Improving English-Arabic Transliteration with Phonemic Memories

Yuanhe Tian[♠♥], **Renze Lou**[♡], **Xiangyu Pang**[◇]
Lianxi Wang[♣], **Shengyi Jiang**[♣], **Yan Song**[♠†]

[♠]University of Science and Technology of China [♥]University of Washington
[♡]Temple University [◇]China Merchants Securities
[♣]Guangdong University of Foreign Studies

[♥]yhtian@uw.edu [♡]renze.lou@temple.edu [◇]xypang11@foxmail.com
[♣]wanglianxi@gdufs.edu.cn [♣]jiangshengyi@163.com [♠]clksong@gmail.com

Abstract

Transliteration is an important task in natural language processing (NLP) which aims to convert a name in the source language to the target language without changing its pronunciation. Particularly, transliteration from English to Arabic is highly needed in many applications, especially in countries (e.g., United Arab Emirates (UAE)) whose most citizens are foreigners but the official language is Arabic. In such a task-oriented scenario, namely transliterating the English names to the corresponding Arabic ones, the performance of the transliteration model is highly important. However, most existing neural approaches mainly apply a universal transliteration model with advanced encoders and decoders to the task, where limited attention is paid to leveraging the phonemic association between English and Arabic to further improve model performance. In this paper, we focus on transliteration of people's names from English to Arabic for the general public. In doing so, we collect a corpus named *EANames* by extracting high quality name pairs from online resources which better represent the names in the general public than linked Wikipedia entries that are always names of famous people). We propose a model for English-Arabic transliteration, where a memory module modeling the phonemic association between English and Arabic is used to guide the transliteration process. We run experiments on the collected data and the results demonstrate the effectiveness of our approach for English-Arabic transliteration.[1]

1 Introduction

With the rapid development of globalization, the number of people working and living cross countries has been significantly grown in the past decades. Therefore, there are increasing needs of transliterating the name of the migrants from their original language to the local language, especially for countries such as United Arab Emirates (UAE), whose majority citizens are migrants coming from foreigner countries such as India, Pakistan, and Philippines, etc. In these cases, a well-performing model to transliteration the names of the general public from one language (e.g., English) to the other (e.g., Arabic) is highly needed.

Conventional studies use grapheme-based, phoneme-based, and hybrid approaches to learn the mapping of the features and phonemes between the source and target languages (Arbabi et al., 1994; Knight and Graehl, 1998; Li et al., 2004; Habash et al., 2007; Pervouchine et al., 2009; Ravi and Knight, 2009; Kumar and Kumar, 2013). Recently, end-to-end neural approaches are also applied to transliteration tasks (Finch et al., 2016; Hadj Ameur et al., 2017; Upadhyay et al., 2018; Kundu et al., 2018; Moran and Lignos, 2020; Alkhatib and Shaalan, 2020) and achieve good performance. Among these studies, most neural approaches focus more on using advanced encoders and decoders (such as RNN, GRU, LSTM, and Transformer (Vaswani et al., 2017)) following the standard sequence-to-sequence paradigm, where less attention is paid to leverage the phonemic information of the source and target languages to improve model performance, especially for transliteration from English to Arabic.

In this paper, we focus on transliteration from English to Arabic[2] for the general public. To train a model, it is intuitive to directly use existing corpora for named entity transliteration (Rosca and Breuel, 2016; Chen and Skiena, 2016; Merhav and

[†]Corresponding author.

[1]The source of the paper is available under `https://github.com/synlp/EATrans`.

[2]In this paper, we follow previous studies to focus on the transliteration to modern standard Arabic (MSA) since it is used in formal occasions where name transliteration is needed.

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 3262–3272
December 7-11, 2022 ©2022 Association for Computational Linguistics

Ash, 2018; Chen et al., 2018; Benites et al., 2020; Murikinati et al., 2020). However, many of those corpora are not public available and most of them are constructed by extracting the named entity pairs from linked Wikipedia entries without distinguishing different types of named entities. Therefore, the instances from the other types of named entities (e.g., locations) could introduce noise to a model that is designed mainly for name transliteration. More importantly, since the names in Wikipedia are more likely to come from famous people, they may fail to represent the names in the general public. As a result, models trained with existing corpora may not be satisfying in real applications (e.g., transliterate a migrant's name from English to Arabic). Therefore, we collect a corpus named *EANames* for English-Arabic transliteration. The training and development data of *EANames* are obtained by crawling English names from the profile of LinkedIn[3] users who are working in a representative Arabic speaking contrary, i.e., UAE, and the corresponding Arabic ones are obtained from well-known transliteration systems and human annotation. The test set contains 3,000 English-Arabic name pairs extracted from large English-Arabic parallel news corpus with their quality justified by well-known transliteration systems and human annotation.

