Attentive Sequence-to-Sequence Learning for Diacritic Restoration of Yorùbá Language Text

Iroro Fred Ọ̀nọ̀mẹ̀ Orife

Niger-Volta Language Technologies Institute

iroro@alumni.cmu.edu

Abstract

Yorùbá is a widely spoken West African language with a writing system rich in tonal and orthographic diacritics. With very few exceptions, diacritics are omitted from electronic texts, due to limited device and application support. Diacritics provide morphological information, are crucial for lexical disambiguation, pronunciation and are vital for any Yorùbá text-to-speech (TTS), automatic speech recognition (ASR) and natural language processing (NLP) tasks. Reframing Automatic Diacritic Restoration (ADR) as a machine translation task, we experiment with two different attentive Sequence-to-Sequence neural models to process undiacritized text. On our evaluation dataset, this approach produces diacritization error rates of less than 5%. We have released pre-trained models, datasets and source-code as an open-source project to advance efforts on Yorùbá language technology.

Index Terms: automatic diacritization, Yorùbá language, neural machine translation, sequence-to-sequence models

1. Introduction

Yorùbá is a tonal language spoken by more than 40 Million people in the countries of Nigeria, Benin and Togo in West Africa. There are an additional million speakers in the African diaspora, making it the most broadly spoken African language outside Africa [1]. The phonology is comprised of eighteen consonants *(b, d, f, g, gb, h, j, k, l, m, n, p, r, s, ṣ, t, w, y)*, seven oral vowel *(a, e, ẹ, ị, o, ọ, u)* and five nasal vowel phonemes *(an, ẹn, in, ọn, un)* with three kinds of tones realized on all vowels and syllabic nasal consonants *(ḿ, ń)* [2]. Accordingly, Yorùbá orthography makes significant use of tonal diacritics to signify tonal patterns, and orthographic diacritics like underdots for various language sounds [3]. For example, ẹ signifies a half-open vowel, while ṣ represents a palatoalveolar fricative [4].

On modern computing platforms, the vast majority of Yorùbá text is written in plain ASCII, without diacritics. This presents grave problems for usage of the standard orthography via electronic media, which has implications for the unambiguous pronunciation of Yorùbá's lexical and grammatical tones by both human speakers and TTS systems. Improper handling of diacritics also degrades the performance of document retrieval via search engines and frustrates every kind of Natural Language Processing (NLP) task, notably machine translation to and from Yorùbá [5]. Finally, correct diacritics are mandatory in reference transcripts for any Automatic Speech Recognition (ASR) task.

Automatic Diacritic Restoration (ADR), which goes by other names such as Unicodification [6] or deASCIIfication [7] is a process which attempts to resolve the ambiguity present in undiacritized text. Table 1 shows diacritized forms for each non-diacritic character.

Undiacritized Yorùbá text has a high degree of ambiguity [3, 8, 9]. Adegbola et al. state that for ADR the "prevailing error factor is the number of valid alternative arrangements of the diacritical marks that can be applied to the vowels and syllabic nasals within the words" [3]. For our training corpus of 1M words, we quantify the ambiguity by the percentage of all words that have diacritics, 85%; the percentage of unique non-diacritized word types that have two or more diacritized forms, 32%, and the lexical diffusion or *LexDif* metric, which conveys the average number of alternatives for each non-diacritized word, 1.47. Further, 64% of all unique, non-diacritized monosyllabic words possess multiple diacritized forms [10, 11]. When we consider the distribution of ambiguity over grammatical function, we recognize the added difficulty of tasks like the lexical disambiguation of non-diacritized Yorùbá verbs, which are predominantly monosyllabic.

