Generalized Character-Level Spelling Error Correction

Noura Farra, Nadi Tomeh†, Alla Rozovskaya, Nizar Habash

Center for Computational Learning Systems, Columbia University
{noura,alla,habash}@ccls.columbia.edu
†LIPN, Université Paris 13, Sorbonne Paris Cité
nadi.tomeh@lipn.univ-paris13.fr

Abstract

We present a generalized discriminative model for spelling error correction which targets character-level transformations. While operating at the character level, the model makes use of word-level and contextual information. In contrast to previous work, the proposed approach learns to correct a variety of error types without guidance of manually-selected constraints or language-specific features. We apply the model to correct errors in Egyptian Arabic dialect text, achieving 65% reduction in word error rate over the input baseline, and improving over the earlier state-of-the-art system.

1 Introduction

Spelling error correction is a longstanding Natural Language Processing (NLP) problem, and it has recently become especially relevant because of the many potential applications to the large amount of informal and unedited text generated online, including web forums, tweets, blogs, and email. Misspellings in such text can lead to increased sparsity and errors, posing a challenge for many NLP applications such as text summarization, sentiment analysis and machine translation.

In this work, we present GSEC, a Generalized character-level Spelling Error Correction model, which uses supervised learning to map input characters into output characters in context. The approach has the following characteristics:

Character-level Corrections are learned at the character-level[1] using a supervised sequence labeling approach.

Generalized The input space consists of all characters, and a single classifier is used to learn common error patterns over all the training data, without guidance of specific rules.

Context-sensitive The model looks beyond the context of the current word, when making a decision at the character-level.

Discriminative The model provides the freedom of adding a number of different features, which may or may not be language-specific.

Language-Independent In this work, we integrate only language-independent features, and therefore do not consider morphological or linguistic features. However, we apply the model to correct errors in Egyptian Arabic dialect text, following a conventional orthography standard, CODA (Habash et al., 2012).

Using the described approach, we demonstrate a word-error-rate (WER) reduction of 65% over a do-nothing input baseline, and we improve over a state-of-the-art system (Eskander et al., 2013) which relies heavily on language-specific and manually-selected constraints. We present a detailed analysis of mistakes and demonstrate that the proposed model indeed learns to correct a wider variety of errors.

2 Related Work

Most earlier work on automatic error correction addressed spelling errors in English and built models of correct usage on native English data (Kukich, 1992; Golding and Roth, 1999; Carlson and Fette, 2007; Banko and Brill, 2001). Arabic spelling correction has also received considerable interest (Ben Othmane Zribi and Ben Ahmed, 2003; Haddad and Yaseen, 2007; Hassan et al., 2008; Shaalan et al., 2010; Alkanhal et al., 2012; Eskander et al., 2013; Zaghouani et al., 2014).

Supervised spelling correction approaches trained on paired examples of errors and their corrections have recently been applied for non-native English correction (van Delden et al., 2004; Li et al., 2012; Gamon, 2010; Dahlmeier and Ng, 2012;

[1] We use the term 'character' strictly in the alphabetic sense, not the logographic sense (as in the Chinese script).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 161–167,
Baltimore, Maryland, USA, June 23-25 2014.

Rozovskaya and Roth, 2011). Discriminative models have been proposed at the word-level for error correction (Duan et al., 2012) and for error detection (Habash and Roth, 2011).

In addition, there has been growing work on lexical normalization of social media data, a somewhat related problem to that considered in this paper (Han and Baldwin, 2011; Han et al., 2013; Subramaniam et al., 2009; Ling et al., 2013).

The work of Eskander et al. (2013) is the most relevant to the present study: it presents a character-edit classification model (CEC) using the same dataset we use in this paper.[2] Eskander et al. (2013) analyzed the data to identify the seven most common types of errors. They developed seven classifiers and applied them to the data in succession. This makes the approach tailored to the specific data set in use and limited to a specific set of errors. In this work, a single model is considered for all types of errors. The model considers every character in the input text for a possible spelling error, as opposed to looking only at certain input characters and contexts in which they appear. Moreover, in contrast to Eskander et al. (2013), it looks beyond the boundary of the current word.

