Utilizing Character and Word Embeddings for Text Normalization with Sequence-to-Sequence Models

Daniel Watson, Nasser Zalmout and Nizar Habash
Computational Approaches to Modeling Language Lab
New York University Abu Dhabi
{daniel.watson,nasser.zalmout,nizar.habash}@nyu.edu

Abstract

Text normalization is an important enabling technology for several NLP tasks. Recently, neural-network-based approaches have outperformed well-established models in this task. However, in languages other than English, there has been little exploration in this direction. Both the scarcity of annotated data and the complexity of the language increase the difficulty of the problem. To address these challenges, we use a sequence-to-sequence model with character-based attention, which in addition to its self-learned character embeddings, uses word embeddings pre-trained with an approach that also models subword information. This provides the neural model with access to more linguistic information especially suitable for text normalization, without large parallel corpora. We show that providing the model with word-level features bridges the gap for the neural network approach to achieve a state-of-the-art F_1 score on a standard Arabic language correction shared task dataset.

1 Introduction

Text normalization systems have many potential applications – from assisting native speakers and language learners with their writing, to supporting NLP applications with sparsity reduction by cleaning large textual corpora. This can help improve benchmarks across many NLP tasks.

In recent years, neural encoder-decoder models have shown promising results in language tasks like translation, part-of-speech tagging, and text normalization, especially with the use of an attention mechanism. In text normalization, however, previous state-of-the-art results rely on developing many other pipelines on top of the neural model. Furthermore, such neural approaches have barely been explored for this task in Arabic, where previous state-of-the-art systems rely on combining various statistical and rule-based approaches.

We experiment with both character embeddings and pre-trained word embeddings, using several embedding models, and we achieve a state-of-the-art F_1 score on an Arabic spelling correction task.

2 Related Work

The encoder-decoder neural architecture (Sutskever et al., 2014; Cho et al., 2014) has shown promising results in text normalization tasks, particularly in character-level models (Xie et al., 2016; Ikeda et al., 2016). More recently, augmenting this neural architecture with the attention mechanism (Bahdanau et al., 2014; Luong et al., 2015) has dramatically increased the quality of results across most NLP tasks. However, in text normalization, state-of-the-art results involving attention (e.g., Xie et al. 2016) also rely on several other models during inference, such as language models and classifiers to filter suggested edits. Neural architectures at the word level inherently rely on multiple models to align and separately handle out-of-vocabulary (OOV) words (Yuan and Briscoe, 2016).

In the context of Arabic, we are only aware of one attempt to use a neural model for end-to-end text normalization (Ahmadi, 2017), but it fails to beat all baselines reported later in this paper. Arabic diacritization, which can be considered forms of text normalization, has received a number of neural efforts (Belinkov and Glass, 2015; Abandah et al., 2015). However, state-of-the-art approaches for end-to-end text normalization rely on several additional models and rule-based approaches as hybrid models (Pasha et al., 2014; Rozovskaya et al., 2014; Nawar, 2015; Zalmout and Habash, 2017), which introduce direct human knowledge into the system, but are limited to correcting specific mistakes and rely on expert knowledge to be developed.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 837–843
Brussels, Belgium, October 31 - November 4, 2018.

3 Approach

Many common mistakes addressed by text normalization occur fundamentally at the character level. Moreover, the input data tends to be too noisy for a word-level neural model to be an end-to-end solution due to the high number of OOV words. In Arabic, particularly, mistakes may range from simple orthographic errors (e.g., positioning of Hamzas) and keyboard errors to dialectal code switching and spelling variations, making the task more challenging than a generic language correction task. We opt for a character-level neural approach to capture these highly diverse mistakes. While this method is less parallelizable due to the long sequence lengths, it is still more efficient due to the small vocabulary size, making inference and beam search computationally feasible.

