashiny;mtdiab}@gwu.edu

Abstract

In this paper, we describe our system HASP-2015 (Hybrid Arabic Spelling and Punctuation Corrector) in which we introduce significant improvements over our previous version HASP-2014 and with which we participated in the QALB-2015 Second Shared Task on Arabic Error Correction. Our system utilizes probabilistic information on errors and their possible corrections in the training data and combine that with an open-source reference dictionary (or word list) for detecting errors and generating and filtering candidates. We enhance our system further by allowing it to generate candidates for common semantic and grammatical errors. Eventually, an n-gram language model is used for selecting best candidates. We use a CRF (Conditional Random Fields) classifier for correcting punctuation errors in a two-pass process where first the system learns punctuation placement, and then it learns to identify punctuation types.

1 Introduction

In this paper we describe our system for Arabic spelling error detection and correction, HASP-2015 (Hybrid Arabic Spelling and Punctuation Corrector). We introduce significant improvements to our previous version HASP-2014 (Attia et al., 2014). We participate with HASP-2015 in the QALB-2015 Second Shared Task on Arabic Error Correction (Rozovskaya et al., 2015).

The problem of Arabic spelling error correction has been investigated in a number of papers (Haddad and Yaseen, 2007; Alfaifi and Atwell, 2012; Hassan et al., 2008; Attia et al., 2012; Alkanhal et al., 2012). Significant contributions were also introduced in the 2014 Shared Task on Arabic Error Correction (Mohit et al., 2014) including (Rozovskaya et al., 2014; Nawar and Ragheb, 2014; Jeblee et al., 2014; and Mubarak and Darwish, 2014).

The QALB-2015 shared task is an extension of the first QALB shared task (Mohit et al., 2014) that took place in 2014. 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 are scored on blind test sets Alj-test-2015 and L2-test-2015. Our system is ranked third and fourth on the Alj and L2, respectively.

The shared task data deals with "errors" in the general sense which comprise: a) punctuation errors; b) non-word errors; c) real-word spelling errors; d) grammatical errors (related to case, number and gender); and, e) affective variations such as elongation (kashida) and speech effects such as character multiplication for emphasis. Our previous system, HASP-2014, handles only types (a), (b), and (e) errors. We extend our systemt HASP-2015 to provide coverage for and address types (d) and (e) spelling errors.

[1] This work was supported by the Defense Advanced Research Projects Agency (DARPA) Contract No. HR0011-12-C-0014, BOLT program with subcontract from Raytheon BBN.

Proceedings of the Second Workshop on Arabic Natural Language Processing, pages 138–143,
Beijing, China, July 26-31, 2015.

2 Our Methodology

Our system uses a pipeline of four components: 1) regular expression normalization for deterministic errors, 2) A discriminative classifier for punctuation errors, 3) Spelling detection and handling, and, 4) Post-processing for fixing common system errors.

For punctuation errors, we use a classifier in a two-pass process where first the system learns punctuation placement, and then it learns to identify punctuation types. The reason for this staging is that learning six punctuation types at once could be problematic for the classifier, and we hypothesize that splitting the task of placement from identification, where in the first step it makes a binary decision of whether or not to insert a punctuation mark, and in the second step it predicts the type of that punctuation mark.

In HASP-2014, we only rely on a reference dictionary (or word list) for detecting errors and generating candidates. The candidates were generated according to the edit distance between the erroneous word and possible candidates.

In HASP-2015, we generate probabilistic information from the training data on errors and their possible corrections and utilize this information in detecting errors and generating candidates. The reference dictionary is relegated to as a back-off function when no probabilistic information is available in the training data. Our system is able to detect and generate candidates for common semantic and grammatical errors. Candidates and their probabilistic scores are passed an n-gram language model for selecting best candidates. Our system is explained in detail in the next section.

For organizational purposes, we divide errors into two types: a) nonverbal errors which include affective variations, punctuation, word merges and word splits; and b) verbal errors, which include non-word error, real-word error, grammatical errors, and dialectal words/expressions. In other words, verbal errors are related to the alphabetical buildup of words, and non-verbal errors go beyond this alphabetical buildup.

