Improved Spelling Error Detection and Correction for Arabic

Mohammed Attia[1, 4] *Pavel Pecina*[2] *Younes Samih*[3]
Khaled Shaalan[1] *Josef van Genabith*[4]

(1) The British University in Dubai, UAE
(2) Institute of Formal and Applied Linguistics, Faculty of Mathematics and Physics, Charles University in Prague, Czech Republic
(3) Heinrich-Heine-Universität, Germany
(4) School of Computing, Dublin City University, Ireland

{mattia,josef}@computing.dcu.ie, pecina@ufal.mff.cuni.cz, samih@phil.uni-duesseldorf.de, khaled.shaalan@buid.ac.ae

ABSTRACT

A spelling error detection and correction application is based on three main components: a dictionary (or reference word list), an error model and a language model. While most of the attention in the literature has been directed to the language model, we show how improvements in any of the three components can lead to significant cumulative improvements in the overall performance of the system. We semi-automatically develop a dictionary of 9.3 million fully inflected Arabic words using a morphological transducer and a large corpus. We improve the error model by analysing error types and creating an edit distance based re-ranker. We also improve the language model by analysing the level of noise in different sources of data and selecting the optimal subset to train the system on. Testing and evaluation experiments show that our system significantly outperforms Microsoft Word 2010, OpenOffice Ayaspell and Google Docs.

TITLE IN ARABIC

تحسين اكتشاف وتصحيح الأخطاء الإملائية في اللغة العربية

ABSTRACT IN ARABIC

تقوم تطبيقات اكتشاف وتصحيح الأخطاء الإملائية على ثلاث مكونات رئيسية وهي: قاموس (أو قائمة كلمات مرجعية)، ونموذج الخطأ ونموذج اللغة. وبينما ينصب الاهتمام في الأبحاث العلمية على نموذج اللغة، فإننا نبين أن التحسينات التي يتم إدخالها على المكونات الثلاثة يمكن أن تؤدي إلى تحسينات تراكمية في النتائج النهائية لأداء البرنامج. وقد قمنا في هذا البحث بتطوير قاموس يتكون من 9.3 مليون مصرفة تصريفا كاملا كلمة بشكل شبه آلي باستخدام محلل صرفي وذخيرة نصوص كبيرة. وقمنا بتحسين نموذج الخطأ عن طريق تحليل أنواع الأخطاء الإملائية وتطوير وسيلة لإعادة ترتيب المقترحات الناتجة عن خوارزمية مسافة التحرير. وقمنا كذلك بتحسين نموذج اللغة عن طريق تحليل نسبة الأخطاء في مصادر البيانات المختلفة واختيار الجزء المثالي لتدريب البرنامج عليه. وتبين تجارب الاختبار والتقييم أن البرنامج يتفوق بشكل كبير على مايكروسوفت أوفيس 2010 وأوبن أوفيس وملفات جوجل.

KEYWORDS : Arabic, spelling error detection and correction, finite state morphological generation, Arabic spell checker, spelling error model, Arabic word list
KEYWORDS IN ARABIC:
اللغة العربية، اكتشاف وتصحيح الأخطاء الإملائية، التوليد الصرفي باستخدام آلات الحالة المحدودة، التدقيق الإملائي، قائمة كلمات اللغة العربية

1 Introduction

Spelling correction solutions have significant importance for a variety of applications and NLP tools including text authoring, OCR, pre-editing or post-editing for parsing and machine translation, intelligent tutoring systems, etc.

The spelling correction problem is formally defined (Brill, and Moore, 2000) as: given an alphabet Σ, a dictionary D consisting of strings in Σ^*, and a spelling error s, where $s \notin D$ and $s \in \Sigma^*$, find the correction c, where $c \in D$, and c is most likely to have been erroneously typed as s. This is treated as a probabilistic problem formulated as (Kernigan, 1990; Norvig, 2009; Brill, and Moore, 2000):

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

Here c is the correction, s is the spelling error, $P(c)$ 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 all plausible values of the correction c and maximizes its probability.

The definition shows that a good spelling correction system needs a balanced division of labour between the three main components: the dictionary, error model and language model. In this paper we show that in the error model there is a direct relationship between the number of correction candidates and the likelihood of finding the correct correction: the larger the number of candidates generated by the error model, the more likely is to include the best correction. At the same time, in the language model there is an inverse relationship between the number of candidates and the ability of the model to decide on the right correction: the larger the number of candidates, the less likely the language model will be successful in making the correct choice. A language model is negatively affected by a high dimensional search space. A language model is also negatively affected by noise in the data when the size of the data is not very large.