Besides, we propose a neural sequence-to-sequence model following the encoder-decoder paradigm for English-Arabic transliteration, which is enhanced by phonemic memories that are designed to memorize the phonemic association between English and Arabic names and thus guide the transliteration process. Specifically, for each input English letter, we firstly associate it with the phonemic symbols extracted from a phoneme inventory constructed based on the phonetic systems of English and Arabic. Next, in the memory module, we weigh the associated symbols according to their contribution to the transliteration process and incorporate the weighed information into the backbone encoder-decoder model. We run experiments with different models on *EANames*, where our model outperforms strong baselines and achieves the bast results under different metrics on the test set.

2 The *EANames* Corpus

Generally, it is significantly important to train and evaluate models on data that is similar to the ones in real applications. However, most public available transliteration corpora are constructed from associated Wikipedia entity entries in different languages and thus are more representative to names of famous people rather than the general public. Therefore, instead of using existing corpora for English-Arabic transliteration, we collect a corpus named *EANames* with English names and their Arabic counterparts collected from online resources and human annotation.

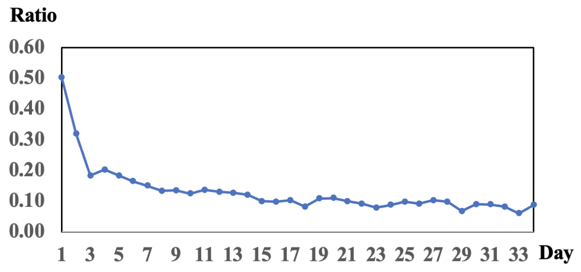

Figure 1: The ratio of the number of newly crawled English names and the size of the crawled datasets with respect to the crawling time.

In doing so, we use different approaches to extract training and test data because they have various requirements on the quantity and quality. Generally, the quantity of training data is much more important, where some noises in it would not significantly hurt the performance of a model trained on it; on the contrary, test data requires more on the quality with limited tolerance on mislabeled data. In the following text, we illustrate our approaches to extract the training and test data and finally report the statistics of the collected *EANames*.

2.1 Training Data Collection

Generally, a larger training data results in a better performing model, since it reduces the rate of unseen instances during the test time. Therefore, to train a well performing model, it is important to collect as many English names, as well as their Arabic transliterations, as possible. Online social platform could be a good resource to collect English names, especially the platforms for job hunting on which users are more likely to post their real names. Therefore, for the training data, we collect English names from online resources and then use machine translation/transliteration systems to obtain the corresponding Arabic names.

Raw Data Collection We use LinkedIn for training data collection. As one of the world-famous online platforms primarily used for registered users to build networks with others, LinkedIn allows job seekers to post their resumes and employers to post

[3]https://www.linkedin.com/

Crawled	117,434
Auto Annotated	68,553
Human Annotated	36,668
Final	105,221

Table 1: The statistics of the unique English name collected from LinkedIn users from UAE. "Crawled" refers to the number of English names crawled from LinkedIn. "Auto Annotated" and "Human Annotated" mean the the number of English names with silver standard Arabic transliteration obtained from the transliteration system voting and human annotation, respectively. "Final" is the final number of English names with silver standard Arabic transliteration.

jobs, where registered users create their profiles including their names, affiliations, regions, experience, etc. We crawl the English names of the registered users in LinkedIn from their profiles. Since most needs of English-Arabic transliteration locate in the Mideast countries, we use United Arab Emirates (UAE) as a representative country and mainly crawl English names from the users who are working in UAE. When crawling, we split the crawled name with more than one word into multiple names, where each resulting English name corresponds to a single word. To monitor the progress of crawling, we compute the ratio of the number of newly crawled names and the size of the crawled dataset. We stop crawling when the ratio converge to a low degree, which means most widely used English names have been crawled. The daily ratio of the crawled names is visualized in Figure 1. The number of the unique English names crawled from LinkedIn users from UAE is reported in the first row (i.e., "Crawled") of Table 1.

Data Annotation To obtain the Arabic transliteration of the crawled datasets, we use several existing machine translation/transliteration systems to transliterate the crawled English names into Arabic and then ask them to vote for the best candidate. In practice, we use the three systems, namely, Google translation, Bing translation, and Bing transliteration, and selected the candidate agreed by at least two systems as the sliver standard for Arabic name transliteration.[4] Through this process, we obtain the auto-annotated silver standards for some of the crawled English names, where the statistics are reported in the second row (i.e., "Auto Annotated") in Table 1. For the remaining crawled English names without silver standard, we asked two human annotators who can speak English and Arabic to annotated whether the Arabic transliteration of the Bing transliteration system[5] could be used as the sliver standard (in other words, whether the Arabic name is similar to the original English name in terms of their pronunciation). In doing so, we split the data into two groups where 10% of them are shared by both annotators. For the shared data, if there are disagreements, the annotators are asked to discuss and resolve it. Table 2 reports the inter-annotator agreement (Cohen's kappa) with the high kappa confirms the high quality of the annotation. The number of Arabic names that are annotated to be used as the silver standard is reported in the third row (i.e., "Human Annotated") of Table 1.

Agreement (Cohen's kappa)	0.934

Table 2: The inter-annotator agreement (Cohen's kappa) of two annotators on the shared data.

