Transliteration of Judeo-Arabic Texts into Arabic Script Using Recurrent Neural Networks

Ori Terner
School of Computer Science
Tel Aviv University
Ramat Aviv, Israel
`oriterner@gmail.com`

Kfir Bar
School of Computer Science
College of Management
Academic Studies
Rishon LeZion, Israel
`kfirb@colman.ac.il`

Nachum Dershowitz
School of Computer Science
Tel Aviv University
Ramat Aviv, Israel
`nachum@tau.ac.il`

Abstract

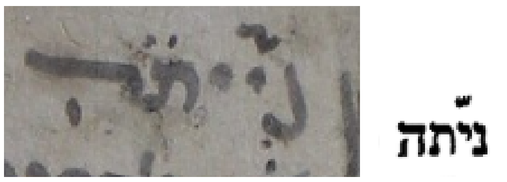
We trained a model to automatically transliterate Judeo-Arabic texts into Arabic script, enabling Arabic readers to access those writings. We employ a recurrent neural network (RNN), combined with the connectionist temporal classification (CTC) loss to deal with unequal input/output lengths. This obligates adjustments in the training data to avoid input sequences that are shorter than their corresponding outputs. We also utilize a pretraining stage with a different loss function to improve network converge. Since only a single source of parallel text was available for training, we take advantage of the possibility of generating data synthetically. We train a model that has the capability to memorize words in the output language, and that also utilizes context for distinguishing ambiguities in the transliteration. We obtain an improvement over the baseline 9.5% character error, achieving 2% error with our best configuration. To measure the contribution of context to learning, we also tested word-shuffled data, for which the error rises to 2.5%.

1 Introduction

Many great Jewish literary works of the Middle Ages were written in Judeo-Arabic, a Jewish dialect of the Arabic language family that adopts the Hebrew script as its writing system. Prominent authors include Maimonides (12th c.), Judah Halevi (11th–12th c.), and Saadia Gaon (10th c.). In this work, we develop an automatic transliteration system that converts Hebrew-letter Judeo-Arabic into readable Arabic text.

Generally speaking, given a text, transliteration is a process of converting the original graphemes to a sequence of graphemes in a target script. Specifically, transliterating a Judeo-Arabic text into the Arabic script almost invariably results in a text that has a similar number of letters. Yet, the correspondence between the letters in the transliteration is not one to one. Judeo-Arabic Hebrew script includes matres lectionis (e.g. ا، و، ى) to mark some of the vowels but it typically does not include nunation – *tanween* in Arabic (e.g. اً). Additionally, the *hamza* letter (ء), a relative latecomer to the Arabic writing system (Shaddel, 2018), is missing in the Judeo-Arabic script when it is placed "on the line" and not as a decoration for one of the matres lectionis.

Some other challenges are: (1) Authors of Judeo-Arabic texts sometimes use different mappings between Hebrew and Arabic letters. Some authors use the Hebrew letter ג to transliterate the Arabic letter ج, and others will use it to transliterate the Arabic letter غ. (2) Diacritic marks (small dots placed either above or below letters) are often omitted in the Hebrew script, another source of ambiguity. For example, the Arabic letters د and ذ are sometimes represented by the same letter in Hebrew. When the diacritic marks are maintained in the original Hebrew script, usually they are used in an inconsistent way. Even if they are used in the original manuscript, in many cases those marks appear differently or are completely missing in digital editions (such as those we used in this work). Figure 1 shows a few examples of this problem. The *apostrophe* is the only diacritic mark that is used in the digital texts we used. One important diacritic mark that is often missing from digital versions is *shadda* (gemination), which may be used


(a) Left: manuscript, right: printed edition, but digital-text reads: ניתא (missing *shadda* diacritic)

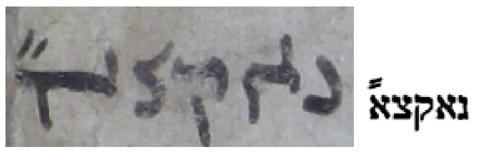

(b) Left: manuscript, right: printed edition, but digital-text reads: נאקצא (missing *tanwin* diacritic)

Figure 1: Information missing from the digital text and present in the manuscript and critical edition.

to easily resolve some lexical ambiguities. For example, the word درس ("he studied") has a different meaning when it appears with a *shadda* on the second consonant درّس ("he taught"). (3) There are several Arabic letters that do not have a one-to-one mapping with a Hebrew letter. For example, the Hebrew letter י typically refers either to ي or ى when appearing at the end of a word. (4) Judeo-Arabic is heavily affected by Hebrew and Aramaic; therefore, Judeo-Arabic texts are typically enriched with Hebrew and Aramaic citations, as well as with borrowed words and expressions from the two languages. Those citations and borrowings should not be blithely transliterated into Arabic, but rather need to be either left in the original script, semantically translated into Arabic, or otherwise annotated. Sometimes those borrowed words get inflected as if they were original Arabic words. For example, the word אלשכינה, composed of the Hebrew שכינה (*shkhina*, "divine spirit") and Arabic definite article אל (in Arabic ال); see (Bar et al., 2015).

The Friedberg Jewish Manuscript Society (`https://fjms.genizah.org`) has recently released a collection of hundreds of Judeo-Arabic works from different periods of time, formatted digitally as plain text and decorated with HTML tags. Hebrew and Aramaic citations and borrowings are annotated in those texts by domain experts. In the present work, we use texts from this collection. We noticed that some common borrowings were missed by the annotators; therefore, in this work we focus on the task of the transliteration of Judeo-Arabic words that originated in Arabic words only. Detecting the boundaries between Arabic-origin words and borrowings from other languages, a task that is known as "code switching", is left for future improvements. Since in our approach we consider the context of a word that needs to be transliterated, mostly to resolve lexical ambiguity, we masked those citation and borrowings, allowing the model to handle a continuous sequence of characters.