3 The GSEC Approach

3.1 Modeling Spelling Correction at the Character Level

We recast the problem of spelling correction into a sequence labeling problem, where for each input character, we predict an **action label** describing how to transform it to obtain the correct character. The proposed model therefore transforms a given input sentence $\mathbf{e} = e_1, \ldots, e_n$ of n characters that possibly include errors, to a corrected sentence $\mathbf{c}$ of m characters, where corrected characters are produced by one of the following four actions applied to each input character e_i :

- ok: e_i is passed without transformation.
- $substitute - with(\mathrm{c})$: e_i is substituted with a character c where c could be any character encountered in the training data.
- $delete$: e_i is deleted.
- $insert(c)$: A character c is inserted $before$ e_i. To address errors occurring at the end of the sentence, we assume the presence of a dummy sentence-final $stop$ character.

[2]Eskander et al. (2013) also considered a slower, more expensive, and more language-specific method using a morphological tagger (Habash et al., 2013) that outperformed the CEC model; however, we do not compare to it in this paper.

Input	Action Label
k	*substitute-with(c)*
o	*ok*
r	*insert(r)*
e	*ok*
c	*ok*
t	*ok*
d	*delete*

Table 1: Character-level spelling error correction process on the input word $korectd$, with the reference word $correct$

	Train	Dev	Test
Sentences	10.3K	1.67K	1.73K
Characters	675K	106K	103K
Words	134K	21.1K	20.6K

Table 2: ARZ Egyptian dialect corpus statistics

We use a multi-class SVM classifier to predict the action labels for each input character $e_i \in \mathbf{e}$. A decoding process is then applied to transform the input characters accordingly to produce the corrected sentence. Note that we consider the space character as a character like any other, which gives us the ability to correct word merge errors with space character insertion actions and word split errors with space character deletion actions. Table 1 shows an example of the spelling correction process.

In this paper, we only model single-edit actions and ignore cases where a character requires multiple edits (henceforth, **complex** actions), such as multiple insertions or a combination of insertions and substitutions. This choice was motivated by the need to reduce the number of output labels, as many infrequent labels are generated by complex actions. An error analysis of the training data, described in detail in section 3.2, showed that complex errors are relatively infrequent (4% of data). We plan to address these errors in future work.

Finally, in order to generate the training data in the described form, we require a parallel corpus of erroneous and corrected reference text (described below), which we align at the character level. We use the alignment tool **Sclite** (Fiscus, 1998), which is part of the SCTK Toolkit.

3.2 Description of Data

We apply our model to correcting Egyptian Arabic dialect text. Since there is no standard dialect orthography adopted by native speakers of Arabic dialects, it is common to encounter multiple

Action	% Errors	Example Error ⇒ Reference
Substitute	**80.9**	
^{E} *Alif A*| forms (ٱٲٳAÂ/Ă/Ā)	33.3	***A**Hdhm* ⇒ ***Â**Hdhm* احدهم ⇒ أحدهم
^{E} *Ya* ى/ي forms (*y/ý*)	26.7	*ςly* ⇒ *ςlý* علي ⇒ على
^{E} *h/ħ* ه/ة , *h/w* ه/و forms	14.9	*kfr**h*** ⇒ *kfrħ* كفره ⇒ كفرة
^{E} *h/H* ه/ح forms	2.2	***h**tςmlhA* ⇒ ***H**tςmlhA* هتعملها ⇒ حتعملها
Other substitutions	3.8	*AltAnyħ* ⇒ *AlθAnyħ* التانية ⇒ الثانية ; *dA* ⇒ *dh* دا ⇒ ده
Insert	**10.5**	
^{EP} *Insert {A}*	3.0	*ktbw* ⇒ *ktbw**A*** كتبو ⇒كتبوا
^{EP} *Insert {space}*	2.9	*mAtzςlš* ⇒ *mA tzςlš* ماتزعلش ⇒ ما تزعلش
Other insertion actions	4.4	*Aly* ⇒ *Ally* الي ⇒ الـلي
Delete	**4.7**	
^{E} *Del{A}*	2.4	*whm**A*** ⇒ *whm* وهما ⇒وهم
Other deletion actions	2.3	*wfy**h*** ⇒ *wfy* وفيه ⇒ وفي
Complex	**4.0**	*mykwnš* ⇒ *mA ykwnš* ميكونش ⇒ ما يكونش

Table 3: Character-level distribution of correction labels. We model all types of transformations except *complex* actions, and rare *Insert* labels with counts below a tuned threshold. The *Delete* label is a single label that comprises all deletion actions. Labels modeled by Eskander et al. (2013) are marked with ^{E}, and ^{EP} for cases modeled partially, for example, the *Insert{A}* would only be applied at certain positions such as the end of the word.

spellings of the same word. The CODA orthography was proposed by Habash et al. (2012) in an attempt to standardize dialectal writing, and we use it as a reference of correct text for spelling correction following the previous work by Eskander et al. (2013). We use the same corpus (labeled "ARZ") and experimental setup splits used by them. The ARZ corpus was developed by the Linguistic Data Consortium (Maamouri et al., 2012a-e). See Table 2 for corpus statistics.