Finally, we collect the English-Arabic name pairs annotated by transliteration system voting and human annotation and obtain 105,221 name pairs which are reported in the last row of Table 1.

2.2 Test Data Collection

The quality of the test set is significantly important, but it is remarkably expensive to create a test set by manually annotating Arabic names with the given English ones from stretch. Therefore, we propose to automatically extract English-Arabic name pairs from existing English-Arabic parallel corpus, which contains more English and Arabic name pairs from people in the general public than linked Wikipedia entries. The details of the raw parallel corpus and the extraction process are elaborated in the following text.

The Raw Parallel Corpus We use the ISI Arabic-English Automatically Extracted Parallel Text corpus[6] to extract English-Arabic name pairs. The corpus contains news articles published by Xinhua News Agency and Agence France Presse and thus are more likely to contain names from the general public. In addition, the raw corpus is significantly

[4]We use the Google Translation API provided by Google Could (https://cloud.google.com/) and Microsoft Translation and Transliteration API provided by Microsoft Azure (https://azure.microsoft.com/).

[5]We use the Arabic transliteration from Bing transliteration system because it is originally designed for transliteration and thus is more likely to generate a plausible transliteration.

[6]https://catalog.ldc.upenn.edu/LDC2007T08/

large, where there are more than 1M English sentences paired with a parallel Arabic sentence. As a result, the corpus is an appropriate choice for name pair extraction.

English-Arabic Name Pair Extraction In this paper, we use an existing named entity tagger to extract English and Arabic names from the parallel corpus. Specifically, we employ the English and Arabic named entity taggers from Stanford CoreNLP Toolkits[7] (Manning et al., 2014). The detailed process to extract English-Arabic name pairs from the parallel text is described as follows.

Firstly, for each English-Arabic sentence pair in the parallel corpus, we use the named entity taggers to extract the person's name from the English and Arabic sentences. If exactly one person's name is recognized in both English and Arabic sentences, we collect the names from the sentences and regard them as an English-Arabic name pair. Herein, we exclude sentence pairs where the taggers extract multiple person's names from either sentence, and thus we do not need to annotate the exact mappings between multiple English and Arabic names. This step results in 74,772 English-Arabic name pairs.

Next, since generally, one word in an English name corresponds to one word in an Arabic name (and vice versa), we filter out English-Arabic name pairs where the number of words on the Arabic side does not match the number of words on the English side. For each English-Arabic name pair, we also split the pair into multiple English-Arabic name pairs if there are more than one word in the English and Arabic name. For example, an English-Arabic name pair (*Ahmad Hussein*, أحمد حسين) with two words on both sides is split into two Arabic-English name pairs, i.e., (*Ahmad*, أحمد) and (*Hussein*, حسين) for the first and last name, respectively. Then, we remove the redundant English-Arabic name pairs and obtain 19,921 name pairs.

Afterwards, to further confirm the Arabic transliteration of the English names, we run aforementioned three machine translation/transliteration systems, i.e., Google translation, Bing translation, and Bing transliteration, to transliterate the English names in the pairs into the corresponding Arabic ones. We keep the English-Arabic name pairs where the transliteration of all the three machine translation/transliteration systems matches the Arabic name in the pair and ask a native Arabic speaker (who are also able to speak English) to double-check the resulting English-Arabic name pairs. Finally, we obtain a corpus with 3,000 English names with their Arabic transliteration.

[7]We use the latest version 4.4.0 downloaded from https://stanfordnlp.github.io/CoreNLP/.

	Train	Dev	Test
# of English-Arabic name pairs	94,688	10,522	3,000
Max length of English names	24	18	15
Max length of Arabic names	33	23	15
Min length of English names	2	2	2
Min length of Arabic names	2	2	2
Avg. length of English names	6.9	7.0	6.4
Avg. length of Arabic names	6.1	6.2	5.7

Table 3: The statistics of the English and Arabic name pairs in the train, dev, and test sets of *EANames*. The length of names is based on the number of letters.

2.3 The Statistics of *EANames*

To summarize, *EANames* contains a large number of English-Arabic name pairs, where the training data of *EANames* is relatively large and the test data is of high quality since the test name pairs are extracted by well performing named entity taggers with the Arabic transliterations further confirmed by well-known machine transliteration systems and human annotators. Furthermore, since all names are extracted from online resources, they are more representative for the general public than the names extracted from linked Wikipedia entries, which are usually names of famous people. In experiments, we randomly sample 10% name pairs from all training data and use them as the development set. The statistics of the final training, development, and test sets in *EANames* are reported in Table 3, where the length of names is based on the number of English and Arabic letters.

