A Unified Model for Arabizi Detection and Transliteration using Sequence-to-Sequence Models

Ali Shazal, Aiza Usman, Nizar Habash
Computational Approaches to Modeling Language (CAMeL) Lab
New York University Abu Dhabi, UAE
{ali.shazal,aiza.usman,nizar.habash}@nyu.edu

Abstract

While online Arabic is primarily written using the Arabic script, a Roman-script variety called Arabizi is often seen on social media. Although this representation captures the phonology of the language, it is not a one-to-one mapping with the Arabic script version. This issue is exacerbated by the fact that Arabizi on social media is Dialectal Arabic which does not have a standard orthography. Furthermore, Arabizi tends to include a lot of code mixing between Arabic and English (or French). To map Arabizi text to Arabic script in the context of complete utterances, previously published efforts have split Arabizi detection and Arabic script target in two separate tasks. In this paper, we present the first effort on a unified model for Arabizi detection and transliteration into a code-mixed output with consistent Arabic spelling conventions, using a sequence-to-sequence deep learning model. Our best system achieves 80.6% word accuracy and 58.7% BLEU on a blind test set.

1 Introduction

The term *Transliteration* generally refers to the process of mapping the orthographic symbols used in one script into another. The types of mapping schemes can range from strict one-to-one transliterations (Beesley, 1997), to named-entity transliteration which can be partially constrained by spelling conventions of the target language (Al-Onaizan and Knight, 2002; Rosca and Breuel, 2016), and all the way to the spontaneous romanizations used in many places around the world, in competition with local languages and spelling traditions. In the Arab world, the latter of these phenomena is often called Arabizi,[1] Since it is used primarily in social media (SMS and chat), it's notoriously noisy and inconsistent although there are some common conventions. And because it is written in the Roman script, it encourages written linguistic code mixing between Arabic and English (or French). Furthermore, since it is not an official orthography, Arabizi is a resource poor language form compared to Modern Standard Arabic and some of its dialects, where there are more resources written in Arabic script. As such, many researchers work on mapping Arabizi to Arabic script as part of general text normalization. This task includes two components: identifying the language, and mapping it to the target script accordingly. For example, the code-mixed sequence *w aletli Tuesday* 'and she-said-to-me Tuesday' would be mapped to *Tuesday* وقالت لي *wqAlt ly Tuesday*[2] 'and-she-said to-me Tuesday'. While the input and output here have the same number of words, they are not aligned one-to-one as Arabic spelling rules define words boundaries differently, an additional complexity of the task.

There have been a number of efforts in natural language processing (NLP) that worked on this interesting phenomenon using a range of techniques from classical machine learning and n-gram language models (Darwish, 2014; Al-Badrashiny et al., 2014; Eskander et al., 2014) to sequence-to-sequence

[1] **Arabizi** is a cross-lingual portmanteau of the English name of the **Arab**ic language and the Arabic name of the English language, انجليزي *Ainjliyziy* 'Engl**izi**'.

[2] All Arabic script examples are paired with a strict 1-to-1 transliteration in the HSB scheme (Habash et al., 2007).

Proceedings of the Fifth Arabic Natural Language Processing Workshop, pages 167–177
Barcelona, Spain (Online), December 12, 2020

neural models (Guellil et al., 2017; Younes et al., 2018; Younes et al., 2020). The various previous solutions differ in (a) whether and how they modeled the subtask of Arabizi detection (*which words are Arabic written in Roman script, and which are English/French*), (b) what amount of context they use, and (c) how they define the Arabic script target (*whether to allow token merges and splits or follow Arabizi word boundaries*). In this paper we present the first effort on a unified model of Arabizi detection and transliteration into a code-mixed output with consistent Arabic spelling conventions, using neural sequence-to-sequence models. Our best system achieves 80.6% word accuracy and 58.7% BLEU score on a blind test set — 9.9%, and 15.9% absolute improvements, respectively, over a simple but robust baseline. Our system's code is public.[3] But the data must be acquired from the LDC (see Section 5).

We discuss related work in Sections 2, and Arabizi challenges and task definition in Section 3. We present our system architecture and approach in Section 4, and our experiments and results in Section 5.