3.1 Neural Network Architecture

Given an input sentence $\mathbf{x}$ and its corrected version $\mathbf{y}$, the objective is to model $P(\mathbf{y}|\mathbf{x})$. The vocabulary can consist of any number of unique tokens, as long as the following are included: a padding token to make input batches have equal length, the two canonical start-of-sentence and end-of-sentence tokens of the encoder-decoder architecture, and an OOV token to replace any character outside the training data during inference. Each character x_i in the source sentence $\mathbf{x}$ is mapped to the corresponding d_{ce}-dimensional row vector $\mathbf{c_i}$ of a learnable $d_{voc} \times d_{ce}$ embedding matrix, initialized with a random uniform distribution with mean 0 and variance 1. For the encoder, we learn d-dimensional representations for the sentence with two gated recurrent unit (GRU) layers (Cho et al., 2014), making only the first layer bidirectional following Wu et al. (2016). Like long short-term memory (Hochreiter and Schmidhuber, 1997), GRU layers are well-known to improve the performance of recurrent neural networks (RNN), but are slightly more computationally efficient than the former.

For the decoder, we use two GRU layers along with the attention mechanism proposed by Luong et al. (2015) over the encoder outputs h_i. The initial states for the decoder layers are learned with a fully-connected $\tanh$ layer in a similar fashion to Cho et al. (2014), but we do so from the first encoder output. During training, we use scheduled sampling (Bengio et al., 2015) and feed the d_{ce}-dimensional character embeddings at every time step, but using a constant sampling probability. While tuning scheduled sampling, we found that introducing a sampling probability provided better results than relying on the ground truth, i.e., teacher forcing (Williams and Zipser, 1989). However, introducing a schedule did not yield any improvement as opposed to keeping the sampling probability constant and unnecessarily complicates hyperparameter search. For both the encoder and decoder RNNs, we also use dropout (Srivastava et al., 2014) on the non-recurrent connections of both the encoder and decoder layers during training.

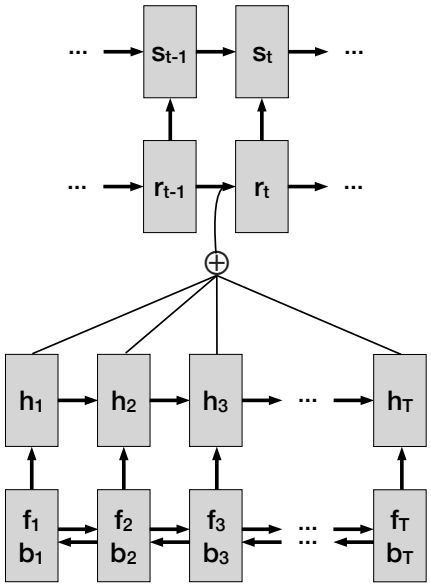

Figure 1: Illustration of the encoder and decoder recurrent layers.

The decoder outputs are fed to a final softmax layer that reshapes the vectors to dimension d_{voc} to yield an output sequence $\mathbf{y}$. The loss function is the canonical cross-entropy loss per time step averaged over the y_i.

3.2 Word Embeddings

To address the challenge posed by the small amount of training data, we propose adding pretrained word-level information to each character embedding. To learn these word representations, we use FastText (Bojanowski et al., 2016), which extends Word2Vec (Mikolov et al., 2013) by adding subword information to the word vector. This is very suitable for this task, not only because many mistakes occur at the character level,

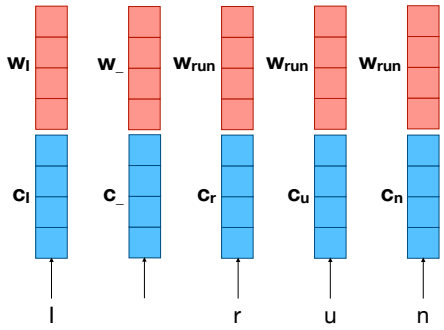

Figure 2: Illustration showing how the character embeddings are enriched with word-level features.

but also because FastText handles almost all OOVs by omitting the Word2Vec representation and simply using the subword-based representation. It is possible, yet extremely rare that FastText cannot handle a word– this can occur if the word contains an OOV n-gram or character that did not appear in the data used to train the embeddings. It should also be noted that these features are only fed to the encoder layer; the decoder layer only receives d_{ce}-dimensional character embeddings as inputs, and the softmax layer has a d_{voc}-dimensional output.