Error Distribution Table 3 presents the distribution of correction action labels that correspond to spelling errors in the training data together with examples of these errors.[3] We group the actions into: *Substitute*, *Insert*, *Delete*, and *Complex*, and also list common transformations within each group. We further distinguish between the phenomena modeled by our system and by Eskander et al. (2013). At least 10% of all generated action labels are not handled by Eskander et al. (2013).

3.3 Features

Each input character is represented by a feature vector. We include a set of basic features inspired by Eskander et al. (2013) in their CEC system and additional features for further improvement.

Basic features We use a set of nine basic features: the given character, the preceding and following two characters, and the first two and last two characters in the word. These are the same features used by CEC, except that CEC does not include characters beyond the word boundary, while we consider space characters as well as characters from the previous and next words.

Ngram features We extract sequences of characters corresponding to the current character and the following and previous two, three, or four characters. We refer to these sequences as bigrams, trigrams, or 4-grams, respectively. These are an extension of the basic features and allow the model to look beyond the context of the current word.

3.4 Maximum Likelihood Estimate (MLE)

We implemented another approach for error correction based on a word-level maximum likelihood model. The MLE method uses a unigram model which replaces each input word with its most likely correct word based on counts from the training data. The intuition behind MLE is that it can easily correct frequent errors; however, it is quite dependent on the training data.

4 Experiments

4.1 Model Evaluation

Setup The training data was extracted to generate the form described in Section 3.1, using the Sclite tool (Fiscus, 1998) to align the input and reference sentences. A speech effect handling step was applied as a preprocessing step to all models.

[3] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007). For more information on Arabic orthography in NLP, see (Habash, 2010).

This step removes redundant repetitions of characters in sequence, e.g., كتيييير *ktyyyyyr* 'veeeeery'. The same speech effect handling was applied by Eskander et al. (2013).

For classification, we used the SVM implementation in YamCha (Kudo and Matsumoto, 2001), and trained with different variations of the features described above. Default parameters were selected for training (c=1, quadratic kernel, and context window of +/- 2).

In all results listed below, the baseline corresponds to the do-nothing baseline of the input text.

Metrics Three evaluation metrics are used. The word-error-rate **WER** metric is computed by summing the total number of word-level substitution errors, insertion errors, and deletion errors in the output, and dividing by the number of words in the **reference**. The correct-rate **Corr** metric is computed by dividing the number of correct output words by the total number of words in the reference. These two metrics are produced by Sclite (Fiscus, 1998), using automatic alignment. Finally, the accuracy **Acc** metric, used by Eskander et al. (2013), is a simple string matching metric which enforces a word alignment that pairs words in the reference to those of the output. It is calculated by dividing the number of correct output words by the number of words in the **input**. This metric assumes no split errors in the data (a word incorrectly split into two words), which is the case in the data we are working with.

Character-level Model Evaluation The performance of the generalized spelling correction model (GSEC) on the dev data is presented in the first half of Table 4. The results of the Eskander et al. (2013) CEC system are also presented for the purpose of comparison. We can see that using a single classifier, the generalized model is able to outperform CEC, which relies on a cascade of classifiers ($p = 0.03$ for the basic model and $p < 0.0001$ for the best model, GSEC+4grams).[4]

Model Combination Evaluation Here we present results on combining GSEC with the MLE component (GSEC+MLE). We combine the two models in cascade: the MLE component is applied to the output of GSEC. To train the MLE model, we use the word pairs obtained from the original training data, rather than from the output of GSEC. We found that this configuration allows us to include a larger sample of word pair errors for learning, because our model corrects many errors, leaving fewer example pairs to train an MLE post-processor. The results are shown in the second half of Table 4.

