Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 117 (2017) 287–297

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Arabic Computational Linguistics, ACLing 2017, 5–6 November 2017, Dubai, United Arab Emirates

Arabic Machine Transliteration using an Attention-based Encoder-decoder Model

Mohamed Seghir Hadj Ameur[a,*], Farid Meziane[b], Ahmed Guessoum[a]

[a] *TALAA Group, USTHB University, Bab-Ezzouar, Algiers, Algeria*
mhadjameur@usthb.dz, aguessoum@usthb.dz
[b] *Informatics Research Centre, University of Salford, M5 4WT, United Kingdom*
f.meziane@salford.ac.uk

Abstract

Transliteration is the process of converting words from a given source language alphabet to a target language alphabet, in a way that best preserves the phonetic and orthographic aspects of the transliterated words. Even though an important effort has been made towards improving this process for many languages such as English, French and Chinese, little research work has been accomplished with regard to the Arabic language. In this work, an attention-based encoder-decoder system is proposed for the task of Machine Transliteration between the Arabic and English languages. Our experiments proved the efficiency of our proposal approach in comparison to some previous research developed in this area.

Peer-review under responsibility of the scientific committee of the 3rd International Conference on Arabic Computational Linguistics.

Keywords: Natural Language Processing; Arabic Language; Arabic Transliteration; Deep Learning; Sequence-to-sequence Models; Encoder-decoder Architecture; Recurrent Neural Networks

1. Introduction

Machine Transliteration [1, 2] is the process of transforming a given word from one alphabet to another while preserving the phonetic and orthographic aspects of the transliterated word. Even though the task of transliteration may appear to be trivial at first glance, it turns out to be a complicated one. The main reason for its difficulty is the absence of some phonetic character correspondences between the source and target languages. In such situations, this kinds of characters will need to be either omitted or even approximated depending on their context of occurrence. For instance, in the task of transliteration from Arabic to English, some Arabic letters such as "ط", "ظ" and "ث" do not have any direct single-letter correspondences in the English alphabet; thus the system will need to transliterate them in a way that best preserves their phonetic aspects.

The accurate transliteration of named entities can be very crucial for many applications. For instance, it can be used to handle Out-Of-Vocabulary (OOV) words in Machine Translation (MT)systems [3, 4], and incorporated to handle proper names transliteration in Cross-lingual Information Retrieval (IR) [5, 6]. With the emergence of deep learning,

* Corresponding author. Tel.: +213-559-264-634
E-mail address: mhadjameur@usthb.dz

1877-0509
Peer-review under responsibility of the scientific committee of the 3rd International Conference on Arabic Computational Linguistics.
10.1016/j.procs.2017.10.120

sequence-to-sequence [7, 8] models have seen a significant improvement. Given the importance of the latter in many applications, several attempts have been made toward improving them using these deep-learning models [1, 2]; yet only little research has been conducted in this direction for the Arabic language.

In this work, we first present a methodology for creating a large supervised training data from raw English-Arabic parallel corpora. Then we present our attention-based encoder-decoder transliteration model. The proposed system is evaluated on the task of transliteration from both English-to-Arabic and Arabic-to-English. The results obtained show a noticeable improvement in the transliteration accuracy over some previous works on these language pairs, which proves the adequacy of our proposal.

The remainder of this paper is organized as follows. Section 2 presents relevant related work in this area and motivates our contribution. The followed methodology to construct the transliteration corpus is then presented in Section 3. The details of our proposed transliteration system is then described in section 4. In Section 5, we present and discuss the tests performed and the obtained results. In Section 6, we conclude our work and highlight some possible future improvements and research directions.

2. Related Work

A considerable amount of work was developed in the area of machine transliteration. Shao et al. [9] used a Convolutional Neural Network (CNN) for the task of transliteration between English and Chinese. They compared their system with a phrase-based Statistical Machine Translation (SMT) system and found that the accuracy they obtained was slightly below that of the SMT system. They justified these results by the higher order language model incorporated in the SMT framework. Finch et al. [10] proposed a method that exploits the agreement between a pair of target-bidirectional Recurrent Neural Networks (RNNs). Their experimental results carried out on various language pairs showed that their proposal performs similar, or even better than the Phrase-based Statistical Machine Translation system (PSMT) on many language pairs. Jadidinejad [2] proposed a sequence-to-sequence model consisting of a bidirectional encoder and an attention-based decoder. They used their proposal for the task of transliteration between several language pairs. Their experimental results showed that their system outperformed the classical PSMT system. Jiang et al. [11] proposed a Maximum Entropy model (MaxEnt) for named entity transliteration from English to Chinese. Their model ranks the transliteration candidates by combining pronunciation similarities and bilingual co-occurrences. They compared their system with some rule-based approaches and reported a slight improvement in the overall accuracy.