Table 1: *Characters with their non-diacritic forms*

Characters		**Examples**
à á ǎ	**a**	gbà *(spread)*, gba *(accept)*, gbá *(hit)*
è é ẹ ẹ̀ ẹ́	**e**	esé *(cat)*, èsè *(dye)*, ẹsẹ̀ *(foot)*
ì í	**i**	ìlú *(town)*, ilu *(opener)*, ìlù *(drum)*
ò ó ọ ọ̀ ọ́ ǒ	**o**	ọkọ́ *(hoe)*, ọ̀kọ̀ *(spear)*, ọkọ̀ *(vehicle)*
ù ú ǔ	**u**	mu *(drink)*, mù *(sink)*, mú *(sharp)*
ǹ ń n̄	**n**	n *(I)*, ń (continuous aspect marker)
ṣ	**s**	sá *(run)*, ṣá *(fade)*, ṣà *(choose)*

To date, methods to solving Yorùbá ADR have used memory-based or Naïve Bayes classifers on n-gram features. Efforts have focused on the word-level or mixed-models, rather than purely character-level models based on results that indicate "tonal diacritics can simply not be solved on the level of the grapheme" [9]. With some studies using corpora as small as 5k words from a 3.5k lexicon [6], the dearth of accurate diacritized electronic text has been the object of study as well as a limiting factor on progress [3].

Recently, neural machine translation (NMT) has emerged as the state-of-the-art approach to solving automatic inter-language machine translation. Expressing ADR as a machine translation problem, we treat undiacritized text and diacritized text as source and target languages respectively in a NMT formulation. Our contributions are as follows:

- We propose two different NMT approaches, using soft-attention and self-attention sequence-to-sequence (seq2seq) models [12, 13], to rectify undiacritized Yorùbá text.
- We release the training datasets, pre-trained models, source code and reproducible results into the public domain as an open-source project.

This paper is organized as follows. Section 2 reviews previous work in Yorùbá ADR. In section 3, we review algorithms for seq2seq learning. In section 4 and 5, we present our experimental setup and results. In section 6, we conclude with a review of applications and future directions.

2. Related work

ADR is an active field of study with on-going efforts in a wide variety of languages, including Czech, Polish, Romanian and Hungarian [14, 15, 16], Turkish [7], Arabic [17, 18, 19, 20], Māori [21], Uyghur [22], Urdu [23], Vietnamese [24, 25, 26], as well as West African languages like Igbo [27] and Yorùbá [3, 6, 8, 9].

ADR investigations have relied on ruled-based morphological analyzers [14, 23], or used a variety of statistical learning techniques including conditional random fields (CRFs) [20], support vector machines (SVMs) [25], Hidden Markov Models (HMMs) [17], finite state transducers [19], n-gram models [23] and recurrent neural networks [18]. Parallel corpora, statistical machine translation (SMT) techniques have also been used [14, 20]. We will focus our review specifically on Yorùbá and ADR techniques using NMT sequence-to-sequence models.

2.1. Yorùbá ADR

The complexity of the Yorùbá ADR task is comparable to languages like Vietnamese where over 90% of words contain diacritics, of which some 80% are ambiguous without diacritics [24, 26]. In a contrastive study of seven resource-scarce African languages and better resourced European and Asian languages, DePauw et al. [9] report *LexDif* scores of 1.26 and 1.21 respectively for Yorùbá and Vietnamese, citing the similar disambiguation tasks due to the marking of phonemic variants and tonal characteristics. In contrast, French and Romanian corpora, where approximately a fifth and two-fifths of words contain diacritics, received *LexDif* scores of 1.04 and 1.05 respectively [28].

For Yorùbá, DePauw et al. trained a Tilburg Memory-Based Learner (TiMBL) classifier that predicted the correct diacritic based on the local graphemic context. Memory-based learning is a variant of the classical k-Nearest Neighbors (*k*-NN) approach to classification, common for NLP tasks. Trained on a corpus of 65.6k words, their best word-level accuracy was 76.8%.

Scannell [6] implemented a Naïve Bayes classifier using word and character-level models. For character-level prediction, each ambiguous character was treated as a separate classification problem, disregarding any previous diacritization. For word-level, two lexicon lookup methods were used: The first replaces ambiguous words with the most frequent candidate, while the second uses a bigram model to determine the output. The corpus was a meager 5k words from a 3.5k word lexicon. Word-level models significantly outperformed character-level models with a top word-level accuracy of 75.2%.

