GWU-HASP: Hybrid Arabic Spelling and Punctuation Corrector[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 Hybrid Arabic Spelling and Punctuation Corrector (HASP). HASP was one of the systems participating in the QALB-2014 Shared Task on Arabic Error Correction. The system uses a CRF (Conditional Random Fields) classifier for correcting punctuation errors, an open-source dictionary (or word list) for detecting errors and generating and filtering candidates, an n-gram language model for selecting the best candidates, and a set of deterministic rules for text normalization (such as removing diacritics and kashida and converting Hindi numbers into Arabic numerals). We also experiment with word alignment for spelling correction at the character level and report some preliminary results.

1 Introduction

In this paper we describe our system for Arabic spelling error detection and correction, Hybrid Arabic Spelling and Punctuation Corrector (HASP). We participate with HASP in the QALB-2014 Shared Task on Arabic Error Correction (Mohit et al., 2014) as part of the Arabic Natural Language Processing Workshop (ANLP) taking place at EMNLP 2014.

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; and, e) orthographical errors such as elongation (kashida) and speech effects such as character multiplication for emphasis. HASP in its current stage only handles types (a), (b), and (e) errors. We assume that the various error types are too distinct to be treated with the same computational technique. Therefore, we treat each problem separately, and for each problem we select the approach that seems most efficient, and ultimately all components are integrated in a single framework.

1.1 Previous Work

Detecting spelling errors in typing is one of the earliest NLP applications, and it has been researched extensively over the years, particularly for English (Damerau, 1964; Church and Gale, 1991; Kukich, 1992; Brill and Moore, 2000; Van Delden et al., 2004; Golding, 1995; Golding and Roth, 1996; Fossati and Di Eugenio, 2007; Islam in Inkpen, 2009; Han and Baldwin, 2011; Wu et al., 2013).

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; Shaalan et al., 2012; Attia et al., 2012; Alkanhal et al., 2012).

In our research, we address the spelling error detection and correction problem with a focus on non-word errors. Our work is different from previous work on Arabic in that we cover punctuation errors as well. Furthermore, we fine-tune a Language Model (LM) disambiguator by adding probability scores for candidates using forward-backward tracking, which yielded better results than the default Viterbi. We also develop a new and more efficient splitting algorithm for merged words.

1.2 Arabic Morphology, Orthography and Punctuation

Arabic has a rich and complex morphology as it applies both concatenative and non-concatenative morphotactics (Ratcliffe, 1998; Beesley, 1998; Habash, 2010), yielding a wealth of morphemes that express various morpho-

[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 EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 148–154,
October 25, 2014, Doha, Qatar.

syntactic features, such as tense, person, number, gender, voice and mood.

Arabic has a large array of orthographic variations, leading to what is called 'typographic errors' or 'orthographic variations' (Buckwalter, 2004a), and sometimes referred to as substandard spellings, or spelling soft errors. These errors are basically related to the possible overlap between orthographically similar letters in three categories: a) the various shapes of *hamzah*s (ا A[2], أ >, إ <, آ |, ئ }, ء ', ؤ &); b) *taa marboutah* and *haa* ة p, ه h); and c) *yaa* and *alif maqsoura* (ي y, ى Y).

Ancient Arabic manuscripts were written in *scriptura continua*, meaning running words without punctuation marks. Punctuation marks were introduced to Arabic mainly through borrowing from European languages via translation (Alqinai, 2013). Although punctuation marks in Arabic are gaining popularity and writers are becoming more aware of their importance, yet many writers still do not follow punctuation conventions as strictly and consistently as English writers. For example, we investigated contemporaneous same sized tokenized (simple tokenization with separation of punctuation) English and Modern Standard Arabic Gigaword edited newswire corpora, we found that 10% of the tokens in the English Gigaword corresponded to punctuation marks, compared to only 3% of the tokens in the Arabic counterpart.

	Train.	%	Dev.	%
Word Count	925,643	--	48,471	--
Total Errors	306,757	33.14	16,659	34.37
Word errors	187,040	60.97	9,878	59.30
Punc. errors	618,886	39.03	6,781	40.70
Split	10,869	3.48	612	3.67
Add_before	99,258	32.36	5,704	34.24
Delete	6,778	2.21	338	2.03
Edit	169,769	55.34	8,914	53.51
Merge	18,267	5.95	994	5.97
Add_after	20	0.01	2	0.01
Move	427	0.14	13	0.08

Table 1. Distribution Statistics on Error Types

1.3 Data Analysis

In our work, we use the QALB corpus (Zaghouani et al. 2014), and the training and development set provided in the QALB shared task (Mohit et. al 2014). The shared task addresses a large array of errors, and not just typical spelling errors. For instance, as Table 1 illustrates punctuation errors make up to 40% of all the errors in the shared task.