Each character embedding $\mathbf{c_i}$ is replaced by the concatenation $[\mathbf{c_i}; \mathbf{w_j}]$ before being fed to the encoder-decoder model, where $\mathbf{w_j}$ is the d_{we}-dimensional word embedding for the word in which $\mathbf{c_i}$ appears in. This effectively handles almost all cases except white spaces, in which we just always append a d_{we}-dimensional vector $\mathbf{w\_}$ initialized with a random uniform distribution of mean 0 and variance 1. For OOVs, we just append the whitespace embedding $\mathbf{w\_}$ to the word's characters.

3.3 Inference

During inference, the decoder layer uses a beam search, keeping a fixed number (i.e., beam width) of prediction candidates with the highest log-likelihood at each step. Whenever an "end-of-sentence" token is produced in a beam, the decoder stops predicting further tokens for it. We then pick the individual beam with the highest overall log-likelihood as our prediction. As a final step, we reduce text sequences that are repeated six or more times to a threshold of 5 repetitions (e.g., `"abababababab"` → `"abababab"`). This helps address rare cases where the decoder misbehaves and produces non-stop repetitions of text, and also helps avoid extreme running times for the NUS MaxMatch scorer (Dahlmeier and Ng, 2012), which we use for evaluation and comparison purposes.

Baseline	P	R	F_1
MLE	77.08	41.56	54.00
MADAMIRA	77.47	32.10	45.39
MLE then MADAMIRA	64.38	38.42	48.12
MADAMIRA then MLE	73.55	44.61	**55.54**

Table 1: Baselines scores on the QALB 2014 shared task development set.

4 Evaluation

4.1 Data

We tested the proposed approach on the QALB dataset, a corpus for Arabic language correction and subject of two shared tasks (Zaghouani et al., 2014; Mohit et al., 2014; Rozovskaya et al., 2015). Following the guidelines of both shared tasks, we only used the training data of the QALB 2014 shared task corpus (19,411 sentences). Similarly, the validation dataset used is only that of the QALB 2014 shared task, consisting of 1,017 sentences. We use two blind tests, one from each year. During training, we only kept sentences of up to 400 characters in length, resulting in the loss of 172 sentences.

4.2 Metric

Like in the QALB shared tasks, we use the *MaxMatch* scorer to compute the optimal word-level edits that map each source sentence to its respective corrected sentence. We report the F_1 score of these edits against those provided in the gold data by the same tool. We compare against the best reported system in the QALB 2014 shared task test set (CLMB) (Rozovskaya et al., 2014), as well as the best in the QALB 2014 shared task development and the QALB 2015 shared task test sets (CUFE) (Nawar, 2015).

4.3 Baselines

We consider two different baselines. The first is the output from MADAMIRA (Pasha et al., 2014), a tool for morphological analysis and disambiguation of Arabic. The second is using maximum likelihood estimation (MLE) based on the counts of the MaxMatch gold edits from the training data; that is, each word or phrase gets either replaced

Model	P	R	F_1
Wide embeds	80.80	59.80	68.73
+ preprocessing	79.63	58.81	67.57
Narrow embeds	80.00	62.46	**70.15**
+ preprocessing	80.25	57.80	67.20
Concat embeds	80.74	61.10	69.56
+ preprocessing	79.81	58.28	67.37
CUFE (Nawar, 2015)	–	–	68.72

Table 2: System scores on the QALB 2014 shared task development dataset for the different FastText embeddings.

Model	P	R	F_1
No word embeds	81.55	56.13	66.49
Word2Vec	**82.16**	51.53	63.33
FastText	80.00	**62.46**	**70.15**

Table 3: Ablation tests on the QALB 2014 shared task development dataset. All settings used no preprocessing and narrow word embeddings.

or kept as is, depending on the most common action in the training data. We found that, unlike Eskander et al. (2013) suggested, first using MADAMIRA and then MLE yields better results than composing these in the reverse order. The results are presented in Table 1.