2 Related Work

2.1 Arabizi Data Collection and Annotation

The various efforts working on Arabizi collected, tagged and mapped different sizes of corpora for different Arabic varieties including Egyptian (Bies et al., 2014; Tobaili, 2016; Chen et al., 2017), Lebanese (Tobaili, 2016), Tunisian (Younes et al., 2015; Masmoudi et al., 2019), and Algerian (Guellil et al., 2017). We focus here on the work of Bies et al. (2014) since it is the largest by far, and because it targets a well-formed conventional orthography for dialectal Arabic, henceforth ARABIZICORPUS. Their data includes over 287K Egyptian Arabizi SMS and chat words that are automatically transliterated and manually validated. The corpus words are tagged for being Foreign, Name, Punctuation, sound, emoji/emoticon or the default Arabic. The Arabic text written in Arabizi is mapped to Arabic script using the conventional orthography for Dialectal Arabic (CODA) (Habash et al., 2012; Eskander et al., 2013; Habash et al., 2018). The CODA convention is close to Standard Arabic orthography while maintaining some of the unique morphological and lexical features of Dialectal Arabic. In addition to using different characters, CODA and Arabizi often use different word boundaries. Moreover, Arabizi is noisy and spontaneous while CODA is intended to be conservative and systematic. In addition to the parallel component, the ARABIZICORPUS includes about 1M Arabizi words that are not tagged or transliterated. An earlier version of this data set was used by Al-Badrashiny et al. (2014) and Eskander et al. (2014). Unfortunately, since the versions used by these earlier efforts do not correspond to the public version of the data set, and are not determinable from it, we cannot compare to them directly. We report in this paper on the latest public version of the data (Chen et al., 2017), and we describe in detail the splits we follow to enable future comparisons with our results (see Section 5.1).

2.2 Pre-neural Models for Arabizi Processing

Chalabi and Gerges (2012) presented a hybrid approach for Arabizi transliteration using manual and learned character mapping rules with a language model for ranking hypotheses. Their work does not address the detection of English words, punctuation, emoticons, and so on. Voss et al. (2014) focus on classifying tokens in Arabizi as Arabic or not. They work on a three-way classification of Moroccan Arabic, French and English. Darwish (2014) was the first effort to model the detection of Arabizi and non-Arabizi words before transliterating the Arabizi text. He employed a two-step system for identification and then conversion. For identification, he used word and sequence-level features with CRF modeling to identify three tags: Arabic, foreign and others. For transliteration, he learned character level mappings from a small parallel corpus and used them to generate alternative mappings, which are filtered using a word trigram language model.

Al-Badrashiny et al. (2014) extended the work of Darwish (2014) on transliteration. They trained a finite state transducer at the character level to generate all possible transliterations for the input Arabizi words to output Arabic words. They then filtered the generated list using a dialectal Arabic morphological analyzer. They picked the best choice for each input word using a word 5-gram language model. Eskander et al. (2014) presented a system that built on top of Al-Badrashiny et al. (2014)'s transliteration

[3] https://github.com/CAMeL-Lab/seq2seq-transliteration-tool

	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
Arabizi Input	Okayyyy	I	confess!	3aiza 'I want'	ageblk 'I bring to you'	el 'the'	swr 'pictures'	3shan 'so as to'	ashooooooofk 'see you'	😃
Token Type	*For*	*For*	*For*	*Ar*	*Ar*	*Ar*	*Ar*	*Ar*	*Ar*	*Em*
ML Input	okayy	i	confess!	3aiza	ageblk	el	swr	3shan	ashoofk	#
ML Output	#	#	#	عايزة	اجيب[-]لك	ال[+]	صور	عشان	اشوفك	#
Postprocessed	Okayyyy	I	confess!	عايزة *ʕAyzħ*	اجيب *Ajyb*	لك *lk*	الصور *AlSwr*	عشان *ʕšAn*	اشوفك *Ašwfk*	😃
Output	😃 اشوفك عشان الصور لك اجيب عايزة okayyyy i confess! Okayyyy I confess! *ʕAyzħ Ajyb lk AlSwr ʕšAn Ašwfk* 😃 'Okayyyy I confess! I want to bring you the pictures so I can seen 😃'									

Figure 1: An example of Arabizi input and target output including the different representations for preprocessing, machine learning input and output, and postprocessing. The token types are Ar (Arabic), For (Foreign) and Em (Emoji).

pipeline. They investigated the issue of processing Arabizi input with code switching using the data from ARABIZICORPUS. They used SVMs and decision trees to identify a larger tag set than Darwish (2014): Arabic, foreign, names, sounds, punctuation and emoticons. For transliteration, they used the Al-Badrashiny et al. (2014) model with more training data.

2.3 Neural Model for Arabizi Processing

Deep learning models, specifically sequence-to-sequence (Seq2Seq) RNN models have shown a lot of success in the task of character-based transliteration for a number of languages (Rosca and Breuel, 2016; Kundu et al., 2018; Dershowitz and Terner, 2020).

In the context of Arabizi, three efforts are particularly notable (Guellil et al., 2017; Younes et al., 2018; Younes et al., 2020). Guellil et al. (2017) and Younes et al. (2018) worked on mapping Algerian Arabizi and Tunisian Arabizi, respectively, to Arabic script. Both used Seq2Seq models at the character level. Younes et al. (2020) redefined the problem as a sequence labeling task using BiLSTM with CRF decoding. All of these approaches were focused on word-level transliteration and did not address issues of code-mixing and Arabizi identification automatically, although they acknowledge these issues. In this paper we present a single unified model that addresses the issues of Arabizi identification and conversion together using a Seq2Seq model.