For further investigation, we annotated 1,100 words from the development set for error types, and found that 85% of the word errors (excluding punctuation marks) are typical spelling errors (or non-word errors), while 15% are real-word errors, or lexical ambiguities (that is, they are valid words outside of their context), and they range between dialectal words, grammatical errors, semantic errors, speech effects and elongation, examples shown in Table 2.

Error Type	Example	Correction
dialectal words	بهاي bhAy 'by this' [Syrian]	بهذه bh*h 'by this' [MSA]
grammatical errors	كبير kbyr 'big.masc'	كبيرة kbyrp 'big.fem'
semantic errors	آتيه |tyh 'come to him'	آتية |typ 'coming'
speech effects	الرجاااال AlrjAAAAl 'men'	الرجال AlrjAl 'men'
elongation	دمـاء dm__A' 'blood'	دماء dmA' 'blood'

Table 2. Examples of real word errors

2 Our Methodology

Due to the complexity and variability of errors in the shared task, we treat each problem individually and use different approaches that prove to be most appropriate for each problem. We specifically address three subtypes of errors: orthographical errors; punctuation errors; and non-word errors.

2.1 Orthographical Errors

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 following:

1. Hindi numbers (٠١٢٣٤٥٦٧٨٩) are converted into Arabic numerals [0-9] (occurs 495 in the training data times);
2. Speech effects are removed. For example, الرجاااال AlrjAAAAl 'men' is converted to الرجال AlrjAl. As a general rule letters repeated three times or more are reduced to one letter (715 times);
3. Elongation or kashida is removed. For example, دمـاء dm__A' 'blood' is converted to

[2] In this paper, we use the Buckwalter Transliteration Scheme as described in www.qamus.com.

دماء dmA' (906 times);

4. Special character U+06CC, the Farsi yeh: ى is converted to U+0649, the visually similar Arabic *alif maqsoura* ى Y (293 times).

2.2 Punctuation Errors

Punctuation errors constitute 40% of the errors in the QALB Arabic data. It is worth noting that by comparison, punctuation errors only constituted 4% of the English data in CoNLL 2013 Shared Task on English Grammatical Error Correction (Ng et al., 2013) and were not evaluated or handled by any participant. In HASP, we focus on 6 punctuation marks: comma, colon, semi-colon, exclamation mark, question mark and period.

The 'column' file in the QALB shared task data comes preprocessed with the MADAMIRA morphological analyzer version 04092014-1.0-beta (Pasha et al., 2014). The features that we utilize in our punctuation classification experiments are all extracted from the 'column' file, and they are as follows:

(1) The original word, that is the word as it appears in the text without any further processing, (e.g., للتشاور llt$Awr 'for consulting');
(2) The tokenized word using the Penn Arabic Treebank (PATB) tokenization (e.g., ل+ التشاور l+Alt$Awr);
(3) Kulick 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: colon_after, comma_after, exclmark_after, period_after, qmark_after, semicolon_after and NA (when no punctuation marks are used);

Window Size	Recall	Precision	F-measure
4	36.24	54.09	43.40
5	**37.95**	59.61	**46.37**
6	36.65	**59.99**	45.50
7	34.50	59.53	43.68

Table 3. Yamcha results on the development set

For classification, we experiment with Support Vector Machines (SVM) as implemented in Yamcha (Kudo and Matsumoto, 2003) and Conditional Random Field (CRF++) classifiers (Lafferty et al. 2001). In our investigation, we vary the context window size from 4 to 8 and we use all 5 features listed for every word in the window. As Tables 3 and 4 show, we found that window size 5 gives the best f-score by both Yamcha and CRF. When we strip clitics from tokenized tag, reducing it to stems only, the performance of the system improved. Overall CRF yields significantly higher results using the same experimental setup. We assume that the performance advantage of CRF is a result of the way words in the context and their features are interconnected in a neat grid in the template file.