Approach	Corr%/WER	Acc%
Baseline	75.9/24.2	76.8
CEC	88.7/11.4	90.0
GSEC	89.7/10.4*	90.3*
GSEC+2grams	90.6/9.5*	91.2*
GSEC**+4grams**	**91.0/9.2***	**91.6***
MLE	89.7/10.4	90.5
CEC + MLE	90.8/9.4	91.5
GSEC+MLE	91.0/9.2	91.3
GSEC**+4grams+ MLE**	**91.7/8.3***	**92.2***

Table 4: Model Evaluation. GSEC represents the generalized character-level model. CEC represents the character-level-edit classification model of Eskander et al. (2013). Rows marked with an asterisk (*) are statistically significant compared to CEC (for the first half of the table) or CEC+MLE (for the second half of the table), with $p < 0.05$.

We first observe that MLE improves the performance of both CEC and GSEC. In fact, CEC+MLE and GSEC+MLE perform similarly ($p = 0.36$, not statistically significant). When adding features that go beyond the word boundary, we achieve an improvement over MLE, GSEC+MLE, and CEC+MLE, all of which are mostly restricted within the boundary of the word. The best GSEC model outperforms CEC+MLE ($p < 0.0001$), achieving a **WER** of 8.3%, corresponding to 65% reduction compared to the baseline. It is worth noting that adding the MLE component allows Eskander's CEC to recover various types of errors that were not modeled previously. However, the contribution of MLE is limited to words that are in the training data. On the other hand, because GSEC is trained on character transformations, it is likely to generalize better to words unseen in the training data.

Results on Test Data Table 5 presents the results of our best model (GSEC+4grams), and best model+MLE. The latter achieves a 92.1% **Acc** score. The **Acc** score reported by Eskander et al. (2013) for CEC+MLE is 91.3% . The two results are statistically significant ($p < 0.0001$) with respect to CEC and CEC+MLE respectively.

Approach	Corr%/WER	Acc%
Baseline	74.5/25.5	75.5
GSEC+4grams	90.9/9.1	91.5
GSEC**+4grams+ MLE**	**91.8/8.3**	**92.1**

Table 5: Evaluation on test data.

[4]Significance results are obtained using McNemar's test.

4.2 Error Analysis

To gain a better understanding of the performance of the models on different types of errors and their interaction with the MLE component, we separate the words in the dev data into: (1) words seen in the training data, or in-vocabulary words **(IV)**, and (2) out-of-vocabulary **(OOV)** words not seen in the training data. Because the MLE model maps every input word to its most likely gold word seen in the training data, we expect the MLE component to recover a large portion of errors in the IV category (but not all, since an input word can have multiple correct readings depending on the context). On the other hand, the recovery of errors in OOV words indicates how well the character-level model is doing independently of the MLE component. Table 6 presents the performance, using the **Acc** metric, on each of these types of words. Here our best model (GSEC+4grams) is considered.

	#Inp Words	Baseline	CEC+MLE	GSEC+MLE
OOV	3,289 (17.2%)	70.7	76.5	**80.5**
IV	15,832 (82.8%)	78.6	**94.6**	**94.6**
Total	19,121 (100%)	77.2	91.5	**92.2**

Table 6: Accuracy of character-level models shown separately on out-of-vocabulary and in-vocabulary words.

When considering words seen in the training data, CEC and GSEC have the same performance. However, when considering OOV words, GSEC performs significantly better ($p < 0.0001$), verifying our hypothesis that a generalized model reduces dependency on training data. The data is heavily skewed towards IV words (83%), which explains the generally high performance of MLE.

We performed a manual error analysis on a sample of 50 word errors from the IV set and found that all of the errors came from gold annotation errors and inconsistencies, either in the dev or train. We then divided the character transformations in the OOV words into four groups: (1) characters that were unchanged by the gold (***X-X*** transformations), (2) character transformations modeled by CEC (***X-Y CEC***), (3) character transformations not modeled by CEC, and which include all phenomena that were only partially modeled by CEC (***X-Y not CEC***), and (4) complex errors. The character-level accuracy on each of these groups is shown in Table 7.

Both CEC and GSEC do much better on the second group of character transformations (that is, ***X-Y CEC***) than on the third group (***X-Y not CEC***). This is not surprising because the former transformations correspond to phenomena that are most common in the training data. For GSEC, they are learned automatically, while for CEC they are selected and modeled explicitly. Despite this fact, GSEC generalizes better to OOV words. As for the third group, both CEC and GSEC perform more poorly, but GSEC corrects more errors (43.48% vs. 31.68% accuracy). Finally, CEC is better at recognizing complex errors, which, although are not modeled explicitly by CEC, can sometimes be corrected as a result of applying multiple classifiers in cascade. Dealing with complex errors, though there are few of them in this dataset, is an important direction for future work, and for generalizing to other datasets, e.g., (Zaghouani et al., 2014).