Although not Arabizi, a recent effort on automatic conversion of Judeo-Arabic text written in Hebrew script to Arabic script is relevant; Dershowitz and Terner (2020) also used a Seq2Seq RNN model and described a number of interesting tricks for addressing length mismatches and forgetfulness in the network. We refer to specific insights from their work in the paper.

Also not Arabizi, but relevant, is the work of Watson et al. (2018), who achieved the current state-of-the-art on Arabic spelling correction against a standard set using Seq2Seq models. Our baseline Seq2Seq set up is inspired by their work.

3 Arabizi Challenges and Task Definition

Our task of mapping code-mixed Egyptian Arabizi and English input into code-mixed Egyptian Arabic and English poses a number of challenges.

Arabizi Noise and Ambiguity While there are some common conventions for Arabizi-to-Arabic mapping, they are not strictly followed by far. This is further exacerbated by typical noisy spelling in spontaneous social media text. As a result, we have many-to-many mappings at the word level. For example, the Arabic word حبيبي *Hbyby* 'my beloved' appears in the ARABIZICORPUS paired with 24 different Arabizi spellings: (in order of frequency) *habibii, hbebe, habibi, 7abiby, 7abibi, hbebee, 7abeby, 7biby, habbii, 7bibi, 7abibii, hbybyy, hbbee, hbbebe, habiibii, habbyy, bbe, 7apipy, 7abiibii, 7abibyy, 7abebe,*

7abby, 3abiibii, and *3abibi*. Similarly, the Arabizi word *arfo* is paired with three different Arabic words: عارفه *ʕArfh*'I know it', قرفوا *qrfwA* 'they made me loathe [something]' and قرفه *qrfh* 'its nastiness'.

We mitigate the noise and variability in modeling the Arabizi-to-Arabic transliteration as a character level sequence-to-sequence process. We also make use of word-level embeddings using FastText (Bojanowski et al., 2017) which models subword units within a bigger word-based context. Additionally, we handle common variations in social media text such as inconsistent capitalization and emphatic repetitions through global lower casing and repetition elision to two characters, e.g., Arabizi *Habiiiiibiiiii* is preprocessed to *habiibii*.

Arabizi-Arabic Mis-alignment The CODA convention we target for Egyptian Arabic does not align word-to-word with Arabizi as explained above. There are both splits and merges of words. In Figure 1, word [5] *ageblk* 'I bring to you' is split into اجيب لك *Ajyb lk*, and words [6] and [7] *el* 'the' *swr* 'pictures' are merged into one: الصور *AlSwr* 'the pictures'. We model these two issues in our work by learning to explicitly map the merge ([+]) and split ([-]) special characters used in the ARABIZICORPUS annotations.

Code-switching and Emoji Within our task, English (or other foreign words) are simply to be mapped to the output, without any attempt to normalize or modify their spelling. As such the task is more oriented towards modeling identification of foreign words – or from a different perspective, detection of Arabizi words. This is not always simple given that some Arabizi words are ambiguous with English words, e.g. the Arabizi word *men* can be the English word 'men' or the Arabic words من *mn* 'from' or مين *myn* 'who'. We model this subtask as mapping to a special symbol (#) used to identify which input words to copy to the output in a postprocessing step.

We handle emojis (e.g., 😃) and emoticons (e.g., :-P) in a comparable way to foreign words. Although for a small set of commonly used emoticons, we use a dictionary and preprocess them into the (#) symbol.

Defining the Task Target The ARABIZICORPUS provides a set of manually assigned tags per each Arabizi word specifying if it is Arabic, Foreign, Name, Sound or Punctuation. In the corpus, all of the words are automatically transliterated and manually validated except for the Foreign word transliterations into Arabic which are not manually validated. This is why Eskander et al. (2014) did not do a joint final evaluation but rather evaluated only on the manually validated words and treated identification of foreign words as a separate problem. In our work, we take a different approach where we define the task target to be a code-mix of Arabic (transliterated from Arabizi) and Foreign words in Roman script. We construct this target reference using the Arabizi word tags and the validated transliterations, by reinserting the foreign input word in place of its automatic non-validated transliteration. We also reinsert Emojis which are mapped to a special token (#) in the target side in the ARABIZICORPUS.

4 System Design

4.1 Approach

Due to the aforementioned complexities and a high number of out-of-vocabulary words, a word-level neural model cannot be used as an end-to-end solution. So, we opt for a character-level approach using sequence-to-sequence models to capture the complexity of the noise, variations and mistakes.