#	Window Size	Recall	Precision	f-measure
1	4	44.03	74.33	55.31
2	5	**44.50**	**75.49**	**55.99**
3	6	44.22	74.93	55.62
4	7	43.81	75.09	55.34
5	8	43.49	75.41	55.17
6	8*	43.31	75.37	55.00

Table 4. CRF results on the development set
* with full tokens; other experiments use stems only, i.e., clitics are removed.

2.3. Non Word Errors

This type of errors comprises different subtypes: merges where two or more words are merged together; splits where a space is inserted within a single word; or misspelled words (which underwent substitution, deletion, insertion or transposition) that should be corrected. We handle these problems as follows.

2.3.1. 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. They constitute 3.67% and 3.48% of the error types in the shared task's development and training data, respectively. Attia et al. (2012) used an algorithm for dealing with merged words in Arabic, that is, $l-3$, where l is the length of a word. For a 7-letter word, their algorithm generates 4 candidates as it allows only a single space to be inserted in a string. Their algorithm, however, is too restricted. By contrast Alkanhal et al. (2012) developed an algorithm with more generative power, that is 2^{l-1}. Their algorithm, however, is in practice too general and leads to a huge fan out. For a 7-letter word, it generates 64 solutions. We develop a splitting algorithm by taking into account that the minimum length of words in Arabic is two. Our modified algorithm is 2^{l-4}, which creates an effective balance between comprehensiveness and compactness. For the 7-letter word, it generates 8 candidates. However, from Table 5 on merged words and their gold splits, one would question

the feasibility of producing more than two splits for any given string. Our splitting algorithm is evaluated in 2.3.3.1.c and compared to Attia et al.'s (2012) algorithm.

	Development	Training
Total Count	631	11,054
1 split	611	10,575
2 splits	15	404
3 splits	3	57
4 splits	1	13
5 splits	1	5

Table 5. Merged words and their splits

2.3.2. 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' (correction is صمام SmAm). This error constitutes 6% of the shared task's both training and development set. 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. Among the 18,267 splits in the training data 15,548 of them involved the *waw*, corresponding to 85.12%. Similarly among the 994 splits in the development data, 760 of them involved the *waw* (76.46%).

Therefore, we opted to handle this problem in our work in a partial and shallow manner using deterministic rules addressing specifically the following two phenomena:

1. Separated conjunction morpheme *waw* و w 'and' is attached to the succeeding word (occurs 15,915 times in the training data);
2. Literal strings attached to numbers are separated with space(s). For example, "دماء2000شهيدا" "dmA'2000$hydF" 'blood of 2000 martyrs' is converted to "دماء 2000 شهيدا" "dmA' 2000 $hydF" (824 times).

2.3.3. Misspelled Word Errors

This is more akin to the typical spelling correction problem where a word has the wrong letters, rendering it a non-word. We address this problem using two approaches: Dictionary-LM Correction, and Alignment Based Correction.

2.3.3.1. Dictionary-LM Correction

Spelling error detection and correction mainly consists of three phases: a) error detection; b) candidate generation; and c) error correction, or best candidate selection.

a. Error Detection

For non-word spelling error detection and candidate generation we use AraComLex Extended, an open-source reference dictionary (or word list) of full-form words. The dictionary is developed by Attia et al. (2012) through an amalgamation of various resources, such as a wordlist from the Arabic Gigaword corpus, wordlist generated from the Buckwalter morphological analyzer, and AraComLex (Attia et al., 2011), a finite-state morphological transducer. AraComLex Extended consists of 9.2M words and, as far as we know, is the largest wordlist for Arabic reported in the literature to date.

We enhance the AraComLex Extended dictionary by utilizing the annotated data in the shared task's training data. We add 776 new valid words to the dictionary and remove 4,810 misspelt words, leading to significant improvement in the dictionary's ability to make decisions on words. Table 6 shows the dictionary's performance on the training and development set in the shared task as applied only to non-words and excluding grammatical, semantic and punctuation errors.

data set	R	P	F
Training	98.84	96.34	97.57
Development	98.72	96.04	97.36

Table 6. Results of dictionary error detection

b. Candidate Generation

For candidate generation we use Foma (Hulden, 2009), a finite state compiler that is capable of producing candidates from a wordlist (compiled as an FST network) within a certain edit distance from an error word. Foma allows the ranking of candidates according to customizable transformation rules.