Investigating the effect of corpus size on ADR accuracy, Adegbola et al. [3] used a Naïve Bayes classifier based on word trigram probabilities and linear interpolation for smoothing. Trained on 100K words with a 7.5k lexicon, the best word-level ADR result was 70.5%.

Asahiah et al. [8] tackle a subset of the full ADR task, uniquely focusing on tone mark restoration, not orthographic diacritics. On a corpus of 250k words, using a TiMBL classifier and syllables as the unit of restoration, they report mean accuracies at the syllable and word-level of 98% and 93% respectively.

Note that in each of the four studies, the evaluation corpora are private and the results from each are not directly comparable to the others.

2.2. Sequence-to-sequence learning for ADR

There have been only two investigations, to our knowledge, that use sequence-to-sequence learning for ADR tasks. Pham et al. [24] compared the performance of a standard RNN Encoder-Decoder pair on a Vietnamese ADR task. The results on a dataset of 180k sentence pairs yielded accuracy scores of 96.15%, in comparison to 97.32% for the state-of-the-art phrase-based approach for Vietnamese.

For Uyghur language text normalization, Tursun et al. [22] use a non-attentive sequence-to-sequence model for character-level restoration. On a small corpus of 1372 words (226 sentences), they report accuracy of 65.7% for the sequence-to-sequence model and in comparison, 64.9%, for a noisy channel model.

3. Sequence-to-sequence learning

Neural machine translation (NMT) is a recent approach to machine translation that uses models belonging to a family of encoder-decoders neural networks, trained to maximize the probability of a correct translation given a source sentence [29, 30]. Recurrent neural networks (RNNs) are a common choice for encoders and decoders because they can learn a probability distribution over a sequence of symbols by being trained to predict the next symbol in the sequence [30].

In the basic *RNN Encoder-Decoder* architecture, an RNN-Encoder operates on a variable-length source sequence $\mathbf{x} = (x_1, ..., x_{T_x})$ to generate a fixed-length summary context vector $\mathbf{c}$ from its sequence of hidden states. At time step t, the encoder's hidden state $h_t^e \in \mathbb{R}^n$ is updated by

$$h_t^e = f_{enc}\left(x_t, \mathbf{h}_{t-1}^e\right) \tag{1}$$

and

$$\mathbf{c} = g_{enc}\left(\{\mathbf{h}_1^e, ..., \mathbf{h}_{T_x}^e\}\right) \tag{2}$$

f_{enc} and g_{enc} are recurrent non-linear activation functions, as simple as a logistic sigmoid or as complex as a gated recurrent unit (GRU) or long short-term memory (LSTM) unit [12]. The RNN-Decoder is another RNN trained to generate an output sequence $\mathbf{y} = (y_1, ..., y_{T_y})$, by predicting the next word y_t from previously predicted words, its hidden state $\mathbf{h}^d$ and the context vector $\mathbf{c}$. At time step t, the decoder's hidden state $\mathbf{h}_t^d$ is updated by

$$\mathbf{h}_t^d = f_{dec}\left(y_{t-1}, \mathbf{h}_t^d, \mathbf{c}\right) \tag{3}$$

and the conditional distribution of the next symbol is

$$p(y_t|\{y_1, ..., y_{t-1}\}, \mathbf{x}) = softmax(g_{dec}(\mathbf{h}_t^d)) \tag{4}$$

where f_{dec} is another recurrent non-linear activation function like an LSTM. g_{dec} is a transformation function that returns a vocabulary-sized vector, from which the softmax function outputs the probability of y_t [31]. The joint training objective is to minimize the conditional negative log-likelihood

$$J = -\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_y^{(n)}} log\, p(y_t = y_t^{(n)}|y_{<t}^{(n)}, \mathbf{x}^{(n)}) \tag{5}$$

where N is the number of parallel training sentence pairs, and $\mathbf{x}^{(n)}$ and $y_t^{(n)}$ are the source sentence and tth target symbol in the nth pair respectively [31]. Though initial RNN-Decoders only observed the last encoder hidden state, with context vector $\mathbf{c} = h^e_{T_x}$, variants of this architecture differ in the type of RNN and how the context vector $\mathbf{c}$ is derived [12, 30].