In the context of the Arabic language, Arbabi et al. [12] presented a hybrid algorithm for English-to-Arabic transliteration that uses both Feed-forward Neural Networks (FFNs) and Knowledge-based Systems (KBSs). Deselaers et al. [1] used Deep Belief Networks (DBNs) which consist of multiple Restricted Boltzmann Machine (RBM) layers. They used their system for the Arabic-to-English city names transcription task. The results they obtained have shown that the PSMT system clearly outperforms their proposal. Despite that, the authors stated that their system can easily be combined with other state-of-the-art systems to achieve better results. AbdulJaleel and Larkey [13] presented a Phrase-based Statistical system for English-to-Arabic transliteration using unigram and bigram character-based translation models. Their results showed a higher accuracy when the bigram model was incorporated. Rosca and Breuel [14] used a neural sequence-to-sequence model for the task of machine transliteration between several languages including Arabic-to-English, in which they achieved 77.1% Word Error Rate (WER) accuracy.

To the best of our knowledge there is no prior research work that has been made for the task of machine transliteration from English-to-Arabic using deep learning methods. Thus we believe that our work will be the first attempt in this direction.

3. Building a Transliteration Corpus

As pointed out by Rosca and Breuel [14], there is a lack in Arabic machine transliteration corpora (that are freely available). In this section, we present our proposed methodology that has allowed us to automatically create a super-

vised named entity transliteration corpus from a raw parallel textual data. We note that our constructed corpus will be made freely available for the NLP research community [1].

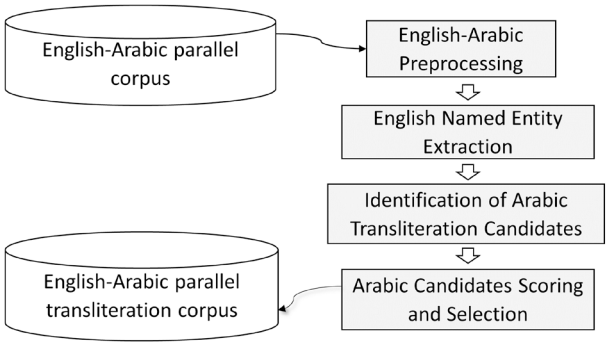

Fig. 1: The architecture of our parallel English-Arabic Named entity extraction system.

Our extraction system (Fig. 1) uses a parallel corpus in order to automatically extract bilingual named entities. The system starts by a preprocessing step in which all the sentences in both the English and Arabic languages are normalized and tokenized. Then, the English named entities will be extracted from each sentence in the English-side of the parallel corpus. A set of Arabic transliteration candidates will then be assigned to each extracted English named entity. Finally, the best Arabic transliteration will be selected for each English named entity to form our final transliteration corpus. The detail of each step will be provided in the remainder of this section.

3.1. Parallel Named entity Extraction

We recall that our goal is to obtain the Arabic transliteration for each English named entity found in the parallel corpus. To this end, we first need to extract the named entities contained in each English sentence in the parallel corpus. Formally, given a set of English-Arabic parallel sentences $S = \{(e_1, a_1), ..., (e_n, a_n)\}$ we perform an English Named Entity recognition to find all the named entities present in the English-side of the corpus $N = \{n_1, n_2, ..., n_k\}$ where k is the total number of named entities. Just like [14], we transform all the named entities containing multiple words to several singleton entities, each one as a standalone named entity. We do this to avoid keeping phrases (having multiple words) in the training corpus, since each single English word can always be transliterated without needing any additional information about its preceding words (history). For each English named entity n_i belonging to a sentence e_j, we keep track of its corresponding Arabic sentence a_j. We end up with a list N of pairs (n_i, a_j) denoting that the i^{th} English named entity (singleton word) is associated with the j^{th} Arabic sentence.