In this paper we deal with Modern Standard Arabic as used in official and edited news web sites. Dialectal Arabic is beyond the scope of this research. Furthermore, we deal with non-word errors; real word errors are not handled in this paper. The problem of spell checking and spelling error correction for Arabic has been investigated in a number of papers. Shaalan et. al. (2003) provide a characterization and classification of spelling errors in Arabic. Haddad and Yaseen (2007) propose a hybrid approach that utilizes morphological knowledge to formulate morphographemic rules to specify the word recognition and non-word correction process. Shaalan et. al. (2012) use the Noisy Channel Model trained on word-based unigrams for spelling correction, but their system performs poorly against the Microsoft Spell Checker.

Our research differs in that we use an n-gram language model (mainly bigrams) trained on the largest available corpus to date, the Arabic Gigaword Corpus 5th Edition. In addition, we classify spelling errors by comparing the errors with the gold correction, and, based on this classification, we develop knowledge-based re-ranking rules for reordering and constraining the number of candidates generated though the Levenshtein edit distance (Levenshtein, 1966) and integrate them into the overall model. Furthermore, we show that careful selection of the language model training data based on the amount of noise present in the data, has the potential to further improve overall results. We also highlight the importance of the dictionary (or reference word list) in the processes of spell checking and candidate generation.

In order to evaluate the system, we create a test corpus of 400,000 running words (tokens) consisting of news articles collected from various sources on the web (and not included in the corpus used in the development phase). From this test corpus, we extract 2,027 spelling errors (naturally occurring and not automatically generated), and we manually provide each spelling error with its gold correction. We test our system against this gold standard and compare it to Microsoft Word 2010, OpenOffice Ayaspell, and Google Docs. Our system performs significantly better than the three other systems both in the tasks of spell checking and automatic correction (or 1st order ranking).

The remainder of this paper is structured as follows: Section 2 shows how the dictionary (or word list) is created from the AraComLex finite-state morphological generator (Attia et al., 2011), and how spelling errors are detected. Section 3 explains how the error model is improved by developing rules to improve the ranking produced through finite-state edit distance. Section 4 shows how the language model is improved by selecting the right type of data to be trained on. Various data sections are analysed to detect the amount of noise they have, then some subsets of data are chosen for the n-gram language model training and the evaluation experiments. Finally Section 5 concludes.

2 Improving the Dictionary

The dictionary (or word list) is an essential component of a spell checker/corrector, as it is the reference against which the decision can be made whether a given word is correct or misspelled. It is also the reference against which correction candidates are filtered. There are various options for creating a word list for spell checking. It can be created from a corpus, a morphological analyser/generator, or both. The quality of the word list will inevitably affect the quality of the application whether in checking errors or generating valid and plausible candidates.

We use the AraComLex Extended word list for Arabic described in Shaalan et. al. (2012) further enhancing it by matching its word list against the Gigaword corpus. Words found in the Gigaword corpus and not included in the AraComLex Extended are double-checked by a second morphological analyser, the Buckwalter Arabic Morphological Analyser (Buckwalter, 2004), and the mismatches are manually revised, ultimately creating a dictionary of 9.3 million fully inflected Arabic word types.

For spelling error detection, we use two methods, the direct method, that is matching against the dictionary (or word list), and a character-based language modelling method in case such a word list is not available. The direct way for detecting spelling errors is to match words in an input text against a dictionary. Such a dictionary for Arabic runs into several million word types, therefore it is more efficient to use finite state automata to store words in a more compact manner. An input string is then compared against the valid word list paths and spelling errors will show as the difference between the two word lists (Hassan et al., 2008, Hulden, 2009a).

Shaalan et. al. (2012) implement error detection through language modelling. They build a character-based tri-gram language model using SRILM[1] (Stolcke et al., 2011) in order to classify words as valid and invalid. They split each word into characters, and create two language models: one for the total list of words pre-classified as valid (9,306,138 words), and one for a list of words classified as invalid (5,841,061 words). Their method yields a precision of 98.2 % at a recall of 100 %.