4.2 Data Preprocessing and Postprocessing

Input-side Preprocessing Each Arabizi input utterance goes through the following preprocessing steps: all letters are lowercased, repetition of more than two characters in a word are reduced to two, accented charactes are converted to their unaccented versions in the standard 7-bit ASCII, and all free-standing emojis, emoticons and punctuation are converted to hashtags. These steps can be seen in Figure 1 (ML Input row).

ML Output-side Preprocessing During training only, we convert all the foreign-tagged words to hashtags on the machine learning output side. Since our training input and output are aligned, the conversion ensures that the model learns to identify foreign words and convert them to hashtags. Identical to the

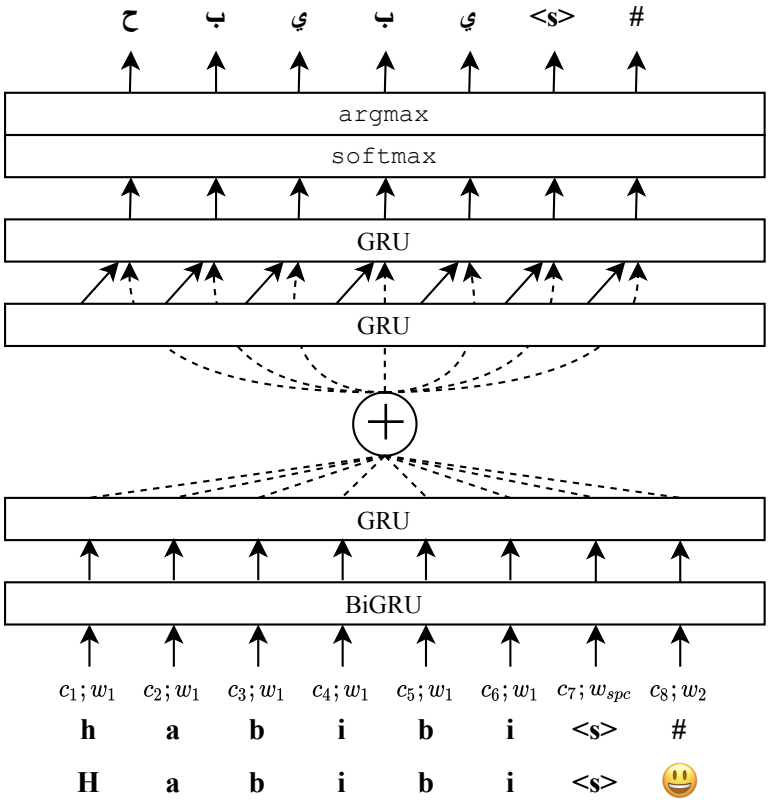

Figure 2: Illustration of the overall Sequence-to-Sequence architecture. The Arabizi (bottom of the figure) is preprocessed before being input into the network. c_i is the character embedding, and w_j is the word embedding.

input-side preprocessing, the ML output-side free-standing emojis, emoticons and punctuation are converted to hashtags during both training and prediction. We illustrate these steps in Figure 1 (ML Output row).

We do not apply AY normalization, i.e., the conversion of different versions of *Alif* and *Ya* to simple *Alif* and *Ya*, respectively (Habash, 2010; El Kholy and Habash, 2012), because it lowered the normalized accuracy by 0.1% absolute on average when we used it.

ML Output-side Postprocessing On the ML output side, we add a postprocessing step that converts hashtags back to the words that were in the source Arabizi. This is only possible if the input and output are aligned, so this step is applied before removing the [+] and [-] tokens. Moreover, to go to the final output, words with the [+] token are merged to the next word and [-] tokens are replaced with white space to split a word into multiple words. These steps can be seen in Figure 1.

4.3 System Architecture

Encoder-Decoder Model We use a character-level sequence-to-sequence architecture following Watson et al. (2018) to model $P(y|x)$ given an input x and a target y. See Figure 2. The encoder consists of two gated recurrent unit (GRU) layers (Cho et al., 2014) with only the first layer being bidirectional following Wu et al. (2016). The decoder also has two GRUs along with the attention mechanism proposed by Luong et al. (2015). The initial states for the decoder layers are learned with a fully-connected `tanh` layer from the first encoder output.

The model uses scheduled sampling (Bengio et al., 2015) with a constant sampling probability, feeding character embeddings concatenated with embeddings of the words the characters appear in at every time step. The model uses dropout (Srivastava et al., 2014) for both encoder and decoder recurrent neural networks on the non-recurrent connections during training. A final `softmax` layer receives the decoder

	CHT			SMS			CHT+SMS		
	Lines	**Words**		**Lines**	**Words**		**Lines**	**Words**	
Train	45,794	185,275	(80.0%)	9,065	45,079	(81.3%)	54,859	230,354	(80.3%)
Dev	4,923	23,302	(10.1%)	485	5,559	(10.0%)	5,408	28,861	(10.1%)
Test	5,583	22,969	(9.9%)	1,070	4,831	(8.7%)	6,653	27,800	(9.7%)

Table 1: Data splits of the corpus into training, development and test sets

output to give the final output sequence y. The loss function is the cross-entropy loss per time step averaged over y_i.