3.2. Candidates Extraction and Scoring

From the previous step, we end up with a set of pairs (n_i, a_j), where n_i is the English named entity (word) and a_j is the Arabic sentence containing its transliteration. To identify the correct transliterated word in the Arabic sentence a_j, we first remove all the frequent Arabic words from it using a vocabulary containing the n most frequent Arabic words. This ensures that most of the remaining words in the Arabic sentence a_j are rare words (hopefully only named entities). All the remaining words in a_j are considered as transliteration candidates $C(a_j) = \{c_{j1}, c_{j2}, ..., c_{jt}\}$, where c_{ji} denotes the i^{th} candidate word found in the j^{th} Arabic sentence, and t is the total number of Arabic candidates in $C(a_j)$. An external multilingual transliteration tool [2] is used to obtain an approximate Arabic transliteration t_i of each English named entity n_i. For each English named entity n_i having the approximate transliteration t_i and the list of Arabic candidates $C(a_j)$, the score of each candidate is estimated using the following three functions:

[1] To obtain a version of the English-Arabic named entity transliteration corpus, please contact the authors of this paper.
[2] The details of all the used tools will be provided in the test and experiment section 5.

1. The number of common characters: this function is used to find the number of shared characters between each candidate c_{ji} and the approximate transliteration t_i.
2. Longest common sequence: this function is used to compute the length of the longest shared sequence of characters between each candidate c_{ji} and the approximate transliteration t_i.
3. Length difference penalty: this function is used to penalize the length difference between each candidate c_{ji} and the approximate transliteration t_i.

The final score for each candidate is then estimated as the average sum of all the considered scoring functions. The candidate having the highest score is then selected if its corresponding final score surpasses a certain confidence threshold. Some examples of the extracted English-Arabic named entities are provided in Table 1. The reader should recall that the Arabic language has no letters for the English sound "v", "p" and "g".

Table 1: Some examples of the extracted English-Arabic named entities

Entity class	English	Arabic
PERSON	Villalon	فيلالون (filaloun)
LOCATION	Nampa	نامبا (namba)
ORGANIZATION	Soogrim	سوغريم (soughrim)

4. The Transliteration System

This section describes our proposed model for machine transliteration. Our model uses the Encoder-decoder architecture proposed by Cho et al. [8] which has been proven to perform extremely well in many sequence-to-sequence applications [15].

Given an input sequence $x = \{x_1, x_2, ..., x_d\}$ and an output sequence $y = \{y_1, y_2, ..., y_d\}$, where each x_t and y_t represent the input and output tokens at time-step t respectively, and d is the maximum sequence length[3], the basic Encoder-decoder uses two Recurrent Neural Networks (RNNs) that will be trained jointly to generate the target sequence y given the input sequence x. The encoder RNN encodes the input sequence into a hidden representation $h = \{h_1^e, h_2^e, ..., h_k^e\}$, where h_t^e is the encoder hidden state at time-temp t and k in the dimension of the hidden representation $\mathbb{R}^k$ (Eq. 1).

$$h_t^e = \sigma(W_e x_t + U_e h_{t-1}^e + b_e) \tag{1}$$

where W_e and U_e are the encoder weight matrices, b_e is the encoder bias vector and σ is a logistic sigmoid activation function. The last hidden state h_k^e will be the summary of all the input sequence x.

The decoder RNN takes the encoder summary h_k^e and a target token o_t at each time step t along with its previous decoder hidden state h_{t-1}^d to estimate the value of its current state h_t^d using Eq. 2:

$$h_t^d = \sigma(W_d o_t + U_d h_{t-1}^d + C_d h_k^e + b_d) \tag{2}$$

where W_d, U_d and C_d are the decoder weight matrices, b_d is the decoder bias vector and σ is a logistic sigmoid activation function. Then the output sequence is predicted (generated) token by token at each time step t from the target vocabulary using a softmax activation function (Eq. 3).

$$o_t^d = Softmax(V_d h_t^d) \tag{3}$$

where V_d is the target vocabulary weight matrix. Figure 2 shows the global architecture of the encoder-decoder model being used for the task of transliteration from English-to-Arabic. The first RNN encodes a padded variable length English named entity into a fixed hidden representation and the second RNN (the decoder) generates a transliteration output from the hidden representation.