3 The Approach

In this paper, we propose a neural model for transliteration from English to Arabic, where the architecture is illustrated in Figure 2. Overall, our approach follows the sequence-to-sequence paradigm for text generation, where phonemic memories are proposed and added on the top of every encoder layer to enhance the model performance by modeling the phonemic information extracted from a phonemic inventory $\mathcal{V}$. At the time step t, we denote the input English name as $\mathcal{X} = x_1 \cdots x_i \cdots x_n$ with n letters and the existing output Arabic name as $\mathcal{Y}_{t-1} = y_1 \cdots y_{t-1}$. Therefore, the object of our transliteration model is to predict the next Arabic

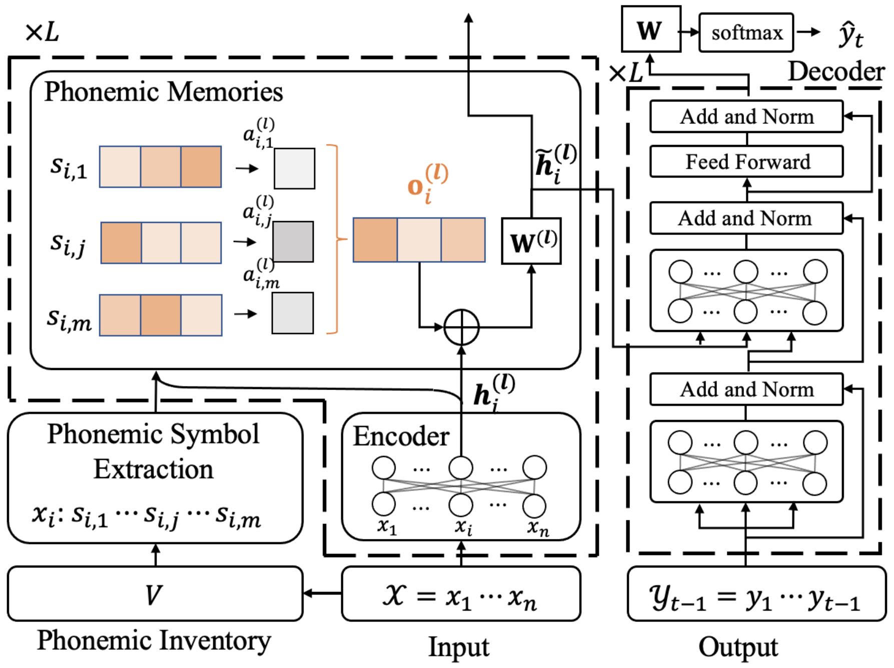

Figure 2: The overall architecture of the proposed model following the sequence-to-sequence paradigm. The proposed phonemic memories is added on the top of every encoder layer, where the phonemic information associated with each input letter is weighed and incorporated into the transliteration process.

letter $\widehat{y}_t$ through

$$\widehat{y}_t = f_d\left(\mathcal{M}\left(f_e\left(\mathcal{X}\right), \mathcal{V}\right), \mathcal{Y}_{t-1}\right) \qquad (1)$$

where f_d and f_e are standard decoder and encoder, respectively, and $\mathcal{M}$ stands for the proposed memory module. In the following text, we first illustrate the process to extract the phonemic information of each input letter. Then we introduce the memory module. Finally elaborate the transliteration process with the phonemic memories.

3.1 Phonemic Information Extraction

In the conventional phoneme-based approaches for transliteration, the word in the source language is firstly converted into a sequence of universal phonemic symbols (e.g., the international phonetic alphabet (IPA)) to represent its sound in the source language. Then, the universal phonemic symbols are modified to represent the sounds in the target language and finally converted into the word in the target language accordingly. Particularly, for the transliteration from English to Arabic, Alshuwaier and Areshey (2011) proposed a rule-based approach which firstly maps an English word to its universal phonemic symbols based on the Carnegie Mellon University pronouncing dictionary[8] (CMUDict), then converts the phonemic symbols into the diacritized Arabic phonemes according to a phoneme set that illustrates the rules for conversion, and finally transforms the diacritized Arabic phonemes to the undiacritized form. Motivated by such process, in our approach for Arabic transliteration, we propose to leverage the phonemic information in the phoneme set proposed by Alshuwaier and Areshey (2011), so as to learn the phonemic association between English and Arabic.

Specifically, we firstly use the universal phonemic symbols in the phoneme set[9] as the phonemic inventory[10] $\mathcal{V}$ in our approach. Next, for each input English letter x_i and each phonemic symbol $s \in \mathcal{V}$, we compute the pointwise mutual information (PMI) score of them by

$$\text{PMI}\left(x_i, s\right) = log\frac{p(x_i, s)}{p(x_i)p(s)} \qquad (2)$$

where $p(x_i)$ and $p(s)$ represent the probability of the English letter x_i and the phonemic symbol s in CMUDict, respectively, and $p(x_i, s)$ denotes the probability that x_i appears in an English word and s is in the corresponding transcription. Herein, a high PMI score indicates that the letter x_i and the phonemic symbol s co-occur a lot in terms of the writing and the corresponding pronunciation, respectively. Therefore, phonemic symbols with higher PMI are more likely to provide useful phone-

[8]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[9]See Table 2 in Alshuwaier and Areshey (2011).

[10]There are 41 distinct phonemic symbols in the inventory.