4.4 Model Settings

In all experiments, we used batch and character embedding sizes of $b = d_{ce} = 128$, hidden layer size of $d = 256$, dropout probability of 0.1, decoder sampling probability of 0.35, and gradient clipping with a maximum norm of 10. When running all the trained models during inference, we used a beam width of 5. We used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.0005, ϵ=$1 \cdot 10^{-8}$, β_1=0.9, and β_2=0.999, and trained the model for 30 epochs. We report three different setups with FastText word embeddings: narrow, wide, and the concatenation of both. For each of these, we report results on two separately trained models: one without preprocessing, and one with MADAMIRA and then MLE preprocessing to the inputs. We also report an ablation study where we choose the best of these six trained models and compare against two separately trained models with identical setups, but using Word2Vec and no word-level features, respectively.

All the word embeddings used are of dimension $d_{we} = 300$, and were trained with a single epoch over the Arabic Gigaword corpus (Parker et al., 2011). In the experiments including preprocessing, the respective word vectors were obtained from Gigaword preprocessed with MADAMIRA. The narrow and wide word embeddings were trained with context windows of sizes 2 and 5, respectively. All other hyperparameters were kept to the default FastText values, except the minimum n-gram size, which was reduced from 3 to 2 to compensate for single-character prefixes and suffixes that appear in Arabic when omitting the short vowels (Erdmann et al., 2018).

4.5 Results

Development set results are shown in Tables 2 and 3, test set results in Table 4. In all models, training without preprocessing consistently yielded better results than their analogues with the inputs pre-fed to MADAMIRA and then MLE. All the FastText embeddings setups with no preprocessing outperformed the previous state-of-the-art results in the development dataset. We hypothesize that this is occurs because the model has access to more examples of errors to normalize during training, thereby increasing performance. The best performing model was that with the narrow word embeddings; consistent with the results of Zalmout et al. (2018) showing the superior performance of narrow word embeddings over both wide embeddings and the concatenation of both. This is justified by Goldberg (2015) and Trask et al. (2015), who illustrate that wider word embeddings tend to capture more semantic information, while narrower word embeddings model more syntactic phenomena.

In our ablation study, we compared the performance of the narrow FastText embeddings against narrow Word2Vec embeddings trained over the same Arabic Gigaword corpus with the same hyperparameters, as well as to no word-level embeddings at all. The results, displayed in Table 3, show that using only Word2Vec slightly increases precision but significantly hurts recall. This highlights the effectiveness of using FastText for text normalization, as well as the need to handle OOVs in a noisy context for word-level representations to help in this particular problem. Despite that having OOV cases can help the model by indicating that a word is likely erroneous, this does not provide linguistic information the way FastText does. The narrow FastText embeddings with no prepro-

Model	(2014) F_1	(2015) F_1
MADAMIRA then MLE	55.56	60.98
CLMB (Rozovskaya et al., 2014)	67.91	–
CUFE (Nawar, 2015)	66.68	72.87
Narrow embeds	**70.39**	**73.19**

Table 4: System score on the QALB 2014 and QALB 2015 shared task test datasets.

cessing setup achieved state-of-the-art results in all three datasets, beating all systems in both the 2014 and 2015 QALB shared tasks in F_1 score.

4.6 Error Analysis

We conducted a detailed error analysis of 101 sentences from the development set (6,370 words). The sample contained 1,594 erroneous words (25%). The errors were manually classified in a number of categories, which are presented in Table 5. The Table indicates the percentage of the error type in the whole set of errors as well as the error-specific F_1 and an example. Some common problems, Hamza (glottal stop) and Ta Marbuta (feminine ending), are well handled in our best system. This is due to their commonality in the training data. Other types are less common – dialectal constructions, consonantal switches and Mood/Case. Punctuation is very common, however it is also very idiosyncratic. We also note the presence of a small percentage (0.5%) of gold errors. For more information on Arabic language orthography issues from a computational perspective, see (Buckwalter, 2007; Habash, 2010; Habash et al., 2012).