We use beam search during inference with a fixed beam width to predict candidates with the highest log-likelihood at each step. We pick the individual beam with the highest overall log-likelihood as our prediction. As a final step in inference, we reduce repetitions of six or more text sequences to five repetitions. This addresses rare cases where the decoder misbehaves and produces non-stop repetitions of text.

Model Settings We used a learning rate of 0.0001, batch size of 1024 and 2048 for different settings (see LINE2LINE vs WORD2WORD below), batch and character embedding sizes of 256, hidden layer size of 256, two RNN layers, dropout probability of 0.1, decoder sampling probability of 0.35, and gradient clipping with a maximum norm of 11.5. We used the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.0005, $\epsilon = 1 * 10^{-8}$, β_1=0.9, and β_2=0.999, and trained the model for 40 epochs. When running all the trained models during inference, we used a beam width of 5.

All the word embeddings were trained using Fasttext and had dimension 300. They were trained on the unannotated Arabizi text that was part of the ARABIZICORPUS. In the experiments that included preprocessing, we also preprocessed the input-side Arabizi before training Fasttext. All Fasttext hyperparameters were kept to the default except the context window and minimum n-gram size which were both kept to two.

The network hyperparameters above were empirically identified in a series of ∼80 experiments starting with the hyperparamters of Watson et al. (2018) and tuning to achieve the best accuracy and BLEU score on the dev set.

Line2Line vs Word2Word During initial experiments, we observed that the model was rather forgetful and not well performing when fed complete Arabizi input utterances (henceforth, we will refer to this Seq2Seq setting as the LINE2LINE setting). So, we considered the option of working at a WORD2WORD level similar to Younes et al. (2018) and Guellil et al. (2017), with the exception of adding a small context window similar to Mubarak et al. (2019), which is not itself mapped, but only provides contextual information. Contextual tokens such as beginning of sentence <bos>, end of sentence <eos>, beginning of word (under focus) <bow>, and end of word (under focus) <eow> were added to aid the model in this setting. We experimented with context windows of size three, two, and one words, and found +/- one word to be the best.

For example, the three-word Arabizi input *Alf salama hahaha* 'A thousand times safe haha', which is paired with the Arabic الف سلامة ههها *Alf slAmħ hhh*, is turned into the following three separate training input-output pairs:

<bos>	<bow>	Alf	<eow>	salama	→	الف	*Alf*
Alf	<bow>	salama	<eow>	haha	→	سلامة	*slAmħ*
salama	<bow>	haha	<eow>	<eos>	→	هههاا	*hhh*

For WORD2WORD experiments, we use batch size of 2048 because the input sequences are shorter than the LINE2LINE setup for which we used batch size 1024. For both settings, we use the same character-based Seq2Seq architecture discussed above. The only difference is the size of the input in terms of words and whether context is used.

			Accuracy		BLEU		Time (s)	
System	**Context**	**Prep**	**Exact**	**Norm**	**Exact**	**Norm**	**Training**	**Prediction**
MLE	No Context	No	74.7	76.4	46.5	49.6	2	0.1
MLE	No Context	Yes	76.7	78.5	50.0	53.0	2	0.1
Seq2Seq	LINE2LINE	No	70.5	72.4	47.0	50.8	7,837	735
Seq2Seq	LINE2LINE	Yes	74.8	76.9	55.4	59.5	7,800	705
Seq2Seq	WORD2WORD +/-1 Context	No	83.5	85.8	63.5	68.0	30,492	1,893
Seq2Seq	WORD2WORD +/-1 Context	Yes	**84.2**	**86.5**	**64.5**	**69.3**	30,151	1,884

Table 2: Results on the development set with different setups

			Accuracy		BLEU		Time (s)
System	**Context**	**Prep**	**Exact**	**Norm**	**Exact**	**Norm**	**Prediction**
MLE	No Context	Yes	70.7	71.8	42.8	44.6	0.1
Seq2Seq	WORD2WORD +/-1 Context	Yes	**80.6**	**82.5**	**58.7**	**62.6**	1824

Table 3: Results on the test set with different setups

	Handled By		Accuracy		BLEU		Prediction
Setup	**MLE**	**Seq2Seq**	**Exact**	**Norm**	**Exact**	**Norm**	**Time (s)**
MLE Only	85%	0%	76.7	78.5	50.0	53.0	0.1
MLE + Seq2Seq	85%	15%	82.9	85.1	61.9	66.3	270
Seq2Seq Only	0%	100%	84.2	86.5	64.5	69.3	1884

Table 4: Performance comparison of the hybrid, MLE, and Seq2Seq systems on the development set.