Letter	Phonemic Symbols				
A	AE0	AA0	EY0	AH0	AW0
D	D	OY0	EY0	AY0	IH0
I	AY0	NG	IH0	ZH	IY0
Q	W	K	EY0	EH0	UH0
W	W	ER2	AW0	DH	HH
Y	OY0	OY1	IY0	Y	AY0

Table 4: An illustration of the top 5 phonemic symbols with the highest PMI scores for some example English letters. The phonemic symbols are based on CMUDict.

mic information for transliteration. Then, for each x_i we rank the phonemic symbols according to their PMI scores. Afterwards, we select the top m phonemic symbols and associate them (denoted as $s_{i,1} \cdots s_{i,j} \cdots s_{i,m}$) with x_i. For reference, Table 4 illustrates the top 5 phonemic symbols of some example English letters. Finally, we feed the phonemic symbols into the phonemic memories to guide the transliteration process.

3.2 Phonemic Memories

To leverage the associated phonemic symbols of x_i, one straightforward approach is to compute the average of their representations and concatenate the resulting vector with the hidden vector of x_i. However, various phonemic symbols may have their distinct contribution in different contexts. Consider that memories have been demonstrated to be effective in encoding and weighing different features in many natural language processing tasks (Miller et al., 2016; Nie et al., 2020; Tian et al., 2020a, 2021; Jain and Lapata, 2021; Chen et al., 2021; Tandon et al., 2022; Tian et al., 2022), we propose to use memories to leverage the phonemic symbols where different weights are assigned to the associated phonemic symbols so as to distinguish their contribution on the transliteration process and leverage them accordingly.

In doing so, for each word x_i, we firstly map every $s_{i,j}$ to its corresponding memory vector $\mathbf{e}_{i,j}$. Next, for the l-th ($1 \leq l \leq L$, L is the total number of encoder layers) encoder layer, we obtain its output hidden vector $\mathbf{h}_i^{(l)}$ for x_i and compute the weight $a_{i,j}^{(l)}$ for $s_{i,j}$ by

$$a_{i,j}^{(l)} = \frac{exp\left(\mathbf{e}_{i,j} \cdot \mathbf{h}_i^{(l)}\right)}{\sum_{j=1}^{m} exp\left(\mathbf{e}_{i,j} \cdot \mathbf{h}_i^{(l)}\right)} \tag{3}$$

where $\cdot$ denotes the inner production of two vectors.

Hyper-parameters	Values
Learning Rate	$1e-4$, $\mathbf{2e-4}$, 5e-4
Warmup Rate	0.06, **0.1**
Dropout Rate	**0.1**
Beam Size	8, **10**
Batch Size	8, 16, **32**

Table 5: The hyper-parameter values tested when tuning our models, and the ones used in our final experiments are in boldface.

Then, we apply the weight $a_{i,j}$ to the corresponding phonemic symbol and compute the weighted sum of the phonemic information (denoted by $\mathbf{o}_i^{(l)}$) by

$$\mathbf{o}_i^{(l)} = \sum_{j=1}^{m} a_{i,j}^{(l)} \mathbf{e}_{i,j} \tag{4}$$

Afterwards, we concatenate $\mathbf{o}_i^{(l)}$ with $\mathbf{h}_i^{(l)}$ to obtain the phonemic enhanced representation and feed the resulting vector to a fully connected layer

$$\widetilde{\mathbf{h}}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \left(\mathbf{o}_i^{(l)} \oplus \mathbf{h}_i^{(l)}\right) + \mathbf{b}^{(l)}\right) \tag{5}$$

where σ stands for the $ReLU$ activation function, $\oplus$ denotes the vector concatenation operation, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are trainable weight matrix and bias vector, respectively, and $\widetilde{\mathbf{h}}_i^{(l)}$ is the output of the memory module and is fed into the decoder and the next encoder layer following the standard process.

3.3 Transliteration with Phonemic Memories

Overall, our transliteration model follows the encoding-decoding paradigm, where a multi-layer encoder and a multi-layer decoder are used and the memory module is added on the top of every encoder layer. Specifically, in the encoder, the l-th layer $f_e^{(l)}$ takes output $\widetilde{\mathbf{h}}_1^{(l-1)} \cdots \widetilde{\mathbf{h}}_n^{(l-1)}$ of the memory module at the (l-1)-th layer (for the first encoder layer, it takes the embedding of the input letter sequence) and compute the output $\mathbf{h}_1^{(l)} \cdots \mathbf{h}_n^{(l)}$ following the standard encoding process (e.g., using multi-head attentions in a transformer-based encoder), which is formally written as

$$\mathbf{h}_1^{(l)} \cdots \mathbf{h}_n^{(l)} = f_e^{(l)}\left(\widetilde{\mathbf{h}}_1^{(l-1)} \cdots \widetilde{\mathbf{h}}_n^{(l-1)}\right) \tag{6}$$

Then, $\mathbf{h}_1^{(l)} \cdots \mathbf{h}_n^{(l)}$ are fed into the memory module to obtain the phonemic enhanced representation $\widetilde{\mathbf{h}}_1^{(l)} \cdots \widetilde{\mathbf{h}}_n^{(l)}$, which are then fed into the next encoder layer and the l-th decoder layer. In the