[1] http://www.speech.sri.com/projects/srilm/

3 Improving the Error Model: Candidate Generation

Having detected a spelling error, the next step is to generate possible and plausible corrections for that error. For a spelling error s and a dictionary D, the purpose of the error model is to generate the correction c, or list of corrections c_1^n where $c_1^n \in D$, and c_1^n are most likely to have been erroneously typed as s. In order to do this, the error model generates a list of candidate corrections $c_1, c_2, \dots, c_n$ that bear the highest similarity to the spelling error s.

We use a finite-state transducer to propose candidate corrections within edit distance 1 and 2 measured by Levenshtein Distance (Levenshtein, 1966) from the misspelled word (Oflazer, 1996; Hulden, 2009b; Norvig, 2009; Mitton, 1996). The transducer works basically as a character-based generator that replaces each character with all possible characters in the alphabet as well as deleting, inserting, and transposing neighbouring characters. There is also the problem of merged (or run-on) words that need to be split, such as أوأي '>w>y' "or any".

Candidate generation using edit distance is a brute-force process that ends up with a huge list of candidates. Given that there are 35 alphabetic letters in Arabic, for a word of length n, there will be n deletions, $n-1$ transpositions, $35n$ replaces, $35(n+1)$ insertions and $n-3$ splits, totaling $73n+31$. For example, a misspelt word consisting of 6 characters will have 469 candidates (with possible repetitions). This large number of candidates needs to be filtered and reordered in such a way that the correct correction comes top or near the top of the list. To filter out unnecessary words, candidates that are not found in the dictionary (or word list) are discarded. The ranking of the candidates is explained in the following sub-section.

3.1 Candidate Ranking

The ranking of candidates is initially based on a simple edit distance measure where the cost assignment is based on arbitrary letter change. In order to improve the ranking, we analyse error types in our gold standard of 2,027 misspelt words with their corrections to determine how they are distributed in order to devise ranking rules for the various edit operations.

```
Insert 2
Substitute 2
Delete 2
```

FIGURE 1 – Simple edit distance measure

```
Insert 3
Substitute 3
Delete 3
Cost 2
&:> r: :r }:y j:p :t :A :w t:n b:y :h :d ':} s:$ :' w: |:A d: :Y p:h v:t :l w:A :n v:t t:v T: :T
$:s b: l: >:| y:} q: :q >:' y:d z:s m: n:t x:H n: t: }:> x: :E >:& }:& :w f:k S: :S :b d:*
Y:A |:< w:r q:f :m j:k :& g:j T:t h:d :p p: ':> }:' :} q: l: :y y: l:S A:
Cost 1
>:A <:A |:A A:| A:> A:< <:> >:< |:> >:| >:& y:Y Y:y h:p h:p &:' H:j j:H S:l y ':} y ':y
```

FIGURE 2 – Re-ranked edit distance

Based on these facts we use a re-ranker to order edit distance operations according to their likelihood to generate the most plausible correction. Our analysis shows that *hamzahs* (>, <, &,

A, }, {, ' and |), the pair of *yaa* (y) and *alif maqsoura* (Y), and the pair of *taa marboutah* (p) and *haa* (h) contribute to the largest amount of spelling errors. Our re-ranker is sensitive to these facts and primes the edit distance scoring mechanism with different rules following the error patterns for Arabic. It assigns a lower cost score to the most-frequently confused character sets (which are often graphemically similar), and a higher score to other operations. We use the foma (Hulden, 2009b) "apply med <string>" command to find approximate matches to the string in the top network by minimum edit distance. Figure 1 and 2 show the different configuration files for the simple and re-ranked edit distance.

We also notice that split words constitute 16 % of the spelling errors in the Arabic data. These are cases where two words are joined together and the space is omitted, such as عبدالدايم 'EbdAldAym' "Abdul-Dayem". The problem with split words is that they are not handled by the edit distance operation. Therefore we add a process for automatically inserting spaces between the various parts of the string. This will create more candidates: a word of length n will have $n - 3$ candidates, given that the minimum word length in Arabic is two characters.

3.2 Evaluation of the Candidate Ranking Technique

Our purpose in ranking generated candidates is to see the correct candidate (the gold correction) at or near the top of the list, so that when we reduce the list of candidates we do not lose so many of the correct ones. We test the ranking mechanism using our gold standard of 2,027 misspelt words with their gold correction.