This Judeo-Arabic-to-Arabic transliteration problem has been addressed in one previous work (Bar et al., 2015). We elaborate on this and some other relevant previous works in the next section. In this work, we propose an end-to-end model to handle the transliteration task, by training a flavor of a recurrent neural network (RNN) on relatively short parallel texts, augmented with some synthetically generated data. Our model was designed with the capability of memorizing words in the output language. Section 3 describes the texts at our disposal. It is followed by a description of the baseline algorithm, and then our RNN solution.

2 Related Work

The only previous effort dealing with the transliteration of Judeo-Arabic texts (Bar et al., 2015) employed a method inspired by statistical machine translation, which was state of the art before deep neural nets took over. It consisted of a log linear model where the main component is a phrase table that counts the number of occurrences in the training data. As in (Rosca and Breuel, 2016), they also regarded transliteration as translation at the character level, that is, imagining single letters to be words. They improved their results by reranking their model's top predictions using a word-level language model. This is expected to be beneficial since using a character-level mechanism would incur the danger of generating non-words and nonsensical sequences of letters. A word-level language model that is rich enough to avoid high unknown rates would screen those results out.

Proper noun transliteration is a related task, which has been addressed previously in many works. The input term is usually provided without context, and the task is to rewrite the name in the target script, maintaining the way the name is pronounced. A direct phrase-based statistical approach to transliterating proper nouns from English to Arabic script is described in (AbdulJaleel and Larkey, 2002). It was preceded by largely handcrafted methods for romanization from Arabic to English such as (Arbabi et al., 1994; Stalls and Knight, 1998). A deep-belief approach for proper names was taken in (Deselaers et al., 2009). An attention-based method for transliteration from Arabic named entities to English and vice-versa is taken in (Hadj Ameur et al., 2017).

In Rosca and Breuel (2016), a transliteration system of names that employs a recurrent neural network (RNN), in two different ways, was described. In one way, they developed a sequence-to-sequence model that elegantly handles the different lengths of the input and output, with CTC alignment (see Section 5.1). A minor difference is their use of "epsilon insertion" to deal with input sequences that are shorter than the matching output, which CTC cannot handle. We instead use a similar solution of letter doubling discussed later. The second approach is a model inspired by recent work in the field of machine translation applying an encoder-decoder architecture with an attention mechanism. They report on improved results using RNN compared to previous methods, as is usually the case when employing deep learning techniques with problem previously solved by other means.

Another closely related problem is the transcription of *Arabizi*, which is a romanized transcription of Arabic that emerged naturally as a means for writing Arabic in chats, especially in the days of limited keyboards. Transcription into Arabic graphemes was studied by (Bies et al., 2014). Except for the different dialects and code switches from colloquial Arabic of different dialects to modern standard Arabic (MSA), they also had to handle changes in the language induced by the platform it appears in, including *emoticons* and deliberate manipulation of words, such as repetition of a single letter for emphasis common with internet social media. The same problem was addressed in (Al-Badrashiny et al., 2014), which maps Roman letters to Arabic scripts to produce a set of possibilities and chooses from them using a language model.

In another related task, the goal was to translate written text to the International Phonetic Alphabet (IPA), capturing the pronunciation of the written text. The work in (Rao et al., 2015) tried different RNN models, handling unequal sized input-output pairs by epsilon post-padding. They experimented with time delays, that is, postponing the output by a few timestamps (by pre-padding the output while post-padding the input accordingly). This allows the network to catch more of the input before deciding on the output. They also compared this with a bidirectional long-short-term-memory (LSTM) network that is able to see the entire context backward and forward. They combined the bidirectional LSTM with a CTC layer, handling the longer-output-than-input issue with epsilon post-padding (we used doubling). They reported, as expected, that greater contextual information contributes to performance. The bidirectional LSTM performs better than the unidirectional one even when the full context (whole word) is fed to the network before it starts its prediction. The best performance was obtained using the bidirectional LSTM, combined with an n-gram (non-RNN) model.

This transcription from a written language to IPA can potentially be used as a mediator to transcribe between any two languages were we to have an encoder from IPA to the source language and a decoder from IPA for the target script. Another potential use for this ability of transliteration to IPA would be for improved spell checking and correction (for those who only have basic knowledge of the sounds of the letters). A model could predict the utterances from the graphemes and recognize the word by similarity in the sound domain, to arrive at the correct spelling. For example: *neyber* $\rightarrow$ *neɪber* $\rightarrow$ *neighbor*.

3 Data

Kuzari. To train and evaluate our RNN model, we needed a considerable amount of parallel text in Judeo-Arabic along with its Arabic transliteration. Thankfully, the *Kuzari* (*Kitab al Khazari*), a medieval philosophical treatise composed by Judah Halevi in Andalusia around 1140, was recast in Arabic by Nabih Bashir (Halevi, 2012). We use this Arabic version and the original Judeo-Arabic as a parallel resource for training. The original Judeo-Arabic text of the Kuzari is taken from the Friedberg repository, and is derived from the critical edition edited by David H. Baneth. It was based on several manuscripts and comprises 47,348 word tokens (15,532 unique types), about 11% of which are Hebrew and Aramaic insertions. It is important to mention that Bashir's book provides translations for those insertions, and that there is no distinction in his edition between the *transliterated* and *translated* words. As mentioned before, we postpone handling the Hebrew and Aramaic citation and borrowings in our work, and we use the annotations provided by the Friedberg's experts in the Judeo-Arabic text, to mask out all words that are not originated in Arabic.

After cleaning the data from translation comments, section numbering and other elements irrelevant for transliteration, we needed to align the Arabic translation with the Judeo-Arabic source, on the word level. Aligning the texts is necessary because Bashir's translation does not perfectly match the Judeo-Arabic text, there being some word insertions, deletions and edits. To do that, we begin by breaking the two texts into words, and then iterating the two sequences of words. At each iteration we choose between skipping a word from the source (Judeo-Arabic) text, skipping a word from the target (Arabic) text, or adding both as an aligned source/target pair. The decision is the choice that minimizes the total edit distance of the alignment. To calculate edit distance between strings of different scripts, we applied the simple map to the Arabic source to get a basic transliteration into Hebrew letters. (The details of the algorithm are given in (Terner, 2020, Appendix).) After this alignment process we only keep pairs of words in the alignment that are close enough edit-distance wise. We set the threshold for edit-distance similarity heuristically, after normalizing the edit distance result by the maximum length of the two words that were compared, to the value of 0.5. This left us with a total of 47,083 word pairs, of which 20% were chosen randomly as test data, and the rest left for training. We are aware of the generalization problem of training and testing on data from the same source.