Models	Dev				Test			
	MRR	ACC	F-score	MAP	MRR	ACC	F-score	MAP
Lookup Table	-	0.00	-	-	-	46.10	-	-
LSTM	63.34	60.91	65.43	85.35	91.33	89.90	92.36	98.25
+ $\mathcal{M}$	**64.55**	**61.95**	**67.09**	**87.21**	**91.94**	**90.87**	**93.13**	**98.45**
Transformer	64.18	61.68	67.01	86.97	91.95	90.97	93.26	98.49
+ $\mathcal{M}$	**65.67**	**62.50**	**68.28**	**87.94**	**93.34**	**92.11**	**95.04**	**98.90**

Table 6: Experimental results of different models with LSTM and Transformer as the encoder and decoder on the development and test set of *EANames*. "+ $\mathcal{M}$" denotes our approach with the memory module. "Lookup Table" is an approach that uses the training set as a lookup table and makes predictions by searching the corresponding Arabic transliteration of the input English name in inference. We only report the accuracy of this approach for reference.

decoder, it takes the existing output $\mathcal{Y}_{t-1}$, as well as the output of the memory module in all layers, so as to predict the current letter $\widehat{y}_t$ in the target language through the standard decoding process. Therefore, the decoding process is formalized as

$$\widehat{y}_t = f_d\left(\mathcal{Y}_{t-1}, \widetilde{\mathbf{H}}^{(1)}, \cdots, \widetilde{\mathbf{H}}^{(L)}\right) \qquad (7)$$

where $\mathbf{H}^{(l)} = \widetilde{\mathbf{h}}_1^{(l)} \cdots \widetilde{\mathbf{h}}_n^{(l)}$ $(1 \leq l \leq L)$ denotes the sequence of the phonemic enhanced representation obtained from the memory module in the l-th layer.

4 Experiments

4.1 Settings

Since neural models have achieved state-of-the-art performance in many natural language processing tasks (Han et al., 2018; Devlin et al., 2019; Radford et al., 2019; Tian et al., 2020b; Lewis et al., 2020; Diao et al., 2020; Raffel et al., 2020; Qin et al., 2021; Diao et al., 2021; Song et al., 2021), including machine transliteration/translation, we try two well-known models, namely, LSTM and Transformer (Vaswani et al., 2017), for the English-Arabic transliteration task. For LSTM and Transformer models, we use two layers for encoding and another two layers for decoding. The hidden vector for each LSTM layer is set to 512. For Transformer, we follow the convention in most existing Transformer-based models, where each layer uses 768 dimensional vectors with 12 heads. The trainable parameters in all models are randomly initialized and updated during training. For other hyper-parameters, we report them in Table 5. We tried all combinations of them and used the ones (highlighted in boldface) that achieve the best performance on the development set in the final experiments. All models are performed on an NVIDIA Tesla V100 GPU with 16G memory.

For evaluation, we use four metrics following previous studies (Song and Kit, 2010; Kumaran et al., 2010; Chen et al., 2018), namely, the mean reciprocal rank (MRR), the top-1 accuracy (ACC), the top-1 mean F-score (F-score), and mean average precision (MAP).[11]

4.2 Overall Results

We run experiments with LSTM and Transformer with and without our memory module (i.e., $\mathcal{M}$) on the collected *EANames* corpus. We run each model five times with different random seeds and report the average results (i.e., MRR, ACC, F-score, and MAP) on the development and the test set in Table 6. For reference, we also employ an approach named "Lookup Table", which uses the training set as an English-Arabic dictionary and predicts the Arabic transliteration of the input English names by searching, and report its accuracy in Table 6.

Here are some observations. First, the Lookup Table approach obtains 0.00% and 46.10% accuracy on the development and test sets, respectively, indicating that there is no overlap between the training and development set and 46.10% test instances are seen in the training data. This observation demonstrates the validity of our approach in collecting training data for transliteration, which results in a high overlapping rate between the training and test data, and thus enhance model performance. Second, overall, the performance of all models on the development set is much lower than that on the test set, which is expected since it is normally challenging to handle unseen cases for any NLP models. Third, for both LSTM and Transformer

[11]We use the top-5 candidates for MRR and MAP.

based models, our approach with memories (i.e., "+ $\mathcal{M}$") outperforms the baselines without the memories with respect to all evaluation metrics, although the baselines have already achieved outstanding performance on the test set. This demonstrates the effectiveness of the proposed memory module in leveraging the phonemic information.

4.3 Ablation Study

To leverage the phonemic information, in our approach, we design an attention mechanism to distinguish the distinct contribution of different phonemic symbols. To explore the effect of the attention design, we conduct experiments with models without the attention mechanism. That is, we compute the average of the memory vectors of all associated phonemic symbols to obtain the phonemic information. We run each model five times with different random seeds and report the average performance in Table 7, where the performance of the standard LSTM and Transformer baseline without the memory module is also reported for reference. It is observed that for both LSTM and Transformer based models, the ablation of the attention mechanism (i.e., "- Att.") significantly hurts the performance of our model. This indicates that equally modeling all associated phonemic symbols could introduce noise to transliterating process, since the contribution of different phonemic symbols varies in a particular context. On the contrary, the attention mechanism in our approach is able to distinguish the contribution of the phonemic symbols and assign different weights to them accordingly, so as to leverage them to improve model performance.