3 Nonverbal Errors

Nonverbal errors include affective variations, punctuation errors, word merges and word splits.

3.1 Affective Variations

There are many instances in the shared task's data that can be treated using simple and straightforward conversion via regular expression replace rules. We estimate that these instances cover 10% of the non-punctuation errors in the development set. In HASP, we use deterministic heuristic rules to normalize the text, including the removal of speech effects, such as الرجاااال AlrjAAAAl 'men' which is converted to الرجال Alrjal, the removal of decorative kashida, e.g. دمـــاء dm__A' 'blood', and the conversion of Hindi digits (٠١٢٣٤٥٦٧٨٩) into Arabic digits [0-9].

3.2 Punctuation Errors

Punctuation errors constitute 40% of the errors in the QALB Arabic data. In HASP-2015, we continue to handle the six basic punctuation marks: comma, colon, semi-colon, exclamation mark, question mark, and period.

For classification, we use a Conditional Random Field, CRF++ classifier (Lafferty et al. 2001) with window size 5. The features we use are extracted from the 'column' file in the QALB shared task data, which includes preprocessing with MADAMIRA morphological disambiguator (Pasha et al., 2014). In HASP-2015, we split the task of the classifier into two subtasks: placement and identification.

Experiment	R	P	F
Baseline	45.70	76.01	57.08
Pass_II + Alj_Training	52.11	72.33	60.58
Pass_II + Merge_Training	**52.17**	**72.38**	**60.63**

Table 1. CRF Pass II results for Alj

Experiment	R	P	F
Baseline	13.87	20.57	16.57
Pass_II + Alj_Training	37.38	30.53	33.61
Pass_II + Merge_Training	**33.98**	**33.73**	**33.86**

Table 2. CRF Pass II results for L2

Pass I: Placement

The placement subtask is a binary classification task where the classifier decides whether a punctuation mark (regardless of the type) should be included or not. We use five features in this process:

(1) The original word, that is the word as it appears in the text without any further processing, (e.g., للتشاور llt$Awr 'for consulting');
(2) Stem. We use the Penn Arabic Treebank (PATB) tokenization (e.g., التشاور+ل l+Alt$Awr) and strip off the clitics (e.g., التشاور Alt$Awr);
(3) Kulick (Kulick et al., 2011) POS tag (e.g., IN+DT+NN);
(4) Buckwalter POS tag (e.g., PREP+DET+ NOUN+CASE_DEF_GN) as produced by MADAMIRA;
(5) Classes to be predicted: punc_after and NA.

Pass II: Identification

This stage uses the same set of features of the placement stage in addition to its output to determine the type of punctuation mark to be placed. The predicted class is one of the following seven: colon_after, comma_after, exclmark_after, period_after, qmark_after, semicolon_after, and NA.

This two-pass process shows significant improvement over the baseline for Alj and L2 data as illustrated in Table 1 and 2.

2.3 Word Merges

Merged words are when the space(s) between two or more words is deleted, such as هذاالنظام h*AAlnZAm 'this system', which should be هذا النظام h*A AlnZAm. These errors constitute 3.67% and 3.48% of the error types in the shared task's development and training data, respectively. We use Attia et al.'s (2012) algorithm for dealing with merged words, $l-3$, where l is word length.

Moreover, we found out that common merge errors and their correction can conveniently be learned from the training data, leading to significant improvement as shown in the final results. Here are some examples of frequent merge errors:

- yArb يارب "O Lord" → yA rb
- EbdAllh عبدالله "Abdullah" → Ebd Allh

2.4 Word Splits

Beside the problem of merged words, there is also the problem of split words, where one or more spaces are inserted within a word, such as صم ام Sm Am 'valve' (the correct form is صمام SmAm). This error constitutes 6% of the shared task's found in the training and development sets. We found that the vast majority of instances of this type of error involve the clitic conjunction *waw* "and", which should be represented as a word prefix. Therefore, we opted to handle this problem in our work in a partial and shallow manner using deterministic rules by the reattachment of the separated conjunction morpheme *waw* و w "and" to the succeeding word.