Words that do not align well according to the predefined threshold, such as Hebrew insertions, are removed and replaced by a fixed symbol (H) both in the source and target texts. We do the same for Hebrew insertions that are annotated as such by Friedberg's annotators. We do this to preserve the sentence structure since we suspect that this information is of value to the model for making better predictions.

We see punctuation as an important feature of the language and we want to keep this information for the model to train with. Where there are disagreements (e.g. comma vs. period) in the source and target languages, we favor the source language.

Additionaly, we removed from the Arabic text the *harakat* marks (short vowel marks): *fathah*, *kasrah*, *dammah* and *sukun*. They appear rarely in written Arabic (except for text intended for children or in religious texts) and are usually considered "noise" for language-related algorithms due to their scarcity (Habash, 2010). In the Judeo-Arabic text, the equivalent to these diacritics is the Hebrew *niqqud* signs. They appear only for Hebrew insertions; hence they are removed incidentally by the replacement of Hebrew insertions. Sometimes we even make use of those marks to identify Hebrew insertions missed by Friedberg's experts. Note that the *tanwīn* symbols when combined with *alif* were standardised to the modern style, so the diacritic symbol appears on top of the *alif* and not on the letter preceding it.

***Beliefs and Opinions*.** We identified an additional parallel text, namely Sa'adiah Gaon's *The Book of Beliefs and Opinions* (*Kitāb al-Amānāt wa l-I'tiqādāt*). It contains roughly the same quantity of data and was also transliterated by Bashir. We did not use it as training data; rather, we extracted 20% of it as additional test data, for evaluating how well the model performs on unseen text.

Synthetic data. Although we have a considerable amount of parallel data, since we use only one text source for training, we faced the problem of the model fitting to the specific writing style of the training data, but generalizing poorly to other texts. To improve the model and make it more robust, we generated synthetic data, using a simple technique, again leveraging the fact that the opposite direction of transliteration, that is, from Arabic to Hebrew, has almost no ambiguities (at least in the sources that we worked with in this study).

We found some additional texts online that correspond to the same era in which our texts were written. We used a simple mapping to produce a pseudo-transliteration from Arabic to Judeo-Arabic for them (Table 2 in Appendix). For instance, for the first two words in the text of *Ilāhiyyāt*, كتاب الشفاء, a pseudo-Judeo-Arabic transliteration is generated: כתאב אלשפא. The generated Judeo-Arabic text along with its Arabic counterpart are added to the parallel data for training the model.

سُئِلْتُ	סילת
إسرائيل	אסראיל

Table 1: Different transliterations of ـئ. In the first row it matches Hebrew י, while in the second row it matches א. The unusual transliteration might stem from the spelling of the translation.

This gave us parallel data that are genuine on the Arabic side but fictitious on the Judeo-Arabic side. Thus, it might include words written in a different way than an original Judeo-Arabic author might have written them. The use of such training data is justified partly by the fact that we are likewise only interested in the accuracy of our model's predictions on the target (Arabic) side. Therefore, we are less concerned about providing the model with noisy examples. This synthetic data significantly enlarged the quantity available for training.

4 Baseline Algorithm

The evaluation metric that we use is simply the average edit distance over all examples in the test dataset. The edit distance (ED) that we use is the Levenshtein distance, which is calculated between the predicted characters and the ground truth. It is then normalized by the length of the ground truth. The formula for *label error rate (LER)* is $\frac{1}{|S|}\sum_{(x,z)\in S} ED(h(x),z)/|z|$ for model h on test data $S \subseteq X \times Z$, where X are the inputs, z is ground truth and $|z|$ is the length of z. This is a natural measure for a model where the aim is to produce a correct label sequence (Graves et al., 2006).

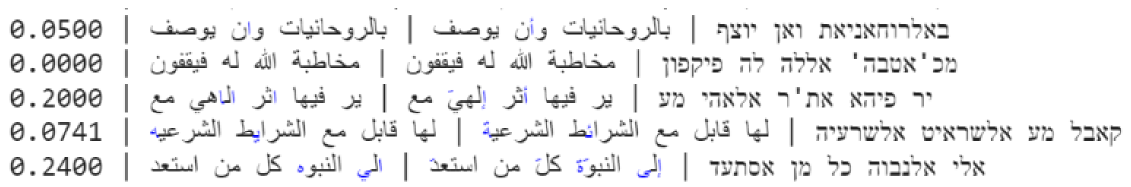
To evaluate results, we start by creating a baseline transliteration on the test data that translate each Hebrew letter to the most common letter according to the predefined mapping mentioned earlier (Table 2 in Appendix). We produced the list manually according to a modern convention for transliterating Arabic into Hebrew. This simple mapping achieves a relatively high accuracy (LER 9.51%) and demonstrates the nature of this Judeo-Arabic transliteration problem. Though it is easy to achieve high accuracy, to produce readable text and to be able to confront ambiguities in the text, some language ability is desirable. The baseline results still do not guarantee fluent reading of the generated target text.

Common baseline mistakes. The baseline errors arise mainly from ambiguous letters that have more than a single mapping. In what follows, we enumerate the most prominent ambiguities.