3.1. Soft-attention

The first architecture used in this study is based on the work of Bahdanau et al. [12], which extends the RNN-Encoder-Decoder design with an attention mechanism that allows the decoder to observe *different* source words for each target word. A bi-directional RNN is used for the encoder network to create representations that consider both past and future inputs. The state $\mathbf{h}^e_j$ corresponds to the concatenation of states produced by the forward and backward RNNs: $\mathbf{h}^e_j = [\overrightarrow{h^e_j}; \overleftarrow{h^e_j}]$. At each decoding time step, the context vector $\mathbf{c}_i$ is now computed as a weighted sum of the encoder hidden states

$$c_i = \sum_{j=1}^{T_y} \alpha_{ij} \mathbf{h}^e_j \tag{6}$$

where $\alpha_{ij} = softmax(e_{ij})$ and $e_{ij} = att(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j)$. The magnitude of the attentional weight α_{ij} represents how important the jth source token x_j is to the ith target token y_i. e_{ij} is an unnormalized compatability score between the encoder state $\mathbf{h}^e_j$ (inputs around position j) and the decoder state $\mathbf{h}^d_{i-1}$ (outputs around position i) [12, 32]. This score can be computed most simply as the dot-product between the vectors, where the matrices W transform the encoder and decoder states into a same-sized representation. Bahdanau et al. define att as a small feed-forward neural network with a single hidden layer with v_a as an additional weight matrix [12].

$$att(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) = \begin{cases} \langle W_d \mathbf{h}^d_{i-1}, W_e \mathbf{h}^e_j \rangle & \textbf{dot} \\ v_a^\top tanh(W_d \mathbf{h}^d_{i-1} + W_e \mathbf{h}^e_j) & \textbf{add} \end{cases} \tag{7}$$

The att network is jointly trained end-to-end with the other components of the system by minimizing the conditional negative log-likelihood of the target words. The conditional distribution of the next decoded symbol, as expressed in Equation 4, remains the same. The attention mechanism only affects the computation of the context vector c_i, effectively relieving the encoder from the trouble of encoding all source information into a fixed-length vector. Rather, distributed information throughout the encoder hidden states can be selectively retrieved by the decoder, yielding both performance and scalability benefits.

3.2. Self-attention

The second attentive seq2seq architecture employed in this study aims to improve on limitations of RNNs, i.e. high computational complexity and non-parallelizeable computation. For both the encoder and decoder, the Transformer model, proposed by Vaswani et al. [13], employs stacks of self-attention layers in lieu of RNNs. Intuitively, self-attention, or intra-attention, computes a representation of a single sequence, modeling dependencies between words from the same sequence. It serves the same purpose as the bi-directional RNN, capturing more directly and efficiently, the relevant context for each word in a sequence.

Self-attention layers use multiple attention heads, with each head mapping a source sequence $\mathbf{x} = (x_1, ..., x_{T_x})$ into a new sequence of the same length $\mathbf{z} = (z_1, ..., z_{T_x})$. The source x is linearly transformed into queries, keys and values matrices using parameter matrices W^Q, W^K, W^V for each layer and each attention head. An output element z_i is computed as

$$z_i = \sum_{j=1}^{T_x} \alpha_{ij} (x_j W^V) \tag{8}$$

where as with soft-attention above, $\alpha_{ij} = softmax(e_{ij})$ and

$$e_{ij} = \frac{1}{\sqrt{d_z}} (x_i W^Q)(x_j W^K)^\top \tag{9}$$

The scaled-dot product function in Equation 9 computes a compatibility score e_{ij}, between queries and keys. h parallel heads are used to attend to different parts of the value vectors, concatenating the output of each head to form a single output vector. Because self-attention layers are inherently invariant to sequence ordering, explicit positional encodings are concatenated along with the input. Residual connections also help distribute position information to higher layers.

Overall, the Transformer encoder is composed of 6 identical layers, each combining a self-attention sub-layer with a fully-connected feed-forward network. The decoder has a similar construction, adding a third sub-layer, which does multi-head attention over the output of the encoder stack, i.e. encoder-decoder attention. The joint training objective remains the same as in the previous sequence-to-sequence architectures.