4 Verbal Errors

Verbal errors include non-word errors, real-word errors, grammatical errors, and dialectal words/expressions.

4.1 Error Detection

The method for detecting spelling errors have usually varied according to the type of error. A non-word spelling error is typically defined as (adapted from Brill, and Moore, 2000): given an alphabet Σ, a reference dictionary D consisting of strings in Σ^*, a given word is a spelling error s if $s \in \Sigma^*$ and $s \notin D$.

For real-word errors, a reference dictionary will not help, as both the error and the correction are valid words in isolation. Instead, a language model, for example, is used to estimate the likelihood of words in a certain context, and words that fall below a certain threshold are considered as a possible error. POS bigrams and tri-grams have also been used for that purpose (Kukich, 1992).

We employ a single algorithm to detect all types of spelling errors, whether non-word, semantic, grammatical or dialectal. Our algorithm for error detection is to find words in the training data where $n(P(s \mid c)) > P(s \mid s')$, where s is a spelling error, c is the correction, n is a threshold and s` is s considered as a candidate. This translates to the probability of c given s times n is greater than the probability of s' given s. In our system, we set the threshold $n = 2$ which effectively mean that a semantic error is only considered when the probability of the correction is more than half the probability of the reference word. The threshold estimation is an empirical question determined by the robustness of the language model and the quantity of noise in the training data.

In HASP-2015, the reference dictionary is not totally discarded, but used as a back-off resource to cover instances not included in the training data. We use AraComLex Extended, an open-source reference dictionary (or word list) of 9.2M full-formed words (Attia et al., 2012) as our backup reference dictionary.

4.2 Candidate Generation

Correcting spelling errors is ideally treated as a probabilistic problem formulated as (Kernigan, 1990; Norvig, 2009; Brill, and Moore, 2000):

$$argmax_c\ P(s \mid c)\ P(c)$$

Here $P(c)$ is the probability that c is the correct word (or the language model), and $P(s \mid c)$ is the probability that s is typed when c is intended (the error model or noisy channel model), $argmax_c$ is the scoring mechanism that computes the correction c that maximizes the probability.

In HASP-2014, we ranked candidates according to their edit distance score using the finite state compiler, foma (Hulden, 2009), but in HASP-2015, we rank candidates according to their probability, $(s \mid c)$, as derived from the training data, and we pass candidates along with their probability scores to the language model. Again, the edit distance candidates and their ranking are used when no probability information is available from the training data. The following are some illustrative examples of the statistical information extracted from the training data for the various error types.

Non-word errors:
An ان "that" >n#7781; <n#1485; |n#29
AlA الا "but" <lA#1442; >lA#225

Semantic errors:
Alhm الهم "worry" Alhm#20; Allhm#17
Ely علي "on" ElY#818; Ely#318

Grammatical errors:
mjrmyn مجرمين "criminals" mjrmyn#31; mjrmwn#16
lyl ليل "night" lyl#34; lylA#16

Dialectal words:
bs بس "but" lkn#67; fqT#27
AHnA احنا "we" nHn#65; >HnA#9

Additionally, we use some generic rules to generate candidates for possible dialectal errors:

- Add *A* after final *w* as in آمنو |*manuw* "they believe",
- Remove the colloquial aspectual clitic particle *b* before the perfective initials *n, y, t.*

5 Error Correction and Final Results

For error correction, namely selecting the best solution among the list of candidates, we use an n-gram language model (LM), as implemented in the SRILM package (Stolcke et al., 2011). We use the 'disambig' tool for selecting candidates from a map file where erroneous words are provided with a list of possible corrections. We also use the 'ngram' utility in post-processing for deciding on whether a split-word solution has a better probability than a single word solution. Our tri-gram language model is trained on the Arabic Gigaword Corpus, 5th edition (Parker et al., 2011) and a corpus crawled from Al-Jazeera (Attia et al.; 2012).