[3] Padding is used to pad all the input sequences into the same length

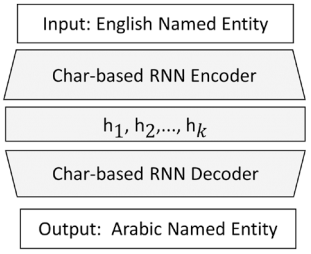

Fig. 2: The global architecture of the Encoder-decoder model being used for the task of transliteration from English-to-Arabic

4.1. The Gated Recurrent Unit

Standard Recurrent Neural Networks (RNNs) [16] are known to have a limitation when dealing with long-term dependencies. It arises when the input sequence spans long intervals causing what is known as vanishing and exploding gradient problems [17] in the Backpropagation Through Time (BPTT) training algorithm [18]. A common solution is to use either the Long Short-term Memory (LSTM) [19] or the Gated Recurrent Unit (GRU) [20] neural networks, which solve these problems and have proven to perform equally well at capturing long-term dependencies.

In this work, a GRU unit (proposed in [20]) is used in both the encoder and decoder layers. The GRU does not use the activation function directly in each recurrent component; instead, two gates are used to control the flow of information throughout the network. The first one is called a reset gate r, which is responsible for combining the previous memory with the new input, allowing the cell to remember or forget information as needed. The second gate is an update gate z, which decides how much activation should be kept from the previous memory. These two gates can take values between 0 and 1, where a value of zero indicates that the gate is closed, a value of 1 indicates that the gate is completely open.

Given an input sequence $x = \{x_1, x_2, ..., x_d\}$, the activation h_t^j at time-step t of the j^{th} GRU unit is estimated by combining the previous activation h_{t-1}^j with a candidate activation $\tilde{h}_t^j$ (Eq. 4).

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j\tilde{h}_t^j \tag{4}$$

in which the candidate activation $\tilde{h}_t^j$ is estimated using Eq. 5.

$$\tilde{h}_t^j = \sigma(Wx_t + U(r_t^j \circ h_{t-1}^j) + b_h) \tag{5}$$

The update gate z_t^j and the reset gate r_t^j are calculated using Eq. 6 and Eq. 7 respectively.

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1}^j + b_z) \tag{6}$$

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1}^j + b_r) \tag{7}$$

where W_z, U_z and W_r, U_r are the weight matrices concerning the update and reset gates, $\circ$ is an element-wise multiplication, the b term denotes the bias vector and σ is a logistic sigmoid activation function.

4.2. Bidirectional Encoder

Bidirectional Recurrent Neural Networks (BiRNNs) [21] are an extension of the standard RNNs consisting of a forward and a backward RNN cells that scan the input sequence in the direct and reversed directions. The most interesting thing about this BiRNN is that at the level of each time-step t we will have a summary of the whole input sequence surrounding that step (its left and right contexts). In this work, our bidirectional encoder is composed of two GRU cells as shown in Fig. 3.

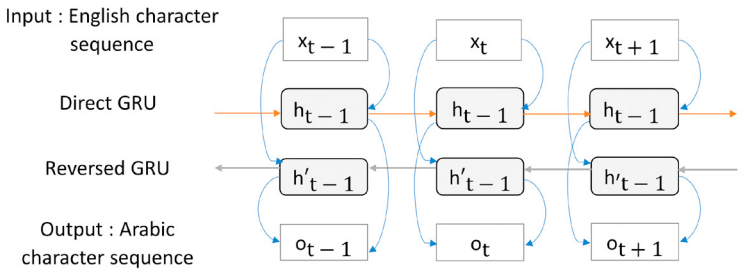

Fig. 3: A bidirectional English-to-Arabic encoder with a direct and reversed GRU cells

4.3. The Decoder Attention Mechanism

The basic encoder-decoder architecture compresses all the input sequence into a single fixed-length hidden representation h_k^e. This limited representation does not always preserve all the necessary aspects contained in the input sequence which leads to a degradation in performance when dealing with long sequences. Instead of considering only the last hidden state, the Attention Mechanism [7] allows to take into account all the encoder hidden states $\{h_1^e, h_2^e, ..., h_k^e\}$ and learns to focus the attention only on the most important ones among them. The Attention decoder uses the Eq. 8 to estimate the value of the hidden state at time-step t.