Models	MRR	ACC	F-score	MAP
Full Model	**91.94**	**90.87**	**93.13**	**98.45**
- Att.	91.39	89.93	92.50	98.30
Baseline	91.33	89.90	92.36	98.25

(a) LSTM

Models	MRR	ACC	F-score	MAP
Full Model	**93.34**	**92.11**	**95.04**	**98.90**
- Att.	92.10	91.02	93.39	98.60
Baseline	91.95	90.97	93.26	98.49

(b) Transformer

Table 7: Test set results of models based on LSTM (a) and Transformer (b). "Full Model" denotes our model with the attention mechanism to leverage phonemic information; "- Att." refers to the model where the attention is ablated. We also report the results of standard LSTM and Transformer baselines for reference.

5 Related Work

Transliteration is an important task that is relevant to translation and has been studied for decades. Conventional approaches for transliteration are categorized into grapheme-based, phoneme-based, and hybrid approaches to learn the phonemic connections between the sounds of the source and target languages (Knight and Graehl, 1998; Al-Onaizan and Knight, 2002; Oh et al., 2006; Song et al., 2009; Pervouchine et al., 2009; Ravi and Knight, 2009; Song and Kit, 2010; Alshuwaier and Areshey, 2011; Chalabi and Gerges, 2012; Al-Badrashiny et al., 2014). These approaches usually contain the following steps. First, the text in the source language is converted into the sounds in the source language. Then, the sounds are modified to fit the sound inventory of the target language. Finally, the sounds are transformed into the target language. Recently, many studies applied end-to-end neural approaches to transliteration (Finch et al., 2016; Guellil et al., 2017; Hadj Ameur et al., 2017; Kundu et al., 2018; Grundkiewicz and Heafield, 2018; Le et al., 2019; Moran and Lignos, 2020) and achieved good performance. Compared with conventional approaches, the neural approaches provide a one-step solution for transliteration and do not require manually created rules. To train a well-performing neural model, particularly, for the transliteration from English and Arabic, several datasets are created (Kumaran et al., 2010; Chen and Skiena, 2016; Merhav and Ash, 2018; Chen et al., 2018). However, most existing studies mainly apply standard sequence-to-sequence approaches to English-Arabic transliteration, without leveraging the phonemic information between the two languages. In addition, the datasets used in existing studies are constructed from linked Wikipedia entries in different languages with limited attention paid to other resources.

Compared with previous studies, the name pairs in *EANames* are collected online sources rather than Wikipedia entries and the proposed neural approach for English-Arabic transliteration uses a memory-based module to leverage the language specific phonemic information.

6 Conclusions

In this paper, we collect a corpus named *EANames* for English-Arabic transliteration, where the names pairs are collected from online resources rather than Wikipedia entries. Based on the real data from *EANames*, we propose a neural transliteration model enhanced by memories to take advantage of phonemic information from English and Arabic. Specifically, in the memory module, the phonemic symbols associated with each input English letter are weighed and leveraged discriminatively to guide the transliteration process from English to Arabic. The experimental results and analysis on *EANames* demonstrate the effectiveness of our approach, which outperforms strong baselines with respect to all widely used evaluation metrics.

References

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*, pages 30–38, Ann Arbor, Michigan.

Yaser Al-Onaizan and Kevin Knight. 2002. Machine Transliteration of Names in Arabic Texts. In *Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages*, Philadelphia, Pennsylvania, USA.

Manar Alkhatib and Khaled Shaalan. 2020. Boosting Arabic Named Entity Recognition Transliteration with Deep Learning. In *The thirty-third international flairs conference*.

Faisal Alshuwaier and Ali Areshey. 2011. Translating English Names to Arabic Using Phonotactic Rules. In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation*, pages 485–492, Singapore.

M. Arbabi, S. M. Fischthal, V. C. Cheng, and E. Bart. 1994. Algorithms for Arabic Name Transliteration. *IBM Journal of Research and Development*, 38(2):183–194.

Fernando Benites, Gilbert François Duivesteijn, Pius von Däniken, and Mark Cieliebak. 2020. Translit: A Large-scale Name Transliteration Resource. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 3265–3271.

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.

Guimin Chen, Yuanhe Tian, Yan Song, and Xiang Wan. 2021. Relation Extraction with Type-aware Map Memories of Word Dependencies. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2501–2512, Online.

Nancy Chen, Rafael E. Banchs, Min Zhang, Xiangyu Duan, and Haizhou Li. 2018. Report of NEWS 2018 Named Entity Transliteration Shared Task. In *Proceedings of the Seventh Named Entities Workshop*, pages 55–73, Melbourne, Australia.