For the LM disambiguation we use the '-fb' option (forward-backward tracking), and we provide candidates with probability scores collected from the QALB training data. Both of the forward-backward tracking and the probability scores in tandem yield better results than the default values. We evaluate the performance of our system against the gold standard using the *Max-Match* (M^2) method for evaluating grammatical error correction by Dahlmeier and Ng (2012).

Our best f-score is obtained by priming candidates from the training data, adding Al-Jazeera corpus to Gigaword 5, and using the two-pass CRF punctuation prediction. Table 3 and 4 show the results on Alj and L2 development sets respectively. Table 5 shows the results on Alj and L2 test sets.

#	Experiment	R	P	F
1	Baseline (HASP'14)	52.98	75.47	62.25
2	Prime non-word candidates from the training set	55.26	77.40	64.48
3	Include real-word candidates from the training data	57.87	77.03	66.09
4	Prime merge errors from the training set	58.67	77.70	66.86
5	Post-processing	58.80	77.83	66.99
6	Two-pass punctuation correction	60.40	76.57	67.53
7	3 gram LM and adding Al-Jazeera corpus to Gigaword	**60.59**	**76.65**	**67.68**

Table 3. Results for Alj-test-2014 (dev set)

#	Experiment	R	P	F
1	Baseline	22.27	56.80	31.99
2	3 gram LM and adding Al-Jazeera corpus to Gigaword	**22.35**	**57.17**	**32.14**

Table 4. System results for L2-dev-2015

#	Experiment	R	P	F
1	Alj-test-2015	67.51	74.69	70.92
2	L2-test-2015	23.32	55.66	32.87

Table 5. System results for the test sets

For the baseline, we use the older version of our system (HASP-2014), and the results show significant improvement in performance. The biggest two gains in performance, as shown in Table 3, came from experiments 2 and 3 when candidates and their probabilities were extracted from the training data and used to supplement candidates generated from the reference dictionary using edit distance. Experiment 3, i.e. using real-word candidate allowed our system to handle semantic and grammatical errors, a domain which was beyond the scope of the previous version. Dialectal errors were included in Experiment 2 dealing with non-word candidates. It is to be noted the system can benefit from a larger training set if that becomes available in the future.

The slight improvements gained by experiments 4 through 7 are an indication of the dimensions along which future improvements might be achieved. These dimensions include better way of handling merge errors, post-processing for correcting system-specific errors, better handling of punctuation errors, and better selection of data for training the language model.

It is also to be noted that the gold data suffers from instances of inconsistency. For example لابد lAbd "must" is split as two words لا بد lA bd in 64% of the cases, while مازال mAzAl "still" is split in 32% of the cases.

Moreover, while conducting error analysis we found many errors in the manual annotation of the gold development data. For example, اللذي All*y "who" is incorrectly corrected as الذى Al*Y while the correct correction is الذي Al*y and many more errors are not detected at all in the gold data, such as انكم، Ankm "you" and الملتحدة AlmltHdp for المتحدة AlmtHdp "united". In total, we automatically found over 200 errors in the gold development data, but with manual checking it is found that some of the instances are incorrectly reported. However, we assume that more investigation of the consistency and accuracy of the gold data can lead to better performance and better evaluation of the systems participating in the shared task.

6 Conclusion

We have described our system HASP for the automatic correction of spelling and punctuation mistakes in Arabic. To our knowledge, this is the first system to handle punctuation errors. We utilize and improve on an open-source full-form dictionary, introduce a better algorithm for handing merged word errors, tune the LM parameters, and combine the various components together, leading to cumulative improved results.

References

Alkanhal, Mohamed I., Mohamed A. 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, and Language Processing, Vol. 20, No. 7, September 2012.

Attia, Mohammed, Mohamed Al-Badrashiny, Mona Diab. GWU-HASP: Hybrid Arabic Spelling and Punctuation Corrector. Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 148–154, October 25, 2014, Doha, Qatar.

Attia, Mohammed, Pavel Pecina, Younes Samih, Khaled Shaalan, Josef van Genabith. 2012. Improved Spelling Error Detection and Correction for Arabic. COLING 2012, Bumbai, India.