$$h_t^d = \sigma(W_d o_t + U_d h_{t-1}^d + C_d Att_t^d + b_d) \tag{8}$$

where Att_t^d is the weighted sum of all the encoder hidden states (Eq. 9).

$$Att_t^d = \sum_{j=1}^{k} (\alpha_{tj} h_j^d) \tag{9}$$

A single layer feed-forward neural network is then used to estimate these coefficients α_{tj} for each hidden state h_j^d. These coefficients are values between 0 and 1 denoting the level of focus that should be made by the decoder.

5. Tests and Analysis of the Results

This section presents an in-depth discussion and the details of the tests performed. These tests examine two important aspects. First, we want to investigate the effect of using sequence-to-sequence deep neural network models on the task of transliteration between English and Arabic. The second aspect aims at comparing the performance of our proposal with the Phrase-based Statistical Machine Translation system which has been proven to perform very well at sequence-to-sequence prediction tasks. We start this section by presenting some statistics about the data we have used and the preprocessing stage. Then we provide all the hyperparameters that have been incorporated in our models. Finally, we address the two aforementioned key tests.

5.1. Data and Preprocessing

To build our transliteration corpus, we have used a set of four English-Arabic parallel corpora obtained from the "lingfil" website[4]. The statistics of these corpora are provided in Table 2.

For the Arabic language, our preprocessing includes the removal of diacritic signs, the normalization of Arabic characters and word tokenization using the Python NLTK toolkit [5]. For the English side, only a word tokenization is performed using the same NLTK toolkit. English named entities were identified by means of the Stanford Named

[4] http://opus.lingfil.uu.se
[5] http://www.nltk.org/

Table 2: Statistics about the used English-Arabic parallel corpora

Corpus	Sentences (in millions)
United Nation	10.6M
Open Subtitles	24.4M
News Commentary	0.2M
IWSLT2016	0.2M
All	35.4M

entity recognition system [22] [6]. A vocabulary containing the $n = 40000$ most frequent words has been used to filter the functional words during our corpus construction phase (Section 3). The approximate transliterations were obtained using the polyglot multilingual NLP library [23] [7]. Table 3 shows the statistics about the count of each named entity class that is found in our constructed transliteration corpus.

Table 3: The count of the Person, Location and Organization named entities present in our constructed transliteration corpus

Named entity	Count
Person	61,662
Location	12,679
Organization	5,583
All	79,924

The corpus has been divided into training, development, and test data as shown in Table 4.

Table 4: Instance counts in the train, development and test datasets of our transliteration corpus

Sets	Train	Dev	Test
Named entities count	75,898	1004	3013

5.2. Phrase-based SMT

In the sequel, we provide a brief introduction to the functioning mechanism of our Phrase-based SMT model along with the hyperparameters. Given a source input sequence f that we want to translate (in our case transliterate) into a target sequence e, the phrase-based statistical machine translation [24, 25, 26] finds the best translation $\hat{e}$ from the space of all possible translations of f.

$$\hat{e} = argmax_e = p(e|f) \tag{10}$$

The noisy channel decomposition [27] is then used to split the model into a translation model $p(f|e)$ and a language model $p(e)$.

$$\hat{e} = argmax_e = (p(e) * p(f|e)) \tag{11}$$

The translation model is used to measure the accuracy of the translation hypothesis, and the language model is used to ensure its fluency. One common extension to this basic formulation is the log-linear model [27] which enables the incorporation of multiple features, with the assumption that these features are independent from each other.

$$\hat{e} = argmax_e = (\frac{exp^{\sum_1^M \alpha_m \varphi_m(f,e)}}{\sum_{e'} exp^{\sum_1^M \alpha_m \varphi_m(f,e')}}) \tag{12}$$

[6] https://nlp.stanford.edu/software/CRF-NER.shtml
[7] http://polyglot.readthedocs.io

M denotes the number of features, $\varphi_m(f,e)$ is the m^{th} feature and α_m is its corresponding weight in the log-linear model. Given that the denominator $\sum_{e'} exp^{\sum_1^M \alpha_m \varphi_m(f,e')}$ is constant for all possible translations e', it can be omitted at decoding time.

$$\hat{e} = argmax_e = exp^{\sum_1^M \alpha_m \varphi_m(f,e)} \tag{13}$$

The log-linear features are trained using the Minimum Error Rate Training (MERT) [28].