4. Experimental setup

We review our dataset collection and preprocessing methods then discuss tools and techniques for ADR model training.

4.1. Text preparation

To prepare source and target texts for parallel training, we obtained a very small but fully diacritized text from the Lagos-NWU conversational speech corpus by Niekerk, et. al [33]. We also created our own medium-sized corpus by web-crawling the two Yorùbá-language websites with full diacritics, a current events blog and an online Bible.

Collecting data from varied sources this way is resourceful but also introduces covariate shift between the data subsets. This necessitated text preprocessing of all web-crawled text to ensure common character sets with minimal punctuation. We also cleaned up texts to ensure consistent, error-free diacritization, splitting lines on full-stops to give one sentence per line. To ensure our splits are drawn from similar distributions, we combined all text, shuffled and split utterances into a ratio of 80%, 10%, 10%, for training, test and dev sets respectively.

Table 2: *Training data subsets*

# words	Source URL	Description
24,868	rma.nwu.ac.za	Lagos-NWU corpus
50,202	theyorubablog.com	language blog
910,401	bible.com	online bible webite

Next, all characters in the texts were dispossessed of their diacritics. This entailed converting diacritized text to Unicode Normalization Form Canonical Decomposition (NFD) which separates a base character from its diacritics. We then filtered out all the *UnicodeCategory.NonSpacingMark* characters, which house the diacritic modifications to a character. This yielded two sets of text, one stripped of diacritics (the source) and the other with full diacritics (the training target). To better understand the dataset split, we computed a perplexity of 575.6 for the test targets with a language model trained over the training targets [34]. The $\{$source, target$\}$ vocabularies for training and test sets have $\{11857, 18979\}$ and $\{4042, 5641\}$ word types respectively.

4.2. Training

We built the soft-attention and self-attention models with the Python 3 implementation of OpenNMT, an open-source toolkit created by the Klein et al. [35]. Our choice of programming language included practical engineering considerations, including Unicode support and modern asynchronous execution. In a training framework, we needed an expressive, extensible interface, documentation and Tensorboard integration. Our training hardware configuration was a standard AWS EC2 p2.xlarge instance with a NVIDIA K80 GPU, 4 vCPUs and 61GB RAM. Training the various models took place over the course of a few days.

Experiments varied over the dimensionality of the word embedding layers, the number and size of the RNN layers, the attention type ($tanh$, dot) and optimizer's hyperparameters. Soft-attention training usually converged within 50 epochs, using the Adam optimizer with learning rate decay. The Transformer model however needed only 25 epochs. We selected the top 5 models with the best results on the held-out test set in Table 3.

5. Results

To evaluate the performance of our ADR models, we computed the accuracy score as the ratio of correct words restored to all words. We calculate the perplexity of each model's predictions based on the test set targets. The type of RNN or attention did not make much difference in accuracy, with GRUs and $tanh$ attention being as accurate as dot-product attention with LSTMs. We suspect that attention type will have a significant influence on accuracy when training on a much larger corpus.

Table 3: *Training & Test Accuracy and Perplexity*

Attention	Size	RNN	Train%	Test%	PPL
soft + dot	2L 512	LSTM	96.2	90.1	1.68
soft + add	2L 512	LSTM	95.9	90.1	1.85
soft + tanh	2L 512	GRU	96.2	89.7	1.83
soft + tanh	1L 512	GRU	97.8	89.7	1.86
self	6L 512	-	98.5	95.4	1.32

Early in the training process, models exhibited word order errors in addition to partial and incorrect diacritizations. As learning slowed, models from both attention architectures were able to satisfactorily learn long-range dependencies and the correct context for most tonal and orthographic diacritics, with the Transformer model being slightly more accurate.

For example, the verbs **bàjẹ́** (to spoil), or **jùlọ** (to be more than) are discontinuous morphemes, or splitting verbs. In Table 4, the first example shows the model has learnt the diacritics necessary for **lọ** following a previously predicted **jù**. In the second example, we note the ambiguity of the three occurences of the undiacritized **si**, with two diacritized forms, **sì**, **sí**. An examination of the attention weight matrix for this example revealed that the third instance **sí** attends to the previous **sí** and that the first two attend to each other.