#	Error Type	Count	Ratio %
1.	أ > typed as ا A	59,507	31.82
2.	Insert	28.945	15.48
3.	إ < typed as ا A	25.392	13.58
4.	Delete	18.246	9.76
5.	ة p typed as ه h	14.639	7.83
6.	Split	11.419	6.11
7.	ي y typed as ى Y	6.419	3.43

Table 7. Error types in the training set

We develop a re-ranker based on our observation of the error types in the shared task's training data (as shown in Table 7) and examining the character transformations between the misspelt words and their gold corrections. Our statistics

shows that soft errors (or variants as explained in Section 1.2) account for more than 62% of all errors in the training data.

c. Error Correction

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 bigram language model is trained on the Gigaword Corpus 4th edition (Parker et al., 2009).

For the LM disambiguation we use the '–fb' option (forward-backward tracking), and we provide candidates with probability scores. We generate these probability scores by converting the edit distance scores produced by the Foma FST re-ranker explained above. Both of the forward-backward tracking and the probability scores in 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).

The best f-score achieved in our system is obtained when we combine the CRF punctuation classifier (merged with the original punctuations found in data), knowledge-based normalization (norm), dictionary-LM disambiguation and split-1, as shown in Table 8. The option split-1 refers to using the splitting algorithm $l-3$ as explained in Section 2.3.1, while split-2 refers to using the splitting algorithm 2^{l-4}.

#	Experiment	R	P	F
1	LM+split-1	33.32	**73.71**	45.89
2	+CRF_punc+split-1	49.74	65.38	56.50
3	+ norm+split-1	38.81	69.08	49.70
4	+CRF_punc+norm +split-1	**54.79**	67.65	60.55
5	+CRF_punc+norm +orig_punc+split-1	53.18	73.15	**61.59**
6	+CRF_punc+norm +orig_punc+split-2	53.13	73.01	61.50

Table 8. LM correction with 3 candidates

In the QALB Shared Task evaluation, we submit two systems: System 1 is configuration 5 in Table 8, and System 2 corresponds to configuration 6, and the results on the test set are shown in Table 9. As Table 9 shows, the best scores are obtained by System 1, which is ranked 5th among the 9 systems participating in the shared task.

#	Experiment	R	P	F
1	System 1	52.98	**75.47**	**62.25**
2	System 2	**52.99**	75.34	62.22

Table 9. Final official results on the test set provided by the Shared Task

2.3.3.2. Alignment-Based Correction

We formatted the data for alignment using a window of 4 words: one word to each side (forming the contextual boundary) and two words in the middle. The two words in the middle are split into characters so that character transformations can be observed and learned by the aligner. The alignment tool we use is Giza++ (Och and Ney, 2003). Results are reported in Table 10.

#	Experiment	R	P	F
1	for all error types	36.05	45.13	37.99
2	excluding punc	32.37	54.65	40.66
3	2 + CRF_punc+norm	46.11	62.02	52.90

Table 10. Results of character-based alignment

Although these preliminary results from Alignment are significantly below results yielded from the Dictionary-LM approach, we believe that there are several potential improvements that need to be explored:

- Using LM on the output of the alignment;
- Determining the type of errors that the alignment is most successful at handling: punctuation, grammar, non-words, etc;
- Parsing training data errors with the Dictionary-LM disambiguation and retraining, so instead of training data consisting of errors and gold corrections, it will consist of corrected errors and gold corrections.

3 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 better algorithm for handing merged word errors, tune the LM parameters, and combine the various components together, leading to cumulative improved results.

References

Alfaifi, A., and Atwell, E. (2012) Arabic Learner Corpora (ALC): a taxonomy of coding errors. In Proceedings of the 8th International Computing Conference in Arabic (ICCA 2012), Cairo, Egypt.

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.

Alqinai, Jamal. (2013) Mediating punctuation in English Arabic translation. Linguistica Atlantica. Vol. 32.

Attia, M., Pecina, P., Tounsi, L., Toral, A., and van Genabith, J. (2011) An Open-Source Finite State Morphological Transducer for Modern Standard Arabic. International Workshop on Finite State Methods and Natural Language Processing (FSMNLP). Blois, France.

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

Beesley, Kenneth R. (1998). Arabic Morphology Using Only Finite-State Operations. In The Workshop on Computational Approaches to Semitic languages, Montreal, Quebec, pp. 50–57.