Brill, Eric and Moore, Robert C. (2000) An improved error model for noisy channel spelling correction. Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, pp. 286–293.

Dahlmeier, Daniel and Ng, Hwee Tou. 2012. Better evaluation for grammatical error correction. In Proceedings of NAACL.

Haddad, B., and Yaseen, M. (2007) Detection and Correction of Non-Words in Arabic: A Hybrid Approach. International Journal of Computer Processing of Oriental Languages. Vol. 20, No. 4.

Hassan, A, Noeman, S., and Hassan, H. (2008) Language Independent Text Correction using Finite State Automata. IJCNLP. Hyderabad, India.

Hulden, M. (2009) Foma: a Finite-state compiler and library. EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics Stroudsburg, PA, USA

Jeblee, S., Bouamor, H., Zaghouani, W., and Oflazer, K. 2014. CMUQ@QALB-2014: An SMT-based System for Automatic Arabic Error Correction. In Proceedings of EMNLP Workshop on Arabic Natural Language Processing: QALB Shared Task.

Kulick Seth. Exploiting separation of closed-class categories for Arabic tokenization and part-of-speech tagging. In Graham Katz and Mona Diab, editors, Special Issue on Arabic Computational Linguistics, ACM Transactions on Asian Language Information Processing. 2011.

Kernigan, M., Church, K., Gale W. (1990). A Spelling Correction Program Based on a Noisy Channel Model. AT & T Laboratories, 600 Mountain Ave., Murray Hill, NJ.

Kukich, Karen. (1992) Techniques for automatically correcting words in text. Computing Surveys, 24(4), pp. 377–439.

Lafferty, John, Andrew McCallum, and Fernando Pereira. (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data, In Proceedings of the International Conference on Machine Learning (ICML 2001), MA, USA, pp. 282-289.

Mohit, Behrang, 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.

Mubarak, H. and Darwish, K. 2014. Automatic Correction of Arabic Text: a Cascaded Approach. In Proceedings of EMNLP Workshop on Arabic Natural Language Processing: QALB Shared Task.

Nawar, M. and Ragheb, M. 2014. Fast and Robust Arabic Error Correction System. In Proceedings of EMNLP Workshop on Arabic Natural Language Processing: QALB Shared Task.

Ng, Hwee Tou, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. (2013) The CoNLL-2013 Shared Task on Grammatical Error Correction. Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, pages 1–12, Sofia, Bulgaria, August 8-9 2013.

Norvig, P. (2009) Natural language corpus data. In Beautiful Data, edited by Toby Segaran and Jeff Hammerbacher, pp. 219- "-242. Sebastopol, Calif.: O'Reilly.

Och, Franz Josef, Hermann Ney. (2003) A Systematic Comparison of Various Statistical Alignment Models. In Computational Linguistics, volume 29, number 1, pp. 19-51 March 2003.

Parker, R., Graff, D., Chen, K., Kong, J., and Maeda, K. (2011) Arabic Gigaword Fifth Edition. LDC Catalog No.: LDC2011T11, ISBN: 1-58563-595-2.

Pasha, Arfath, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, Ryan Roth. (2014) MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland.

Rozovskaya, Alla, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid and Behrang Mohit. (2015) The Second QALB Shared Task on Automatic Text Correction for Arabic. Proceedings of ACL Workshop on Arabic Natural Language, Beijing, China.

Rozovskaya, A., Habash, N., Eskander, R., Farra, N., and Salloum, W. (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.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011) SRILM at sixteen: Update and outlook. in Proc. IEEE Automatic Speech Recognition and Understanding Workshop. Waikoloa, Hawaii.

Zaghouani, Wajdi, Habash, Nizar, Bouamor, Houda, Rozovskaya, Alla, Mohit, Behrang, Heider, Abeer and Oflazer, Kemal. 2015. Correction Annotation for Non-Native Arabic Texts: Guidelines and Corpus. Proceedings of The 9th Linguistic Annotation Workshop, Denver, Colorado, USA, pp. 129-139.

Zaghouani, Wajdi, 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.