Table 4: *A sample diacritization from the test set*

src	emi ni oye ju awon agba lo nitori mo gba eko re
tgt	èmi ni òye jù àwọn àgbà lọ nítorí mo gba ẹ̀kọ́ rẹ
pred	èmi ni òye **jù** àwọn àgbà **lọ** nítorí mo gba ẹ̀kọ́ rẹ
src	emi yoo si si oju mi si juda
tgt	èmi yóò sì sí ojú mi sí ilé júdà
pred	èmi yóò sì sí ojú mi sí ilé júdà

While performing error analyses on the model predictions, we observed inconsistent diacritizations in the training set. This was especially apparent for words with multiple diacritcized characters but a unique diacritized form, for example **nínú** (inside) or **oríṣìíríṣìí** (all kinds of). Errors in the training set lead to incomplete and erratic diacritizations during inference and while a partial restoration may reduce the ambiguity of text for a human reader, it still presents problems for automatic text processing and search applications. Across the full corpus, to detect incorrect variants of both single and multiple diacritized forms, we can first train a word vector model using fastText [36]. Then given a word that was incorrectly predicted in the test set, we can look at it's nearest neighbours and amend the word forms in the training set that are not valid diacritizations.

Lastly, regarding the composition of the dataset (Table 2), numerous sections contained lengthy passages with a literary style or referenced foreign locations and names that were adapted to a characteristic Yorùbá form. To make ADR more suitable as a preprocessing step for end-user TTS, ASR applications or any business usage, it'll be necessary to augment the corpus with more general purpose text.

6. Conclusions

We have presented a practical study of attention-based sequence-to-sequence learning approaches for diacritic restoration. Avenues for future work include evaluating previous approaches to Yorùbá ADR on this new dataset, growing the training corpus and training superior word embeddings.

We foresee many applications for our work. Firstly, it minimizes the amount of manual correction needed to create a high quality text corpus. Secondly, it is important that the largest repositories of Yorùbá language material on the web are not invisible to the standard search engines. Finally, the integration of automatic diacritizers into text editors, web browser extensions and mobile phone keyboards will greatly simplify input tasks, allowing users to enter text in plain ASCII and have the correct orthography appear on the screen.

The source code and dataset will be available at `https://github.com/Niger-Volta-LTI/yoruba-adr`

7. Acknowledgements

We thank lexicographer and linguist, Kọ́lá Túbọ̀sún for kindly reviewing the training corpora and providing feedback.

8. References

[1] Wikipedia. (2004) Yoruba language, wikipedia. [Online]. Available: https://en.wikipedia.org/wiki/Yoruba_language

[2] A. Akinlabi, "The sound system of Yorùbá," *Lawal, N. Sadisu, MNO & Dopamu, A (Eds.) Understanding Yoruba life and culture. Trento: Africa World Press Inc*, pp. 453–468, 2004.

[3] T. Adegbola and L. U. Odilinye, "Quantifying the effect of corpus size on the quality of automatic diacritization of Yorùbá texts," in *Spoken Language Technologies for Under-Resourced Languages*, 2012, pp. 48–53.

[4] J. Wells, "Orthographic diacritics and multilingual computing," *Language problems and language planning*, vol. 24, no. 3, pp. 249–272, 2000.

[5] T. V. Asubiaro, "Effects of diacritics on web search engines," *Journal of Library and Information Sciences*, vol. 12, no. 1, pp. 1–19, 2014.

[6] K. P. Scannell, "Statistical unicodification of African languages," *Language resources and evaluation*, vol. 45, no. 3, p. 375, 2011.

[7] A. Arslan, "Deasciification approach to handle diacritics in Turkish information retrieval," *Information Processing & Management*, vol. 52, no. 2, pp. 326–339, 2016.