5 Evaluation

5.1 Experimental Settings

Data We use the *BOLT Egyptian Arabic SMS/Chat and Transliteration* corpus (ARABIZICORPUS) (Chen et al., 2017; Bies et al., 2014).[4] We split the corpus following the suggestions given by Diab et al. (2013) for Arabic corpora. The documents were first sorted by filename alphabetically. Then, the divisions were applied on the file level, with the divisions being ∼80% training, ∼10% development and ∼10% test, from both chat and SMS subsets of the corpus. The details of the divisions can be seen in Table 1.[5]

Metrics We use two metrics to evaluate our setup. The first, harsher metric, measures word-level accuracy of our Seq2Seq model. It compares the source aligned output of the system, which contains the [+] and [-] separation tokens, to the source aligned gold. In this regard, we measure both the exact-matching accuracy and the AY-normalized accuracy. After we remove the separation tokens, the number of words in the output may differ from the input; this is why we use BLEU (Papineni et al., 2002), which is generally used for translation evaluation, to measure the quality of the transliteration at the end. We measure the BLEU score for both the exact final output and the AY-normalized final output.

Baseline For a baseline, we use Maximum Likelihood Estimate (MLE) to predict the most likely output (Arabic word or hashtag) as seen in training given an input word. This simple baseline experiment is conducted with and without preprocessing.

[4] https://catalog.ldc.upenn.edu/LDC2017T07

[5] Train: CHT_ARZ_{20121228.0001-20150101.0002} and SMS_ARZ_{20120223.0001-20130902.0002}.
Dev: CHT_ARZ_{20120130.0000-20121226.0003} and SMS_ARZ_{20110705.0000-20120220.0000}.
Test: CHT_ARZ_{20150101.0008-20160201.0001} and SMS_ARZ_{20130904.0001-20130929.0000}.

	MLE		Seq2Seq		Input	Reference	Output	English
Acceptable	166	(81%)	195	(95%)				
Correct	155	(76%)	175	(85%)	share3	شارع *šArς*	شارع *šArς*	street
CODA Error	6	(3%)	11	(5%)	3ayznko	عايزينكو *ςAyzynkw*	عايزنكو *ςAyznkw*	we want you
Valid Variant	5	(2%)	9	(4%)	Ok	Ok	أوكيه *Âwkyh*	Ok
Unacceptable	39	(19%)	10	(5%)				
Arabizi	37	(18%)	0	(0%)	fat7na	فتحنا *ftHnA*	fat7na	we opened
Wrong	2	(1%)	10	(5%)	menu:)	menu:)	منه :mnh)	menu:)
Word Total	205	(100%)	205	(100%)				

Table 5: Error analysis summary comparing the MLE baseline and best Seq2Seq model with examples.

5.2 Experimental Results

Development Results Results for the development set are shown in Table 2. We see that the WORD2WORD setup with a context window of +/- one word outperformed the LINE2LINE setup and the baseline MLE. In all setups, preprocessing always helped the model. The best setting (WORD2WORD +/-1 Context + Preprocessing) improves over the strong MLE baseline (with Preprocessing) by 7.5% absolute accuracy and 14.5 BLEU points (in exact matching space).

Blind Test Results For the blind test set, Table 3 presents the results on our best model determined above as and the MLE baseline. The results are consistent with the observations seen in development. The best model improves over the strong MLE baseline by 9.9% absolute accuracy and 15.9 BLEU points (in exact matching space).

Speed vs Quality We would also like to comment on the trade off between speed and quality. In our experiments, we saw that the faster systems had lower quality and vice versa. The MLE system was extremely fast but had much lower quality compared to the WORD2WORD system which was quite slow. Upon a closer look at the results, we saw that the best MLE setup did almost as well on words seen in training as the best Seq2Seq setup, and Seq2Seq easily outperformed MLE on unseen words. This led us to the idea of combining the two systems to create a hybrid system where the MLE model would predict words seen in training and the Seq2Seq model would predict unseen words. This did not affect training time because we used the pre-trained models; however, it decreased prediction time from 31 minutes to 4.5 minutes on the development set. The accuracy and BLEU score of this hybrid system was lower than the WORD2WORD model but higher than the baseline MLE model (See Table 4).