PSMT Hyperparameters:. We used the Phrase-based SMT Model via the Moses framework [29] [8]. Our log-linear PSMT model includes the following components:

- A phrase translation model with a maximum phrase length of 7 tokens.
- A trigram target language model.

In our configuration, the distance-based reordering model has been turned-off since no character-based target reordering is needed in the transliteration task. The default values have been kept unchanged for all of the remaining Moses hyperparameters.

5.3. Encoder-decoder Models

We have investigated the use of both a single GRU encoder and a Bidirectional GRU encoder, along with the presence and absence of the decoder attention mechanism. This led to four different systems:

1. Seq2seq basic: A stranded GRU encoder and decoder.
2. Bi-seq2seq: A Bi-directional encoder with a stranded GRU decoder.
3. Att-seq2seq: A stranded GRU encoder and attention-based decoder.
4. Bi-Att-seq2seq: A Bi-directional encoder and an attention-based decoder.

Hyperparameters:. To choose an adequate number of neurons in our encoder and decoder GRU cells, we have investigated the effect of changing the number of neurons by measuring the error rates we have obtained for each neural configuration and all the four encoder-decoder systems. This variation is shown for the Bi-Att-seq2seq model (Fig. 4).

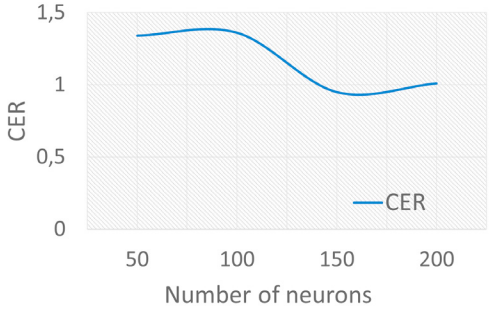

Fig. 4: Results for the English-to-Arabic transliteration when varying the encoder-decoder hidden sizes

Figure 4 shows the effect of varying the network size (the number of neurons in each layer of the encoder-decoder architecture) reported in terms of Character Error Rate (CER) for the Bi-Att-seq2seq model. The best performance in terms of CER has been achieved when the number of neurons was fixed to 150. The training perplexity for this same configuration is shown in Fig. 5.

[8] http://www.statmt.org/moses/

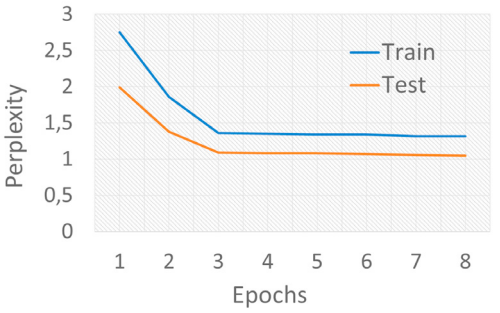

Fig. 5: Perplexity variation on the train and test data for the attention bi-directional encoder-decoder model for the English-to-Arabic transliteration task

The train and test perplexity values (Fig. 5) decrease with the number of epochs until convergence is reached starting from the fourth epoch. In a similar way, we have fixed all the remaining hyperparameters experimentally. In all our sequence-to-sequence models, an embedding dimension of $\mathbb{R}^{10}$ has been used. The maximal sequence length has been set to 50 characters. A mini-batch of size 128 have been incorporated. The training has been done by means of Stochastic Gradient Descent (SGD) with the Adam optimization function [30]. Our models have been trained using the Nvidia $GTX1070$ GPU with $8GB$ of $DDR5$ memory. All the encoder-decoder models have been trained using the OpenNMT toolkit [31] [9].

5.4. Results

The Word Error Rates (WERs) and Character Error Rates (CERs) obtained by all the investigated sequence-to-sequence models are provided in Table 5a and Table 5b for the tasks of transliteration from English-to-Arabic and Arabic-to-English respectively.