Cut-off limit	**Simple edit distance score gold found in candidates**		**Re-ranked edit distance score gold found in candidates**	
	without split words %	**after adding split words %**	**without split words %**	**after adding split words %**
100	79.97	90.97	82.09	93.09
90	79.87	90.87	82.04	93.04
80	79.72	90.73	82.04	93.04
70	79.33	90.33	82.04	93.04
60	78.93	89.94	81.85	92.85
50	78.34	89.34	81.85	92.85
40	77.16	88.16	81.65	92.65
30	75.04	86.04	81.55	92.55
20	71.88	82.88	81.01	92.01
10	64.58	75.58	79.92	90.92
9	62.90	73.90	79.72	90.73
8	61.77	72.77	79.63	90.63
7	59.60	70.60	79.13	90.13
6	56.83	67.83	78.93	89.94
5	53.33	64.33	78.59	89.59
4	48.99	59.99	78.10	89.10
3	44.06	55.06	77.70	88.70
2	37.15	48.15	75.78	86.78
1	23.88	34.88	65.66	76.67

TABLE 1 – Comparing simple edit distance with re-ranked edit distance

We compare the simple edit distance measure to our revised edit distance re-ranking scorer. As Table 1 shows, the re-ranking scorer performs better at all levels. We notice that when the number of candidates is large the difference between the simple edit distance and the re-ranked edit distance is not big (about 2 % absolute at the 100 cut-off limit without splits), but when the limit for the number of candidate is lowered the difference increases quite considerably (about 42 % absolute at 1 cut-off limit without splits). This indicates that our knowledge-based edit distance re-ranker has been successful in pushing good candidates up the top of the list. We also notice that adding splits for merged words has a beneficial effect on all counts.

4 Spelling Correction

Having generated correction candidates and improved their ranking based on the study of the frequency of the error types, we now use language models trained on different corpora to finally choose the single best correction. We compare the results against the Microsoft Spell Checker in Office 2010, Ayaspell used in OpenOffice, and Google Docs.

4.1 Correction Procedure

For automatic spelling correction (or first order ranking) we use n-gram language models. Language modelling assumes that the production of a human language text is characterized by a set of conditional probabilities, $P(w_k|w_1^{k-1})$, where w_1^{k-1} is the history and w_k is the prediction, so that the probability of a sequence of k words $P(w_1, ..., w_k)$ is formulated as a product (Brown et al., 1992):

$$P(w_1^k) = P(w_1)P(w_2 \mid w_1) \dots P(w_k|w_1^{k-1})$$

We use the SRILM toolkit[2] (Stolcke et al., 2011) to train 2-, 3- and 4-gram language models on our data sets. As we have two types of candidates, normal words and split words, we use two SRILM tools: *disambig* and *ngram*. We use the *disambig* tool to choose among the normal candidates. Handling split words is done as a posterior step where we use the *ngram* tool to score the chosen candidate from the first round and the various split-word options. Then the candidate with the least perplexity score is selected. The *perplexity* of a language model is the reciprocal of the geometric average of the probabilities. If a sample text S has $|S|$ words, then the perplexity is $P(S)^{-1/|S|}$ (Brown et al., 1992). This is why the language model with the smaller perplexity is in fact the one with the higher probability with respect to S.

4.2 Analysing the Training Data

Our language model is based on raw data from two sources: the Arabic Gigaword Corpus 5th Edition and a corpus of news articles crawled from the Al-Jazeera web site. The Gigaword corpus is a collection of news articles from nine news sources: Agence France-Presse, Xinhua News Agency, An Nahar, Al-Hayat, Al-Quds Al-Arabi, Al-Ahram, Assabah, Asharq Al-Awsat and Ummah Press.

Before we start using our available corpora in training the language model, we analyse the data to measure the amount of noise in each subset of the data. In order to do this, we create a list of the most common spelling errors. This list of spelling errors is created by analysing the data using MADA (Habash et al., 2005; Roth et al., 2008) and checking instances where words have been normalized. In this case the original word is considered to be a suboptimal variation of the spelling of the diacritized form. We collect these suboptimal forms and sort them by frequency.

[2] http://www.speech.sri.com/projects/srilm/

Then we take the top 100 misspelt forms and see how frequent they are in the different subsets of data in relation to the word count in each data set.