[8] F. O. Asahiah, O. A. Odejobi, and E. R. Adagunodo, "Restoring tone-marks in standard Yorùbá electronic text: improved model," *Computer Science*, vol. 18, no. 3, pp. 301–315, 2017. [Online]. Available: https://journals.agh.edu.pl/csci/article/view/2128

[9] G. De Pauw, P. W. Wagacha, and G.-M. De Schryver, "Automatic diacritic restoration for resource-scarce languages," in *International Conference on Text, Speech and Dialogue*. Springer, 2007, pp. 170–179.

[10] A. Oluseye. (2003) Yorùbá: A grammar sketch: Version 1.0. [Online]. Available: https://www.bible.com/bible/911/GEN.1.BMY

[11] I. O. Delano, *A dictionary of Yorùbá monosyllabic verbs*. Institute of African Studies, University of Ife, 1969, vol. 1.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[14] A. Novák and B. Siklósi, "Automatic diacritics restoration for Hungarian," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 2286–2291.

[15] D. Tufiş and A. Ceauşu, "Diac+: A professional diacritics recovering system," *Proceedings of LREC 2008*, pp. 167–174, 2008.

[16] R. Mihalcea and V. Nastase, "Letter level learning for language independent diacritics restoration," in *Proceedings of the 6th conference on Natural language learning-Volume 20*. Association for Computational Linguistics, 2002, pp. 1–7.

[17] M. Khorsheed, "A hmm-based system to diacritize Arabic text," *Journal of Software Engineering and Applications*, vol. 5, p. 124, 2012.

[18] Y. Belinkov and J. R. Glass, "Arabic diacritization with recurrent neural networks." in *EMNLP*, 2015, pp. 2281–2285.

[19] R. Nelken and S. M. Shieber, "Arabic diacritization using weighted finite-state transducers," in *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Association for Computational Linguistics, 2005, pp. 79–86.

[20] T. Schlippe, T. Nguyen, and S. Vogel, "Diacritization as a machine translation problem and as a sequence labeling problem," in *AMTA-2008. MT at work: In Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas*, 2008, pp. 270–278.

[21] J. Cocks and T. T. Keegan, "A word-based approach for diacritic restoration in Māori," in *Proceedings of the Australasian Language Technology Association Workshop 2011*, 2011, pp. 126–130.

[22] O. Tursun and R. Cakici, "Noisy Uyghur text normalization," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 2017, pp. 85–93.

[23] A. Raza and S. Hussain, "Automatic diacritization for Urdu," in *Proceedings of the Conference on Language and Technology*, 2010, pp. 105–111.

[24] T.-H. Pham, X.-K. Pham, and P. Le-Hong, "On the use of machine translation-based approaches for Vietnamese diacritic restoration," *arXiv preprint arXiv:1709.07104*, 2017.

[25] T. A. Luu and K. Yamamoto, "A pointwise approach for Vietnamese diacritics restoration," in *Asian Language Processing (IALP), 2012 International Conference on*. IEEE, 2012, pp. 189–192.

[26] T. N. D. Do, D. B. Nguyen, D. K. Mac, and D. D. Tran, "Machine translation approach for Vietnamese diacritic restoration," in *Asian Language Processing (IALP), 2013 International Conference on*. IEEE, 2013, pp. 103–106.

[27] I. Ezeani, M. Hepple, and I. Onyenwe, "Automatic restoration of diacritics for Igbo language," in *International Conference on Text, Speech, and Dialogue*. Springer, 2016, pp. 198–205.

[28] M. Simard, "Automatic insertion of accents in French text," in *Proceedings of the Third Conference on Empirical Methods for Natural Language Processing*, 1998, pp. 27–35.

[29] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[30] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[31] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[32] D. Britz, A. Goldie, T. Luong, and Q. Le, "Massive exploration of neural machine translation architectures," *arXiv preprint arXiv:1703.03906*, 2017.

[33] D. R. v. Niekerk and E. Barnard, "Tone realisation in a Yorùbá speech recognition corpus," in *Proceedings of the 3rd International Workshop on Spoken Language Technologies for Under-Resourced Languages*, Cape Town, South Africa, 2012, pp. 54–59.

[34] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh international conference on spoken language processing*, 2002.

[35] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proc. ACL*, 2017. [Online]. Available: https://doi.org/10.18653/v1/P17-4012

[36] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.