Table 5: Transliteration results

Metrics	WER	CER
Seq2seq basic	16.91	3.44
Bi-seq2seq	16.11	3.07
Att-seq2seq	8.81	1.51
Bi-Att-seq2seq	**5.40**	**0.95**
Moses PSMT	6.96	1.03

(a) English-to-Arabic

Metrics	WER	CER
Seq2seq basic	70,02	19.25
Bi-seq2seq	74,53	21.49
Att-seq2seq	66,49	17.10
Bi-Att-seq2seq	**65,16**	**16.35**
Moses PSMT	68,57	16.61

(b) Arabic-to-English

The reported error rates for the English-to-Arabic direction (Table 5a) show that the incorporation of the decoder attention mechanism leads to a significant improvement. Indeed, around 10% WER reduction has been achieved over the basic encoder-decoder when the decoder attention mechanism was incorporated. The PSMT model and the Bi-Att-seq2seq models gave the best results of 6.96% and 5.40% in terms of WER respectively, with a slightly better performance in favor of the attention model. The results for Arabic-to-English direction (Table 5b) were not as good. This is due to the higher ambiguity that is present in this direction, which has also been pointed by Rosca et al. [14] and Deselaers et al [1]. As for the English-to-Arabic test results, the Bi-Att-seq2seq gave the best performance of 65,16% WER followed by the PSMT with a 68.57% WER.

Table 6 provides a comparison between our best models, the Bi-Att-seq2seq and the Moses PSMT systems, to some other research works that have been made in the task of machine transliteration from Arabic-to-English [10].

[9] `http://opennmt.net/`

[10] For the other direction (English-to-Arabic) we have not found any research work that uses our same metrics.

Table 6: Comparing our proposed approaches to some other Arabic-to-English deep learning based transliteration systems

Metrics	CER	WER
Rosca et al. [14]	22.5	78.5
Desela et al. [1]	20,1	-
Our Bi-Att-seq2seq	**16,35**	**65.1**6
Our Moses PSMT	16,61	68.57

The results show that our Bi-Att-seq2seq model gives the lowest error rate, which demonstrates the efficiency of our proposal for the transliteration task [11].

5.5. Error Analysis

In this section, we will provide a quick glance at the errors made by our best Bi-Att-seq2seq model for the task of machine transliteration.

Table 7: Example of some errors made for English-to-Arabic transliteration

Input	Reference	System output
Brandes	براندیس (Brandees)	براندس (Brandes)
Mayhawk	مايهوك (Mayhouk)	مايهاوك (Mayhawk)

As shown in Table 7, for the English-to-Arabic direction most of the errors made by the transliteration system are due to conflicting vowels. Indeed, this is to be expected since there are no unified ways of transliterating named entities between Arabic and English. Instead, many possible transliterations can be made for the same named entity. For instance, a "ي" can either be considered or omitted in the transliteration of "Brandes" as shown in Table 7.

Concerning the Arabic-to-English direction the reported results were much worst then the one reported from English-to-Arabic. This is maybe due to the absence of vowels (diacritic signs) on the Arabic side, for instance, words such as "حسين" and "حسن" are transliterated to "**Houc**ine" and "**Hass**an" respectively even though they share the same prefix "حس". Another problem is the absence of some Arabic character sounds in the English language which often get conflicted, such as the sounds of the Arabic letters "ط", "ظ" and "ث".

6. Conclusion

In this work, we have presented a bidirectional attention-based encoder-decoder model for the task of machine transliteration between Arabic and English. Our system has been compared to several sequence-to-sequence models, and its results prove the efficiency of our proposal.

The contributions of this work can be summarized as follows:

- This work addresses the case of machine transliteration between English and Arabic using a deep learning sequence-to-sequence model, which has not been investigated before for the English-to-Arabic direction.
- We have demonstrated a method that allows automatic construction of a transliteration corpus from a raw English-to-Arabic parallel corpus.
- We have open-sourced the constructed corpus for the use by the NLP community for research purposes.
- A comparative study has been established between several sequence-to-sequence systems.

[11] We note that the systems we have compared have not been tested on the same test set, thus conclusions should be taken with some caution.

This work can be further developed in various directions. One direction is to consider the case of transliteration between Arabic and other languages besides English. Another interesting future direction is to integrate this model into an English-to-Arabic machine translation system to address the problem of named entity transliteration.