5.3 Error Analysis

We manually classified a sample of 50 development set utterances (50 lines, 205 Arabizi words) from the best MLE model and from the best Seq2Seq model. We grouped the output words into Acceptable and Unacceptable sets. The Acceptable set includes correct matches, as well as acceptable transliteration variants and minor CODA errors; while the Unacceptable set includes Arabizi, and wrong implausible outputs. Table 5 summarizes the results and includes examples for the different categories. Overall, our best Seq2Seq model produced acceptable transliterations for 95% of the Arabizi words, compared with 81% in the MLE model. The largest type of error for MLE was passing the input Arabizi to the output untouched. At the utterance level, 46% of the MLE outputs, and 82% of the Seq2Seq outputs were composed completely of acceptable word outputs. The error analysis demonstrates the power of the Seq2Seq model and its general quality.

For 11 words (5% of total words), the gold reference gave us pause. In almost three-quarters of the cases, there were CODA non-compliant, although plausible, variants, e.g., the word تبقي *tbqy* 'you [f.s.] remain' was rendered as تبقى *tbqý*. The most significant gold reference error was transliterating the English word *invite* to the nonsensical Arabic عنفيت *ςnfyt*.

5.4 Negative Result Experiments

We report next on three different streams of experiments that gave us negative results.

Synthetic Data Inspired by successful results in machine translation, we considered the use of additional synthetic data to mitigate the lack of parallel training data (Sennrich et al., 2016). We initially considered a reverse transliteration model that would allow us to take large amounts of Egyptian Arabic text and map it to Arabizi space. However this was not simple and we quickly noticed a number of issues: (a) most of the available data is not in the chat and SMS genre, and (b) working from Arabic text eliminates examples of Arabizi style code-switching and tokenization. Instead, we opted to use the additional unannotated data we have from the ARABIZICORPUS ($\sim$242K Arabizi utterances containing $\sim$1M words). We used our best model to generate a synthetic Arabizi-Arabic parallel training set, and add it to our initial corpus. The results were negative on all used metrics. We hypothesize that the high degree of noise in the input may have resulted in weak or worse, very noisy, models. This may be consistent with the limited benefits of using the additional corpus as part of FastText pre-trained word embeddings.

Foreign Language Embeddings Intuiting that the quality of Arabizi vs Foreign detection can be improved using additional models of English text as was successfully shown by (Eskander et al., 2014) in their non-neural models. We first took about 1M words (to match the size of data we have for Arabizi) from an English chat and SMS corpus (Chen et al., 2018) and trained another set of word embeddings using FastText; these word embeddings were concatenated to the already existing character embeddings and Arabizi word embeddings in the WORD2WORD setup. This did not show an improvement in terms of the used metrics. In a different experiment, we inserted a random sample of 21K words (or about 10% of training data, to avoid biasing the model too much) into random locations throughout the existing training data and paired those additions with the proper # symbol on the target side. This resulted in a slight decrease in scores, so we did not use it.

Modeling Different Input Sizes and Context Windows Since sequence-to-sequence models are known to struggle with very long input sequences, we considered a solution similar to Dershowitz and Terner (2020), where we break the input and target into smaller sequences of size N words during training. During inference, we do the same for the input, then concatenate the predicted output before evaluation. We considered different sizes of N and found the minimal limit of $N = 1$ words to be the best performing system.

6 Conclusion and Future Work

We presented a character-level Seq2Seq model for Arabizi detection and transliteration together into a code-mixed output with a consistent Arabic spelling convention. Our best system improves over the baseline by 9.9% in word accuracy and 15.9 BLEU points on a blind test set. We report on a number of experiments including some negative results.

In the future, we plan to integrate our best model in an open source toolkit for supporting Arabic NLP. Furthermore, we want to work on Arabizi text from a range of Arabic dialects. Finally, we want to integrate our system into other Arabic NLP applications, such as machine translation from Arabizi to demonstrate the relevance of Arabizi-to-Arabic transliteration.

Acknowledgements

This research was carried out on the High Performance Computing resources at New York University Abu Dhabi (NYUAD). We would like to thank Daniel Watson, Ossama Obeid, Nasser Zalmout and Salam Khalifa from the Computational Approaches to Modeling Language Lab at NYUAD for their help and suggestions throughout this project. We thank Owen Rambow, and the paper reviewers for helpful suggestions.

References

Mohamed Al-Badrashiny, Ramy Eskander, Nizar Habash, and Owen Rambow. 2014. Automatic transliteration of romanized Dialectal Arabic. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 30–38, Ann Arbor, Michigan.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine Transliteration of Names in Arabic Text. In *Proceedings of the Workshop on Computational Approaches to Semitic Languages (CASL)*.

Kenneth R. Beesley. 1997. Romanization, Transcription and Transliteration. http://www.xrce.xerox.com/.

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

Ann Bies, Zhiyi Song, Mohamed Maamouri, Stephen Grimes, Haejoong Lee, Jonathan Wright, Stephanie Strassel, Nizar Habash, Ramy Eskander, and Owen Rambow. 2014. Transliteration of Arabizi into Arabic Orthography: Developing a Parallel Annotated Arabizi-Arabic Script SMS/Chat Corpus. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, Doha, Qatar.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (TACL)*, 5:135–146.