Transliteration of Hebrew *alef*. The letter *alef* (א) most commonly should be transliterated as the Arabic letter *alif* (ا). This Arabic grapheme usually indicates an elongated /a/ vowel attached to the preceding consonant. However, it can also sometimes indicate a glottal stop, that is an *alif with hamza on top* (أ). As (Habash, 2010) mentions, Arabic writers often ignore writing the *hamza* (especially with stem-initial *alifs*) and it is "de-facto optional". This will also lead to false negatives for the test data, deciding that the transliteration is an error while in fact it would be accepted by a human reader, unjustifiably increasing LER; see (Rosca and Breuel, 2016, Section 2.3). A more complex model could hopefully predict the places were *hamza* is required for disambiguation (for instance words with stem-*non*-initial *alifs*). Alternatively, such a model would hopefully have a rich enough memory of the Arabic words it has seen, attaching the *hamza* sign for words it has seen in the training data. If indeed there are two legitimate forms, this model will also know to disambiguate according to the context, as we train on sequences, that is, words in context.

Rarer cases for transliteration of א are as follows: *hamza* on the line (ء) even though it is usually not transcribed in Hebrew. Also it can mean an *alif maqsura* (ى) at the end of a word, but *alif maqsura* is usually mapped to the letter *yod* (י). For instance, the 3-letter word וגא is transliterated as the 4-letter وجاء. Here it is not clear whether the letter א corresponds to the *hamza* or to the long vowel *alif* that precedes it, in which case the *hamza* is missing from the transliteration. In the baseline algorithm, we map א to the most common transliteration, that is, a non-*hamza alif*. Thus we miss all the other variations.

Transliteration of Hebrew *yod*. The two most common uses of the *yod* (י) are for the Arabic letter



```
0.0500 | بالروحانيات وان يوصف | بالروحانيات وأن يوصف | באלרוחאניאת ואן יוצף
0.0000 | مخاطبة الله له فيقفون | مخاطبة الله له فيقفون | מכ'אטבה' אללה לה פיקפון
0.2000 | ير فيها اثر الاهي مع | ير فيها أثر إلهيّ مع | יר פיהא את'ר אלאהי מע
0.0741 | لها قابل مع الشرايط الشرعيه | لها قابل مع الشرائط الشرعية | קאבל מע אלשראיט אלשרעיה
0.2400 | الي النبوه كل من استعد | إلى النبوّة كلّ من استعدّ | אלי אלנבוה כל מן אסתעד
```

Figure 2: Baseline results. Columns, right to left: Judeo-Arabic text, ground truth, baseline prediction, error rate. Errors are marked in blue. Observe, for instance, the 4th word in row 3, missing a *shadda* in the prediction and with an extra א in the source sentence.

ya (ي) and for *alif maqsura* (ى), a dotless *ya* appearing always at the end of a word. Less frequently it can also be a transliteration of *ya hamza* (ئ). But there are variations. For instance, in the first of the examples in Table 1, the *ya hamza* is transliterated as *yod* (י), while in the second when followed by a regular *ya*, it can be seen as either transliterated to a Hebrew *alef* (א) or dropped. In this example, the variation can be due to the spelling in Hebrew of the translation that uses א: "ישראל" (Israel in Hebrew).

In Egypt, but not necessarily in other Arabic countries, a final *ya* is often written dotless, that is, as an *alif maqsura* (Habash, 2010). This seems to be the case in the transliteration of Maimonides' book, *The Guide for the Perplexed* (*Dalālat al-ḥā'irīn*; `http://sepehr.mohamadi.name/download/DelalatolHaerin.pdf`), as transliterated by Hussein Attai (e.g. p. 45). Unfortunately, the book is not available as a digital text. (A page of the *Guide*, with human and machine transliterations, is shown in Figure 6 in Appendix.) In the baseline algorithm, we map *yod* invariably to a regular *ya*.

Arabic *shadda* (gemination). Another critical issue is the *shadda* diacritic (e.g. بّ). The shadda is not present in the Judeo-Arabic text, or was omitted in the digitized version of the text as described above. Unfortunately, we could not figure out a simple rule of thumb to handle the presence or absence of shadda that could be adopted for the baseline algorithm.

Baseline results. As mentioned above, the baseline algorithm fully disregards *shadda*, *hamza*, *alif maqsura*, *tanwin* and a few other marks. Nonetheless the error on the test data does not increase (LER 9.5%). Some results of the baseline algorithm are shown in Figure 2.

5 Method and Results

5.1 Training using CTC Loss with Letter Doubling

The connectionist temporal classification (CTC) loss is a method introduced in (Graves et al., 2006) that enables a tagging recurrent neural network (RNN) to learn to predict discrete labels from a continuous signal without requiring the training data to be aligned input to output. Instead, the model produces a distribution over all alignments of all possible labels while facilitating an extra character (the *blank* symbol) added to its softmax layer in order to produce the alignment. Thus, the probability of any label conditioned on the input can be calculated as the sum over all possible alignments of the given label. CTC is appropriate for our mission because the Judeo-Arabic inputs and Arabic outputs are not always of equal length. Also, there is no available alignment of the two texts at the character level.

Applying CTC loss is a convenient solution for handling this problem. However, the CTC loss is only defined when the input sequence is longer than the output sequence, which is not always the case for us. Actually, since there are diacritic signs in the Arabic transliteration included in the character count that do not appear in the Judeo-Arabic source—this is a prevalent situation. This will require an adaptation of the dataset below.

It should be noted that, as (Chan et al., 2016) describes, the CTC mechanism implicitly assumes independence of characters over time. By using the multiplicative rule between probabilities over time, it disregards the dependencies between timestamps. In spite of this strong assumption, CTC achieves a substantial improvement in performance for various sequential tasks, such as speech recognition (Graves et al., 2006), OCR (Shi et al., 2017) and handwriting recognition (Graves et al., 2007).

Doubling. This technique of dealing with shorter input than output sequences that we use is similar to that used by (Rosca and Breuel, 2016). But instead of using a special character (epsilon) inserted between each timestamps a constant number of times, known as *epsilon insertions*, we filled the extra spaces by repetition of the previous timestamp label. For instance, the sequence עלי מא שהד וג'א פי would become עעלליי ממאא ששההדד ווגג''אא פפיי. Note the doubling of the apostrophe sign, which is performed separately from the letter ג that it decorates.