References

[1] T. Deselaers, S. Hasan, O. Bender, H. Ney, A deep learning approach to machine transliteration, in: Proceedings of the Fourth Workshop on Statistical Machine Translation, Association for Computational Linguistics, 2009, pp. 233–241.

[2] A. H. Jadidinejad, Neural machine transliteration: Preliminary results, arXiv preprint arXiv:1609.04253.

[3] U. Hermjakob, K. Knight, H. Daumé III, Name translation in statistical machine translation-learning when to transliterate., in: ACL, 2008, pp. 389–397.

[4] N. Habash, Four techniques for online handling of out-of-vocabulary words in arabic-english statistical machine translation, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, Association for Computational Linguistics, 2008, pp. 57–60.

[5] P. Virga, S. Khudanpur, Transliteration of proper names in cross-lingual information retrieval, in: Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15, Association for Computational Linguistics, 2003, pp. 57–64.

[6] A. Fujii, T. Ishikawa, Japanese/english cross-language information retrieval: Exploration of query translation and transliteration, Computers and the Humanities 35 (4) (2001) 389–420.

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

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

[9] Y. Shao, J. Nivre, Applying neural networks to english-chinese named entitytransliteration, in: Sixth Named Entity Workshop, joint with 54th ACL, 2016.

[10] A. Finch, L. Liu, X. Wang, E. Sumita, Target-bidirectional neural models for machine transliteration, ACL 2016 (2016) 78.

[11] L. Jiang, M. Zhou, L.-F. Chien, C. Niu, Named entity translation with web mining and transliteration., in: IJCAI, Vol. 7, 2007, pp. 1629–1634.

[12] M. Arbabi, S. M. Fischthal, V. C. Cheng, E. Bart, Algorithms for arabic name transliteration, IBM Journal of research and Development 38 (2) (1994) 183–194.

[13] N. AbdulJaleel, L. Larkey, English to arabic transliteration for information retrieval: A statistical approach, Center for Intelligent Information Retrieval Computer Science, University of Massachusetts.

[14] M. Rosca, T. Breuel, Sequence-to-sequence neural network models for transliteration, arXiv preprint arXiv:1610.09565.

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

[16] R. J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural computation 1 (2) (1989) 270–280.

[17] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE transactions on neural networks 5 (2) (1994) 157–166.

[18] C. Goller, A. Kuchler, Learning task-dependent distributed representations by backpropagation through structure, in: Neural Networks, 1996., IEEE International Conference on, Vol. 1, IEEE, 1996, pp. 347–352.

[19] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[20] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv preprint arXiv:1409.1259.

[21] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE Transactions on Signal Processing 45 (11) (1997) 2673–2681.

[22] J. R. Finkel, T. Grenager, C. Manning, Incorporating non-local information into information extraction systems by gibbs sampling, in: Proceedings of the 43rd annual meeting on association for computational linguistics, Association for Computational Linguistics, 2005, pp. 363–370.

[23] Y. Chen, S. Skiena, False-friend detection and entity matching via unsupervised transliteration, arXiv preprint arXiv:1611.06722.

[24] P. F. Brown, J. Cocke, S. A. Della-Pietra, V. J. Della-Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, P. Rossin, A statistical approach to machine translation, Computational Linguistics 16 (2) (1990) 76–85.

[25] R. Zens, F. J. Och, H. Ney, Phrase-based statistical machine translation, in: Annual Conference on Artificial Intelligence, Springer, 2002, pp. 18–32.

[26] F. J. Och, H. Ney, Discriminative training and maximum entropy models for statistical machine translation, in: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2002, pp. 295–302.

[27] P. Koehn, Statistical Machine Translation, 1st Edition, Cambridge University Press, New York, NY, USA, 2010.

[28] F. J. Och, Minimum error rate training in statistical machine translation, in: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1, Association for Computational Linguistics, 2003, pp. 160–167.

[29] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, et al., Moses: Open source toolkit for statistical machine translation, in: Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions, Association for Computational Linguistics, 2007, pp. 177–180.

[30] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[31] G. Klein, Y. Kim, Y. Deng, J. Senellart, A. M. Rush, Opennmt: Open-source toolkit for neural machine translation, arXiv preprint arXiv:1701.02810.