The analysis shows that the data has a varying degree of cleanness, ranging from the very clean to the very noisy. Data in the Agence France-Presse (AFP) is the noisiest while Ummah Press is the cleanest, and Al-Jazeera is the second cleanest. Due to the fact that the Ummah Press data is small in size compared to the AFP data we ignore it in our experiments and use instead the Al-Jazeera data for representing the cleanest data set.

4.3 Automatic Correction Evaluation

For comparison, we first evaluate the automatic correction (or first order ranking) of three industrial text authoring applications: Google Docs[3], Open-Office Ayaspell, and Microsoft Word. We use our test set of 2,027 spellings errors. We test the automatic correction on two levels: at the word type level (that is unique words without repetition) and the word token level (that is words as they are found in the corpus with possible repetition). The results in Table 2 are reported in terms of accuracy (number of correct corrections divided by the number of all errors).

	Google Docs Accuracy %	**OpenOffice Ayaspell Accuracy %**	**MS Word Accuracy %**
Tested on word types	17.02	41.88	71.24
Tested on word tokens	9.32	41.86	57.15

TABLE 2 – Evaluation of spelling correction of Google Docs, Ayaspell and MS Word 2010

cut-off limit	**normal candidates accuracy 2-gram**			**normal candidates + splitword accuracy 2-gram**		
	AFP	**Jazeera**	**Gigaword**	**AFP**	**Jazeera**	**Gigaword**
100	44.58	59.75	61.27	50.75	67.34	68.98
90	44.85	60.32	61.64	51.03	67.90	69.30
80	45.66	60.95	62.19	51.58	68.54	69.76
70	47.46	62.40	64.05	53.39	69.97	71.62
60	47.90	62.92	64.58	53.88	70.43	72.10
50	48.88	63.87	65.34	54.75	71.39	72.82
40	50.50	64.67	66.05	56.29	72.18	73.49
30	51.90	66.10	67.43	57.58	73.53	74.82
20	53.85	67.90	69.20	59.13	74.94	76.37
10	60.94	70.82	72.11	64.95	77.05	78.87
9	61.79	71.21	72.43	65.31	77.33	79.12
8	62.90	71.88	73.07	66.17	77.82	79.58
7	63.87	72.17	73.69	67.04	78.07	80.12
6	66.42	72.79	74.39	69.23	78.51	80.73
5	67.60	73.78	74.91	69.97	79.10	81.03
4	69.37	75.21	75.95	71.21	80.05	81.79
3	72.73	76.48	**77.24**	73.93	80.97	**82.86**
2	70.68	72.47	73.33	70.72	76.37	78.39

TABLE 3 – Correction accuracy with 2-gram LM trained on AFP, Al-Jazeera and Gigaword

[3] Tested in June 2012

Next we evaluate our approach using language models trained on the AFP data (as representing the noisiest type of data), the Al-Jazeera data (as representing the cleanest subset of data) and the entire Gigaword corpus (as representing a huge data set with a moderate amount of noise). We run our experiments on the candidates generated through the re-ranked edit distance processing explained in Section 3 with varying candidate cut-off limits. We test the normal candidates using the SRILM *disambig* tool and the split words using *ngram* tool.

As Table 3 shows, the best score achieved for the automatic correction is 82.86 % using the bigram language model with a candidate cut-off limit of 3, and with the split words added. Table 3 shows that when there are too many candidates (above 10 candidates) the n-gram language model performs poorly and with too few candidates (2 candidates) the performance also deteriorates considerably. Therefore a reasonable range for the number of candidates for the n-gram language model is between 7 and 3, with optimal performance at 3.

Comparing the two data sets which are comparable in size, the AFP and Al-Jazeera, we find that the best score achieved with the AFP data is 73.93 % that is 7.04 % absolute less than the best score achieved with the Al-Jazeera data (80.97 %). The quality of the data is a crucial factor here. The Al-Jazeera data is relatively clean while the AFP data is full of noise and misspellings. This emphasizes the point in language modelling that clean data is better than noisy data when they are comparable in size.

Table 3 also shows that the extremely large corpus ameliorates the effect of the noise and produces the best results among all the data sets. The best score achieved for the language model trained on the Gigaword corpus is 82.86 %, which is 1.89 % absolute better than the score for Al-Jazeera (80.97 %). This could be a further indication in favour of the argument that more data is better than clean data. However, we must notice that the Gigaword data is one order of magnitude larger than the Al-Jazeera data, and in some applications, for efficiency reasons, it could be better to work with the language model trained on Al-Jazeera. We notice that the addition of the split word component has a positive effect on all test results.