Aesthetically speaking, this brings the usage of CTC closer to its original usage of transcribing a continuous signal to discrete labeling (as in speech recognition) in the sense that the input text obtains the appearance of a continuous signal, whereas epsilon insertions break the succession. Further work is required to examine the impact on performance of these two alternatives.

5.2 Training the RNN

We use for our model GRU (Cho et al., 2014) cells, stacked in four layers, followed by a linear layer activated by softmax for the CTC loss. Each layer is bidirectional (meaning the cells observed the input both backwards and forwards), and contains 1,024 units. We use letter embedding for the input of dimension 8. The model is implemented with TensorFlow, and optimized with RMSprop. The text is divided into short 20 characters sequences (according to the lengths in the input side). The sequences contain complete words. If the 20th character happens to be midword, the rest of the word was included in the sequence. The batch size is 128.

5.3 Pretraining on Single Letters

A method that was beneficial for speeding up convergence of the network was to pretrain the network with single letters according to the simple letter mapping between the Hebrew and Arabic alphabet. This is as if "to set the model in the right direction". This is intuitively reasonable if you think of the way children learn how to read. First they learn to identify the individual letters and then to assemble them into words. We use for this training step the *sparse cross entropy loss* trained on generated random parallel character sequences of length 10. As we show, this makes it feasible to train deeper networks. It might also be helpful, for instance in the task of speech recognition, to pretrain the network first on single time samples from a certain phoneme to teach the network to map to the correct grapheme, before training on continuous speech with CTC loss, which is more complex, and with which it is less obvious for the network how to start to optimize than with the simpler cross entropy loss.

Intermediate results. With pretraining with the cross entropy loss, the network converges quite fast to reach error rate of 2% on the test data. On the other hand, without pretraining, it gets stuck at a local minimum, transliterating each character in the input to the most prevalent character in the target data, which is a space character, producing as output strings of repeating spaces. Figure 3 shows the losses and accuracy measures. In the remainder of the experiments, we include this pretrain stage.

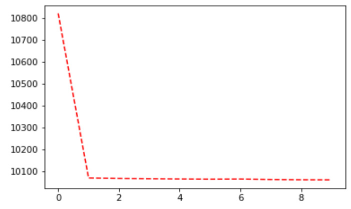
Running this model against the additional text of *Emunoth ve-Deoth* yields a higher error rate of 3.24%. In other words, we see a decrease in performance for unseen texts. We continue with an exploration of ways for improving prediction for unseen texts.

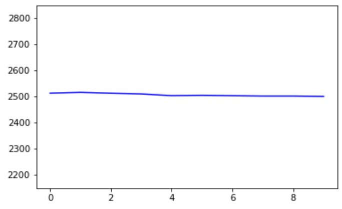
5.4 Training with Synthetic Data

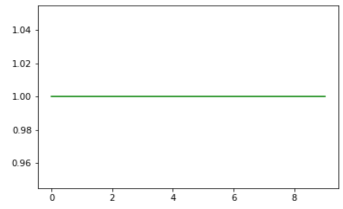
As described in Section 3, we propose a technique to augment the training data by generating pseudo-parallel texts using Arabic writings of roughly the same era in which our Judeo-Arabic texts were written.

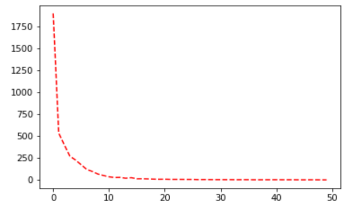
The texts we used for this purpose in the current experiments are: (1) Avicenna's *Ilāhiyyāt*, (2) Al-Farabi's *Kitab Rilasa al-Huruf*, (3) Al-Farabi's *Kitab Tahsil al-Saida* and (4) Averroes's *Al-Darurī fī Isul al-Fiqh*. To all that we add the original "real" dataset of the *Kuzari*. By doing this, we significantly enlarge the amount of data that we have for training.

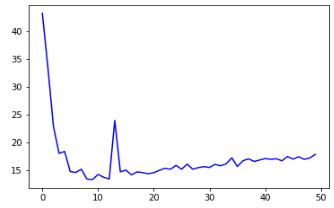
Results. With the synthetic data, the accuracy on the original text data *decreased*, as expected, to 2.48%. On the additional data the accuracy also *decreased* but to a lesser extent, to 3.37%. This drop in performance might indicate that the Arabic texts that we chose for generating the synthetic data are not perfectly suitable for Judeo-Arabic. Perhaps there are other sources to consider that are more suitable.


(a) Training loss: without pretrain


(b) Test loss: without pretrain


(c) Test accuracy: without pretrain


(d) Training loss: with pretrain


(e) Test loss: with pretrain

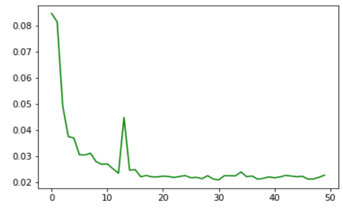
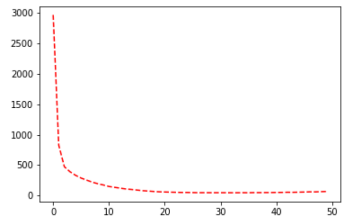
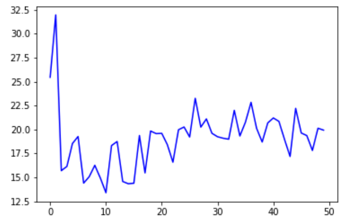

(f) Test accuracy: with pretrain

Figure 3: Comparing results with pretraining with single graphemes with CE loss (bottom) and without (top). Beware of the differences in scales.


(a) Training loss


(b) Test loss

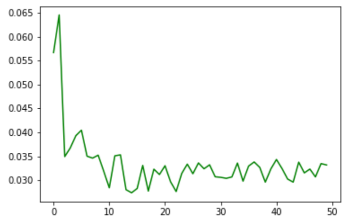

(c) Test accuracy

Figure 4: Results of training with added synthetic data.

Note that the degree of fluctuation in the learning curve on the test is greater. This might be because the synthetic data is shuffled between epochs. The results are presented in Figure 4.