Ben Othmane Zribi, C. and Ben Ahmed, M. (2003) Efficient Automatic Correction of Misspelled Arabic Words Based on Contextual Information, Lecture Notes in Computer Science, Springer, Vol. 2773, pp.770–777.

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.

Brown, P. F., Della Pietra, V. J., de Souza, P. V., Lai, J. C. and Mercer, R. L. (1992) Class-Based n-gram Models of Natural Language. *Computational Linguistics*, 18(4), 467–479.

Buckwalter, T. (2004b) Buckwalter Arabic Morphological Analyzer (BAMA) Version 2.0. Linguistic Data Consortium (LDC) catalogue number: LDC2004L02, ISBN1-58563-324-0.

Buckwalter, Tim. (2004a) Issues in Arabic orthography and morphology analysis. Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages. Pages 31-34. Association for Computational Linguistics Stroudsburg, PA, USA.

Church, Kenneth W. and William A. Gale. (1991) Probability scoring for spelling correction. *Statistics and Computing*, 1, pp. 93–103.

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

Damerau, Fred J. (1964) A Technique for Computer Detection and Correction of Spelling Errors. Communications of the ACM, Volum 7, issue 3, pp. 171–176.

Gao, Jianfeng, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. (2010) A large scale ranker-based system for search query spelling correction. Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010), pages 358–366, Beijing, China

Golding, Andrew R. A Bayesian Hybrid Method for Context-sensitive Spelling Correction. In Proceedings of the Third Workshop on Very Large Corpora. MIT, Cambridge, Massachusetts, USA. 1995, pp.39–53.

Golding, Andrew R., and Dan Roth. (1996) Applying Winnow to Context-Sensitive Spelling Correction. In Proceedings of the Thirteenth International Conference on Machine Learning, Stroudsburg, PA, USA, pp. 182–190

Habash, Nizar Y. (2010) *Introduction to Arabic Natural Language Processing*. Synthesis Lectures on Human Language Technologies 3.1: 1-187.

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.

Han, Bo and Timothy Baldwin. (2011) Lexical Normalisation of Short Text Messages: Makn Sens a #twitter. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 368–378, Portland, Oregon, June 19-24, 2011

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

Islam, Aminul, Diana Inkpen. (2009) Real-Word Spelling Correction using Google Web 1T n-gram with Backoff. International Conference on Natural Language Processing and Knowledge Engineering, Dalian, China, pp. 1–8.

Kiraz, G. A. (2001) *Computational Nonlinear Morphology: With Emphasis on Semitic Languages*. Cambridge University Press.

Kudo, Taku, Yuji Matsumoto. (2003) Fast Methods for Kernel-Based Text Analysis. 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003), Sapporo, Japan.

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.

Levenshtein, V. I. (1966) Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, pp. 707-710.

Magdy, W., and Darwish, K. (2006) Arabic OCR error correction using character segment correction, language modeling, and shallow morphology. EMNLP '06 Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing.

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.

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. (2009) Arabic Gigaword Fifth Edition. LDC Catalog No.: LDC2009T30, ISBN: 1-58563-532-4.

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, Reykjavik, Iceland.

Ratcliffe, Robert R. (1998) *The Broken Plural Problem in Arabic and Comparative Semitic: Allomorphy and Analogy in Non-concatenative Morphology*. Amsterdam studies in the theory and history of linguistic science. Series IV, Current issues in linguistic theory ; v. 168. Amsterdam ; Philadelphia: J. Benjamins.

Roth, R. Rambow, O., Habash, N., Diab, M., and Rudin, C. (2008) Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. Proceedings of ACL-08: HLT, Short Papers, pp. 117–120.

Shaalan, K., Samih, Y., Attia, M., Pecina, P., and van Genabith, J. (2012) Arabic Word Generation and Modelling for Spell Checking. Language Resources and Evaluation (LREC). Istanbul, Turkey. pp. 719–725.

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.

van Delden, Sebastian, David B. Bracewell, and Fernando Gomez. (2004) Supervised and Unsupervised Automatic Spelling Correction Algorithms. In proceeding of Information Reuse and Integration (IRI). Proceedings of the 2004 IEEE International Conference on Web Services, pp. 530–535.

Wu, Jian-cheng, Hsun-wen Chiu, and Jason S. Chang. (2013) Integrating Dictionary and Web N-grams for Chinese Spell Checking. Computational Linguistics and Chinese Language Processing. Vol. 18, No. 4, December 2013, pp. 17–30.

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.