Achraf Chalabi and Hany Gerges. 2012. Romanized Arabic transliteration. In *Proceedings of the Second Workshop on Advances in Text Input Methods*, pages 89–96, Mumbai, India, December. The COLING 2012 Organizing Committee.

Song Chen, Dana Fore, Stephanie Strassel, Haejoong Lee, and Jonathan Wright. 2017. BOLT Egyptian Arabic SMS/Chat and Transliteration LDC2017T07.

Song Chen, Dana Fore, Stephanie Strassel, Haejoong Lee, and Jonathan Wright. 2018. BOLT English SMS/Chat LDC2018T19.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, 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 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar.

Kareem Darwish. 2014. Arabizi Detection and Conversion to Arabic. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, pages 217–224, Doha, Qatar.

Nachum Dershowitz and Ori Terner. 2020. Transliteration of Judeo-Arabic texts into Arabic script using recurrent neural networks. *ArXiv*, abs/2004.11405.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Ahmed El Kholy and Nizar Habash. 2012. Orthographic and morphological processing for English–Arabic statistical machine translation. *Machine Translation*, 26(1-2):25–45.

Ramy Eskander, Nizar Habash, Owen Rambow, and Nadi Tomeh. 2013. Processing spontaneous orthography. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 585–595, Atlanta, Georgia.

Ramy Eskander, Mohamed Al-Badrashiny, Nizar Habash, and Owen Rambow. 2014. Foreign words and the automatic processing of Arabic social media text written in Roman script. In *Proceedings of the First Workshop on Computational Approaches to Code Switching*, pages 1–12, Doha, Qatar, oct. Association for Computational Linguistics.

Imane Guellil, cal Azouaou Fai Mourad Abbas, and Fatiha Sadat. 2017. Arabizi transliteration of Algerian Arabic dialect into Modern Standard Arabic. In *Social MT 2017/First workshop on social media and user generated content machine translation*.

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*, pages 15–22. Springer, Netherlands.

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

Nizar Habash, Fadhl Eryani, Salam Khalifa, Owen Rambow, Dana Abdulrahim, Alexander Erdmann, Reem Faraj, Wajdi Zaghouani, Houda Bouamor, Nasser Zalmout, Sara Hassan, Faisal Al shargi, Sakhar Alkhereyf, Basma Abdulkareem, Ramy Eskander, Mohammad Salameh, and Hind Saddiki. 2018. Unified guidelines and resources for Arabic dialect orthography. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Miyazaki, Japan.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Soumyadeep Kundu, Sayantan Paul, and Santanu Pal. 2018. A deep learning based approach to transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 79–83, Melbourne, Australia, jul. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal.

Abir Masmoudi, Mariem Ellouze Khmekhem, Mourad Khrouf, and Lamia Hadrich Belguith. 2019. Transliteration of Arabizi into Arabic script for Tunisian dialect. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19:1–21.

Hamdy Mubarak, Ahmed Abdelali, Kareem Darwish, Mohamed Eldesouki, Younes Samih, and Hassan Sajjad. 2019. A system for diacritizing four varieties of Arabic. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 217–222, Hong Kong, China, November. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, Pennsylvania, USA.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence neural network models for transliteration. *arXiv*, 1610.09565.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August. Association for Computational Linguistics.

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.

Taha Tobaili. 2016. Arabizi identification in Twitter data. In *Proceedings of the ACL 2016 Student Research Workshop*, pages 51–57, Berlin, Germany, August. Association for Computational Linguistics.

Clare Voss, Stephen Tratz, Jamal Laoudi, and Douglas Briesch. 2014. Finding romanized Arabic dialect in code-mixed tweets. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 2249–2253, Reykjavik, Iceland.

Daniel Watson, Nasser Zalmout, and Nizar Habash. 2018. Utilizing character and word embeddings for text normalization with sequence-to-sequence models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 837–843, Brussels, Belgium. Association for Computational Linguistics.

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, Łukasz 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.

Jihen Younes, Hadhemi Achour, and Emna Souissi. 2015. Constructing linguistic resources for the Tunisian dialect using textual user-generated contents on the social web. In *International Conference on Web Engineering*, pages 3–14. Springer.

Jihene Younes, Emna Souissi, Hadhemi Achour, and Ahmed Ferchichi. 2018. A sequence-to-sequence based approach for the double transliteration of Tunisian dialect. *Procedia Computer Science*, 142:238 – 245. Arabic Computational Linguistics.

Jihene Younes, Hadhemi Achour, Emna Souissi, and Ahmed Ferchichi. 2020. Romanized Tunisian dialect transliteration using sequence labelling techniques. *Journal of King Saud University-Computer and Information Sciences*.