5.5 Dropout

We are mainly interested in making the network to remember what would make sensible Arabic outputs, relying on the letter mapping, but allowing some flexibility. It should be able to output Arabic words that were seen during training even though the input sequence is not the exact one seen during training. Experimenting with a small model showed the plausibility of this direction of thinking. An immediate implication was to use *dropout* on the input sequence to make the network more robust to noise in the Judeo-Arabic input. We achieved this by randomly setting a percentile of the nonspace symbols in the input sequence (before doubling the letters) to the *blank* symbol.

Results. With dropout rate of 15% of the nonspace symbols that was performed on the synthetic data, the test results for the original test of the *Kuzari* is set at 2.26%, which is worse than the accuracy without synthetic data and without dropout but better than the results when trained on the synthetic data without the dropout. On the other hand, for the additional text of *Emunot*, a marked improvement is achieved, yielding an error rate of 3.14% (a 0.1 percent improvement).

(a) 0.0000 | في قصور الملوك , بكتاب | في قصور الملوك , بكتاب | פי קצור אלמלוכ , בכתאב

(b) 0.0870 | H . وفي صدور أهل الريئا | H . وفي صدور أهل الرياء | H . ופי צדור אהל אלריא

(c) 0.0000 | على فعل ما شاء متى شاء | على فعل ما شاء متى شاء | עלי פעל מא שא מתי שא

(d) 0.1364 | بعضهم لولده موصّى عند | بعضهم لولده موصياً عند | בעצ'הם לולדה מוצי ענד

(e) 0.0400 | الله تابوتاً , وأقام عليها | الله تابوتاً , وأقام عليه | אללה תאבותא , ואקאמ עליהא

(f) 0.1667 | وبإيمات من تعجب وسؤال | وبايماءات من تعجّب وسؤال | ובאימאאת מנ תעג'ב וסואל

(g) 0.0000 | H بأفراد كانوا لباباً | H بأفراد كانوا لباباً | H באפראד כאנוא לבאבא

(h) 0.0000 | تشكّلت السماوات والأرض | تشكّلت السماوات والأرض | תשכלת אלסמאואת ואלארצ'

(i) 0.0800 | : إنّا خالق العالم وخالقكم | : أنا خالق العالم وخالقكم | : אנא כ'אלק אלעאלמ וכ'אלקכמ

(j) 0.0000 | المتكرّر عليه , بأن يطلب | المتكرّر عليه , بأن يطلب | אלמתכרר עליה , באנ יטלב

Figure 5: Example transliteration results for proposed model. Right to left: Judeo-Arabic, ground truth, baseline prediction, error rate. In (b), for the last word the network misses the ground truth. Notice the correct positioning of the *Alif Hamza*. In (d) the mistake is partly due to noise in the data since the ground truth form موصياً is a conjugation of the noun موصي (meaning "recommender"), which the Judeo-Arabic מוצי form resembles more. The form that the network finally produced also exists. Example (i) shows a mistake in distinguishing between word senses since both diacritizations of انا produce legitimate words.

6 Discussion

We designed a model for automatic transliteration of Judeo-Arabic texts into Arabic scripts. Endeavoring to overcome the problem of ambiguous mappings in the transliteration, we trained an RNN model using the CTC loss that has enabled us to cope with unequal length input/output sequences due to the addition of diacritics in the target (Arabic) side of the data. As mentioned, we wanted to create a network that will have memory, along with some language capability in the output side that will enable it to distinguish between different word senses and overcome small variations in the transliteration.

The results demonstrate a substantial decrease in error compared to the baseline, from 9.5% to 2% LER, showing that the network is capable of attaching correct diacritics to the Arabic. On unseen text, the network incurs a 3.24% error rate. This is important since Judeo-Arabic texts vary according to the writer, time period, and region. Examples and some remaining problems are included in Figure 5.

We experimented with several methods to enhance training. First, we exercised pre-training of the network with cross-entropy loss to teach the network the simple mapping used in the baseline transliteration. This can enlarge and deepen the network and still guarantee convergence. Without this pretraining stage, the network failed to discover this mapping by itself. Since we only train on parallel text from a single source, and we are interested in making the model generalize better to unseen texts, we augment the training dataset by generating parallel texts out of available medieval Arabic texts. This had a slight adverse affect on accuracy and might depend on the choice of Arab texts that were used for the synthetic data. Dropout was considered and implemented on the synthetic data with the desired affect. While accuracy on the first test data, taken from the same source as the original training data decreased as expected, accuracy for the additional test data improved. We suspect that the network assimilates more language skills due to the dropout on the training data.

Examining the top 5 results of the CTC-decode beam search, instead of only the top one, reveals that sometimes the correct transliteration, at least the one that was chosen by the human transliterator, is among those five. Therefore, running a pure Arabic language model should enhance model accuracy.

One question that arises is how much of the language the network catches. To examine this, we perform a forward pass on the shuffled test data. Shuffling is done at word level. This generates a parallel dataset of words that lack context, sharing the same word distribution as the real test data. Indeed as expected, on the shuffled test data the error increases by $\sim$0.5% on average.

References

Nasreen AbdulJaleel and Leah Larkey. 2002. English to Arabic transliteration for information retrieval: A statistical approach. Technical Report IR-261, Center for Intelligent Information Retrieval Computer Science, University of Massachusetts.

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

Mansur Arbabi, Scott M. Fischthal, Vincent C. Cheng, and Elizabeth Bar. 1994. Algorithms for Arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–193.

Kfir Bar, Nachum Dershowitz, Lior Wolf, Yackov Lubarsky, and Yaacov Choueka. 2015. Processing Judeo-Arabic texts. In *Proceedings of the First International Conference on Arabic Computational Linguistics (ACLing '15, Cairo)*, pages 138–144, April.

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 EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 93–103.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964.

Kyunghyun Cho, Bart van Merriënboer, 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, October. Association for Computational Linguistics.

Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. 2009. A deep learning approach to machine transliteration. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, StatMT '09, pages 233–241. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 369–376. ACM.

Alex Graves, Santiago Fernández, Marcus Liwicki, Horst Bunke, and Jürgen Schmidhuber. 2007. Unconstrained on-line handwriting recognition with recurrent neural networks. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, volume 20, pages 577–584, January.

Nizar Y. Habash. 2010. *Arabic Natural Language Processing*. Synthesis Digital Library of Engineering and Computer Science. Morgan & Claypool Publishers.

Mohamed Seghir Hadj Ameur, Farid Meziane, and Ahmed Guessoum. 2017. Arabic machine transliteration using an attention-based encoder-decoder model. In *Third International Conference On Arabic Computational Linguistics (ACLING 2017, November 2017, Dubai, United Arab Emirates)*, volume 117 of *Procedia Computer Science*, pages 287–297, November.

Yehuda Halevi. 2012. *The Kuzari – In Defense of the Despised Faith*. Al-Kamel Verlag, Beirut. Transliterated, translated into Arabic and annotated by Nabih Bashir.

Kanishka Rao, Fuchun Peng, Haşim Sak, and Françoise Beaufays. 2015. Grapheme-to-phoneme conversion using long short-term memory recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4225–4229.

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

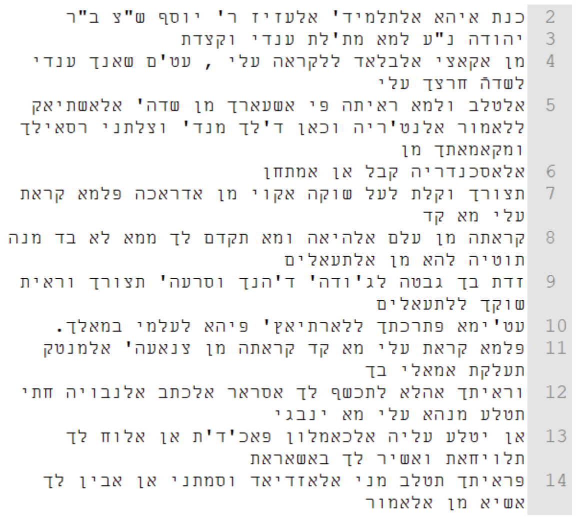
Mehdy Shaddel. 2018. Traces of the hamza in the early Arabic script: The inscriptions of Zuhayr, Qays the Scribe, and "Yazd the King". *Arabian Epigraphic Notes*, 4:35–52.

Baoguang Shi, Xiang Bai, and Cong Yao. 2017. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(11):2298–2304, November.

Bonnie Glover Stalls and Kevin Knight. 1998. Translating names and technical terms in Arabic text. In *Proceedings of the COLING/ACL Workshop on Computational Approaches to Semitic Languages (Semitic '98)*, pages 34–41, August.

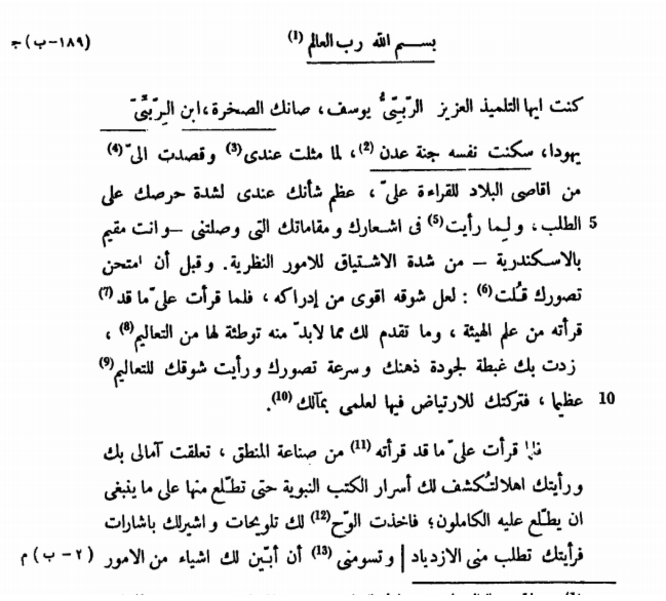
Ori Terner. 2020. Transliteration of Judeo-Arabic texts to Arabic using RNN. M.Sc. thesis, School of Computer Science, Tel Aviv University.

Appendix: Figures and Tables



כנת איהא אלתלמיד' אלעזיז ר' יוסף ש"צ ב"ר
יהודה נ"ע למא מת'לת ענדי וקצדת
מן אקאצי אלבלאד ללקראה עלי , עט'ם שאנך ענדי לשדה חרצך עלי
אלטלב ולמא ראיתה פי אשעארך מן שדה' אלאשתיאק ללאמור אלנט'ריה וכאן ד'לך מנד' וצלתני רסאילך ומקאמאתך מן
אלאסכנדריה קבל אן אמתחן
תצורך וקלת לעל שוקה אקוי מן אדראכה פלמא קראת עלי מא קד
קראתה מן עלם אלהיאה ומא תקדם לך ממא לא בד מנה תוטיה להא מן אלתעאלים
זדת בך גבטה לג'ודה' ד'הנך וסרעה' תצורך וראית שוקך ללתעאלים
עט'ימא פתרכתך ללארתיאץ' פיהא לעלמי במאלך.
פלמא קראת עלי מא קד קראתה מן צנאעה' אלמנטק תעלקת אמאלי בך
וראיתך אהלא לתכשף לך אסראר אלכתב אלנבויה חתי תטלע מנהא עלי מא ינבגי
אן יטלע עליה אלכאמלון פאכ'ד'ת אן אלוח לך תלויחאת ואשיר לך באשאראת
פראיתך תטלב מני אלאזדיאד וסמתני אן אבין לך אשיא מן אלאמור