Compared to other spelling error detection and correction systems we notice that our best accuracy score (82.86 %) is significantly higher than that for Google Docs (9.32 %), Ayaspell for OpenOffice (41.86 %) and Microsoft Word 2010 (57.15 %).

Conclusion

We have developed methods for improving the main three components in a spelling error correction application: the dictionary (or word list), the error model and the language model. These three components are highly interconnected and interrelated. Without the dictionary being an exhaustive and accurate representation of the language words space, without an error model being able to generate a plausible and compact list of candidates, and without a language model being trained on either clean data or an extremely large amount of data, no high quality correction results can be expected. Our spelling correction methodology significantly outperforms the three industrial applications of Ayaspell, MS Word, and Google Docs in first order ranking of candidates.

Acknowledgments

This research is funded by the Irish Research Council for Science Engineering and Technology (IRCSET), the UAE National Research Foundation (NRF) (Grant No. 0514/2011), the Czech Science Foundation (grant no. P103/12/G084), and the Science Foundation Ireland (Grant No. 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

References

Attia, Mohammed, Pavel Pecina, Lamia Tounsi, Antonio Toral, Josef van Genabith. (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.

Beesley, K. R., and Karttunen, L. (2003). *Finite State Morphology*: CSLI studies in computational linguistics. Stanford, Calif.: CSLI.

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, 2003, Vol. 2773, pp.770–777.

Brill, Eric and Robert C. Moore. (2000). An improved error model for noisy channel spelling correction. ACL '00 Proceedings of the 38th Annual Meeting on Association for Computational Linguistics.

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

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

Choudhury, Monojit, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar and Anupam Basu. (2007). Investigation and modeling of the structure of texting language. *International Journal on Document Analysis and Recognition*. Volume 10, Numbers 3-4, 157-174, DOI: 10.1007/s10032-007-0054-0

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

Hajič, J., Smrž, O., Buckwalter, T., and Jin, H. (2005). Feature-Based Tagger of Approximations of Functional Arabic Morphology. In: The 4th Workshop on Treebanks and Linguistic Theories (TLT 2005), Barcelona, Spain.

Habash, Nizar and Rambow, Owen. (2005). Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05), pp. 573—580

Hassan, Ahmed, Sara Noeman and Hany Hassan. (2008). Language Independent Text Correction using Finite State Automata. IJCNLP. Hyderabad, India

Hulden, Mans. (2009a). Fast Approximate String Matching with Finite Automata. Proceedings of the 25th Conference of the Spanish Society for Natural Language Processing (SEPLN).

Hulden, Mans. (2009b). 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

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.

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

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

Magdy, Walid and Kareem Darwish. (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.

Mitton, Roger (1996). *English spelling and the computer.* Harlow, Essex: Longman Group

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

Oflazer, K. (1996) Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. Computational Linguistics 22(1): 73-90

Parker, Robert, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. (2011). Arabic Gigaword Fifth Edition. LDC Catalog No.: LDC2011T11, ISBN: 1-58563-595-2.

Roth, Ryan and Rambow, Owen and Habash, Nizar and Diab, Mona and Rudin, Cynthia. (2008). Arabic Morphological Tagging, Diacritization, and Lemmatization Using Lexeme Models and Feature Ranking. Proceedings of ACL-08: HLT, Short Papers, pp. 117--120.

Shaalan, Khaled,Younes Samih, Mohammed Attia, Pavel Pecina, and Josef van Genabith. (2012). Arabic Word Generation and Modelling for Spell Checking. Language Resources and Evaluation (LREC). Istanbul, Turkey. Pages: 719-725

Shaalan K., Allam A., Gomah A. (2003). Towards Automatic Spell Checking for Arabic, In Proceedings of the 4th Conference on Language Engineering, Egyptian Society of Language Engineering (ELSE), PP. 240-247, Oct. 21-22, Cairo, Egypt.

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

Watson, J. (2002). *The Phonology and Morphology of Arabic*, New York: Oxford University Press.

Wintner, Shuly. (2008). Strengths and weaknesses of finite-state technology: a case study in morphological grammar development Natural Language Engineering 14(4):457-469, October 2008.