Type	#Chars	Example	CEC	GSEC
X-X	16502	*m-m, space-space*	99.25	**99.33**
X-Y *(CEC)*	609	*ħ-h, h-ħ, Á-A A-Á, y-ý*	80.62	**83.09**
X-Y *(not CEC)*	161	*t-θ , del{w} n-ins{space}*	31.68	**43.48**
Complex	32	*n-ins{A}{m}*	**37.5**	15.63

Table 7: Character-level accuracy on different transformation types for out-of-vocabulary words. For complex transformations, the accuracy represents the complex category recognition rate, and not the actual correction accuracy.

5 Conclusions

We showed that a generalized character-level spelling error correction model can improve spelling error correction on Egyptian Arabic data. This model learns common spelling error patterns automatically, without guidance of manually selected or language-specific constraints. We also demonstrate that the model outperforms existing methods, especially on out-of-vocabulary words.

In the future, we plan to extend the model to use word-level language models to select between top character predictions in the output. We also plan to apply the model to different datasets and different languages. Finally, we plan to experiment with more features that can also be tailored to specific languages by using morphological and linguistic information, which was not explored in this paper.

Acknowledgments

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

References

Mohamed I. Alkanhal, Mohammed 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 & Language Processing*, 20:2111–2122.

Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France, July.

Chiraz Ben Othmane Zribi and Mohammed Ben Ahmed. 2003. Efficient Automatic Correction of Misspelled Arabic Words Based on Contextual Information. In *Proceedings of the Knowledge-Based Intelligent Information and Engineering Systems Conference*, Oxford, UK.

Andrew Carlson and Ian Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*.

Daniel Dahlmeier and Hwee Tou Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 568–578.

Huizhong Duan, Yanen Li, ChengXiang Zhai, and Dan Roth. 2012. A discriminative model for query spelling correction with latent structural svm. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1511–1521, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing spontaneous orthography. In *The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '13.

Jon Fiscus. 1998. Sclite scoring package version 1.5. US National Institute of Standard Technology (NIST), URL http://www. itl. nist. gov/iaui/894.01/tools.

Michael Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 163–171, Los Angeles, California, June.

Andrew R. Golding and Dan Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Nizar Habash and Ryan M. Roth. 2011. Using deep morphology to improve automatic error detection in arabic handwriting recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 875–884, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Mona Diab, and Owen Rambow. 2012. Conventional orthography for dialectal Arabic. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological Analysis and Disambiguation for Dialectal Arabic. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Bassam Haddad and Mustafa Yaseen. 2007. Detection and Correction of Non-Words in Arabic: A Hybrid Approach. *International Journal of Computer Processing Of Languages (IJCPOL)*.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2013. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5.

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

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Yanen Li, Huizhong Duan, and ChengXiang Zhai. 2012. A generalized hidden markov model with discriminative training for query spelling correction. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 611–620, New York, NY, USA. ACM.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84, Seattle, Washington, USA, October. Association for Computational Linguistics.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012a. Egyptian Arabic Treebank DF Part 1 V2.0. LDC catalog number LDC2012E93.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012b. Egyptian Arabic Treebank DF Part 2 V2.0. LDC catalog number LDC2012E98.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012c. Egyptian Arabic Treebank DF Part 3 V2.0. LDC catalog number LDC2012E89.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012d. Egyptian Arabic Treebank DF Part 4 V2.0. LDC catalog number LDC2012E99.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Dalila Tabassi, and Michael Ciul. 2012e. Egyptian Arabic Treebank DF Part 5 V2.0. LDC catalog number LDC2012E107.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for esl correction tasks. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, Portland, Oregon, 6. Association for Computational Linguistics.

Khaled Shaalan, Rana Aref, and Aly Fahmy. 2010. An approach for analyzing and correcting spelling errors for non-native Arabic learners. *Proceedings of Informatics and Systems (INFOS)*.

L Venkata Subramaniam, Shourya Roy, Tanveer A Faruquie, and Sumit Negi. 2009. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122. ACM.

Sebastian van Delden, David B. Bracewell, and Fernando Gomez. 2004. Supervised and unsupervised automatic spelling correction algorithms. In *Information Reuse and Integration, 2004. Proceedings of the 2004 IEEE International Conference on*, pages 530–535.

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 9th edition of the Language Resources and Evaluation Conference*.