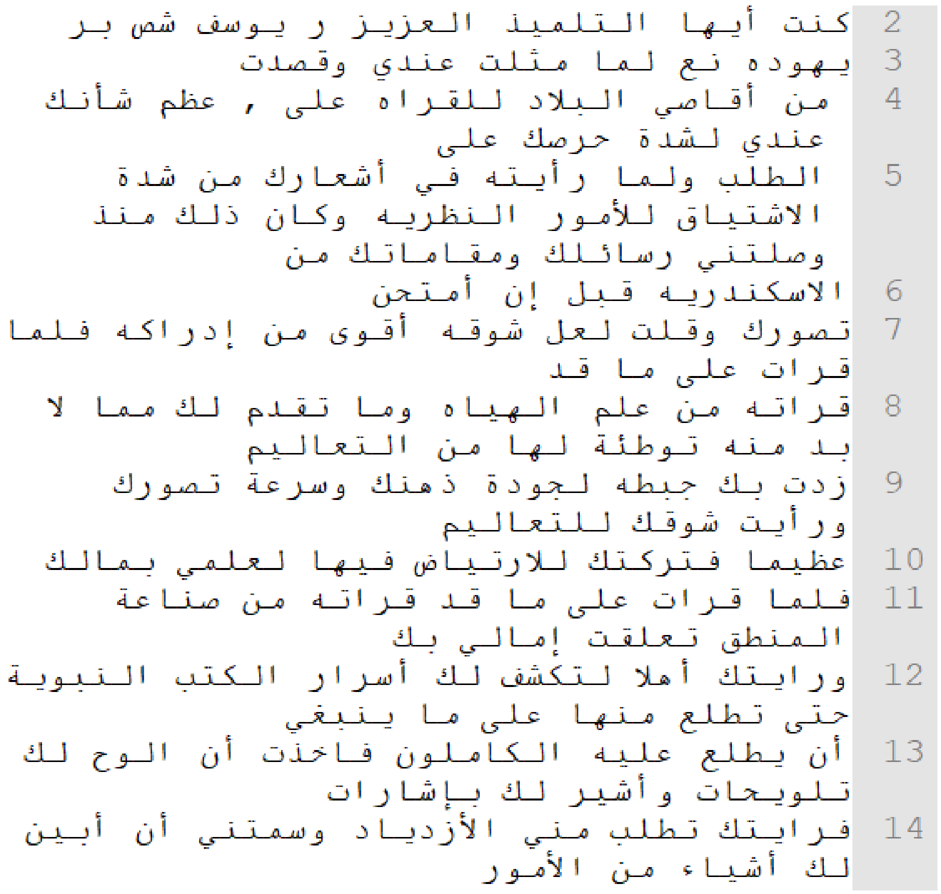
(a) Original Judeo-Arabic orthography.



بســـم الله رب العالم (1) (١٨٩-ب) ج

كنت ايها التلميذ العزيز الرّبّيّ يوسف، صانك الصخرة،ابن الرّبّيّ
يهودا، سكنت نفسه جنة عدن (2)، لما مثلت عندى(3) وقصدت الىّ(4)
من اقاصى البلاد للقراءة علىّ ، عظم شأنك عندى لشدة حرصك على
الطلب ، ولِما رأيت(5) فى اشعارك و مقاماتك التى وصلتنى —وانت مقيم
بالاسكندرية — من شدة الاشتياق للامور النظرية. وقبل أن امتحن
تصورك قُلت(6) : لعل شوقه اقوى من إدراكه ، فلما قرأت علىّ ما قد(7)
قرأته من علم الهيئة ، وما تقدم لك مما لابدّ منه توطئة لها من التعاليم(8) ،
زدت بك غبطة لجودة ذهنك وسرعة تصورك ورأيت شوقك للتعاليم(9)
عظيما ، فتركتك للارتياض فيها لعلمى بمآلك(10).
فلما قرأت علىّ ما قد قرأته(11) من صناعة المنطق ، تعلقت آمالى بك
ورأيتك اهلالتُكشف لك أسرار الكتب النبوية حتى تطّلع منها على ما ينبغى
ان يطّلع عليه الكاملون؛ فاخذت الوّح(12) لك تلويحات و اشيرلك باشارات
فرأيتك تطلب منى الازدياد | وتسومنى(13) أن أبيّن لك اشياء من الامور (٢-ب) م

(b) Transliteration by Hussein Attai.



كنت أيها التلميذ العزيز ر يوسف شص بر
يهوده نع لما مثلت عندي وقصدت
من أقاصي البلاد للقراه على , عظم شأنك عندي لشدة حرصك على
الطلب ولما رأيته في أشعارك من شدة الاشتياق للأمور النظريه وكان ذلك منذ وصلتني رسائلك ومقاماتك من
الاسكندريه قبل إن أمتحن
تصورك وقلت لعل شوقه أقوى من إدراكه فلما قرات على ما قد
قراته من علم الهياه وما تقدم لك مما لا بد منه توطئة لها من التعاليم
زدت بك جبطه لجودة ذهنك وسرعة تصورك ورأيت شوقك للتعاليم
عظيما فتركتك للارتياض فيها لعلمي بمالك
فلما قرات على ما قد قراته من صناعة المنطق تعلقت إمالي بك
ورايتك أهلا لتكشف لك أسرار الكتب النبوية حتى تطلع منها على ما ينبغي
أن يطلع عليه الكاملون فاخذت أن الوح لك تلويحات و أشير لك بإشارات
فرايتك تطلب مني الأزدياد وسمتني أن أبين لك أشياء من الأمور

(c) Our model transliteration.

Figure 6: First page of Maimonides' *The Guide for the Perplexed*.

Judeo-Arabic	Arabic	Judeo-Arabic	Arabic
א	ا	כ	ك
ב	ب	כ'	خ
ג'	ج	ל	ل
ג	غ	מ	م
ד	د	נ	ن
ד	ذ	ס	س
ה	ه	ע	ع
ה'	ة	פ	ف
ו	و	צ	ص
ז	ز	צ'	ض
ח	ح	ק	ق
ט	ط	ר	ر
ט'	ظ	ש	ش
י	ي	ת	ت
		ת'	ث

Table 2: Simple mapping rules for baseline transliteration.