Gold%	Error Type	F_1	Example
4.8	Ta Marbuta	95.4	كتابه → كتابة
29.8	Hamza	92.8	تأييد → تاييد
10.5	Space	87.5	ما سبب → ماسبب
0.8	Alif Maqsura	83.3	التي → التى
0.7	Repeated Letter	81.8	الرجال → الرجاااال
0.6	Wa of Plurality	66.7	قالوا → قالو
39.3	Punctuation	56.4	NIL → .
2.2	Multiple Types	43.1	القيامة → ألقيامه
1.7	Consonant Switch	41.0	شخص → شحص
1.6	Other Types	38.3	القتل → التقل
2.3	Mood & Case	33.3	مصريين → مصريون
2.8	Dialect	32.8	سيكتب → هيكتب
1.3	Deleted Letter	n/a	سينتصر → سيتصر
1.1	Grammar	n/a	يتجاوزون → يتجاوز
0.5	Gold Error	n/a	التى → اللتى

Table 5: Error analysis on a sample from the QALB 2014 shared task development set, ordered by F_1 score.

5 Conclusion and Future Work

We propose a novel approach to text normalization by enhancing character embeddings with word-level features that model subword information and model syntactic phenomena. This significantly improves the neural model's recall, allowing the correction of more complex and diverse errors. Our approach achieves state-of-the-art results in the QALB dataset, despite it being small and seemingly unsuited for a neural model. Moroever, our neural model is sophisticated enough to not benefit from preprocessing techniques that reduce the number of errors in the data. Our approach is general enough to be implemented for any other text normalization task and does not rely on domain-specific knowledge to develop.

Future directions include expanding the number of training pairs via synthetic data generation, where generative models can potentially add human-like errors to a large, unannotated corpus. Different sequence-to-sequence architectures, such as the Transformer module (Vaswani et al., 2017), could also be explored and researched more exhaustively. The word-level features provided by FastText could also be replaced by separately trained neural approaches that generate word embeddings from a word's characters (e.g., ELMo embeddings, Peters et al. 2018), and could also be fine-tuned towards specific applications. Another interesting direction includes hybrid word-character architectures, where the encoder receives word-level features, while the decoder operates at the character level. We are also interested in applying our approach to other languages and dialects.

Acknowledgment

The second author was supported by the New York University Abu Dhabi Global PhD Student Fellowship program. The support and resources from the High Performance Computing Center at New York University Abu Dhabi are also gratefully acknowledged.

References

Gheith A. Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. 2015. Automatic diacritization of Arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197.

Sina Ahmadi. 2017. Attention-based encoder-decoder networks for spelling and grammatical error correction. Master's thesis, Paris Descartes University, 9.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yonatan Belinkov and James Glass. 2015. Arabic diacritization with recurrent neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2281–2285.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tim Buckwalter. 2007. Issues in Arabic Morphological Analysis. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Alexander Erdmann, Nasser Zalmout, and Nizar Habash. 2018. Addressing Noise in Multidialectal Word Embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing Spontaneous Orthography. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Atlanta, GA.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *CoRR*, abs/1510.00726.

Nizar Habash, Mona Diab, and Owen Rambow. 2012. Conventional Orthography for Dialectal Arabic. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 711–718, Istanbul, Turkey.

Nizar Y Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese text normalization with encoder-decoder model. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, pages 129–137.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47.

Michael Nawar. 2015. CUFE@QALB-2015 shared task: Arabic error correction system. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*, pages 133–137, Beijing, China.

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

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alla Rozovskaya, Houda Bouamor, Nizar Habash, Wajdi Zaghouani, Ossama Obeid, and Behrang Mohit. 2015. The second QALB shared task on automatic text correction for arabic. In *Proceedings of the*

Second Workshop on Arabic Natural Language Processing, pages 26–35.

Alla Rozovskaya, Nizar Habash, Ramy Eskander, Noura Farra, and Wael Salloum. 2014. The Columbia system in the QALB-2014 shared task on Arabic error correction. In *ANLP@EMNLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

Andrew Trask, David Gilmore, and Matthew Russell. 2015. Modeling order in neural word embeddings at scale. *arXiv preprint arXiv:1506.02338*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, ÅĄukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Daniel Jurafsky, and Andrew Y. Ng. 2016. Neural language correction with character-based attention. *CoRR*, abs/1603.09727.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386.

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 *International Conference on Language Resources and Evaluation (LREC 2014)*.

Nasser Zalmout, Alexander Erdmann, and Nizar Habash. 2018. Noise-Robust Morphological Disambiguation for Dialectal Arabic. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Nasser Zalmout and Nizar Habash. 2017. Don't throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.