Yanqing Chen and Steven Skiena. 2016. False-friend Detection and Entity Matching via Unsupervised Transliteration. *arXiv preprint arXiv:1611.06722*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. 2020. ZEN: Pre-training Chinese Text Encoder Enhanced by N-gram Representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4729–4740.

Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. 2021. Taming Pre-trained Language Models with N-gram Representations for Low-Resource Domain Adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3336–3349, Online.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-Bidirectional Neural Models for Machine Transliteration. In *Proceedings of the Sixth Named Entity Workshop*, pages 78–82, Berlin, Germany.

Roman Grundkiewicz and Kenneth Heafield. 2018. Neural Machine Translation Techniques for Named Entity Transliteration. In *Proceedings of the Seventh Named Entities Workshop*, pages 89–94, Melbourne, Australia.

Imane Guellil, Faiçal Azouaou, Mourad Abbas, and Sadat Fatiha. 2017. Arabizi Transliteration of Algerian Arabic Dialect into Modern Standard Arabic. *Social MT*, pages 1–8.

Nizar Habash, Abdelhadi Soudi, and Timothy Buckwalter. 2007. *On Arabic Transliteration*, pages 15–22. Dordrecht.

Mohamed Seghir Hadj Ameur, Farid Meziane, and Ahmed Guessoum. 2017. Arabic Machine Transliteration using an Attention-based Encoder-decoder Model. *Procedia Computer Science*, 117:287–297. Arabic Computational Linguistics.

Jialong Han, Yan Song, Wayne Xin Zhao, Shuming Shi, and Haisong Zhang. 2018. Hyperdoc2vec: Distributed Representations of Hypertext Documents. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2384–2394, Melbourne, Australia.

Parag Jain and Mirella Lapata. 2021. Memory-Based Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 9:1197–1212.

Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. *Computational Linguistics*, 24(4):599–612.

Pankaj Kumar and V Kumar. 2013. Statistical Machine Translation based Punjabi to English Transliteration System for Proper Nouns. *International Journal of Application or Innovation in Engineering & Management*, 2(8):318–321.

A Kumaran, Mitesh M. Khapra, and Haizhou Li. 2010. Report of NEWS 2010 Transliteration Mining Shared Task. In *Proceedings of the 2010 Named Entities Workshop*, pages 21–28, Uppsala, Sweden.

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.

Ngoc Tan Le, Fatiha Sadat, Lucie Menard, and Dien Dinh. 2019. Low-Resource Machine Transliteration Using Recurrent Neural Networks. 18(2).

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online.

Haizhou Li, Min Zhang, and Jian Su. 2004. A Joint Source-Channel Model for Machine Transliteration. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 159–166, Barcelona, Spain.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Yuval Merhav and Stephen Ash. 2018. Design Challenges in Named Entity Transliteration. In *Proceedings of the 27th international conference on computational linguistics*, pages 630–640.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409.

Molly Moran and Constantine Lignos. 2020. Effective Architectures for Low Resource Multilingual Named Entity Transliteration. In *Proceedings of the 3rd Workshop on Technologies for MT of Low Resource Languages*, pages 79–86, Suzhou, China.

Nikitha Murikinati, Antonios Anastasopoulos, and Graham Neubig. 2020. Transliteration for Cross-lingual Morphological Inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Yuyang Nie, Yuanhe Tian, Yan Song, Xiang Ao, and Xiang Wan. 2020. Improving Named Entity Recognition with Attentive Ensemble of Syntactic Information. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4231–4245.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006. A Machine Transliteration Model Based on Correspondence between Graphemes and Phonemes. *ACM Transactions on Asian Language Information Processing*, 5(3):185–208.

Vladimir Pervouchine, Haizhou Li, and Bo Lin. 2009. Transliteration Alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 136–144, Suntec, Singapore.

Han Qin, Guimin Chen, Yuanhe Tian, and Yan Song. 2021. Improving Arabic Diacritization with Regularized Decoding and Adversarial Training. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-text Transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Sujith Ravi and Kevin Knight. 2009. Learning Phoneme Mappings for Transliteration without Parallel Data. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 37–45, Boulder, Colorado.

Mihaela Rosca and Thomas Breuel. 2016. Sequence-to-sequence Neural Network Models for Transliteration. *arXiv preprint arXiv:1610.09565*.

Yan Song and Chunyu Kit. 2010. Does Joint Decoding Really Outperform Cascade Processing in English-to-Chinese Transliteration Generation? The Role of Syllabification. In *2010 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3323–3328.

Yan Song, Chunyu Kit, and Xiao Chen. 2009. Transliteration of Name Entity via Improved Statistical Translation on Character Sequences. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 57–60, Suntec, Singapore.

Yan Song, Tong Zhang, Yonggang Wang, and Kai-Fu Lee. 2021. ZEN 2.0: Continue Training and Adaption for N-gram Enhanced Text Encoders. *arXiv preprint arXiv:2105.01279*.

Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Learning to Repair: Repairing Model Output Errors after Deployment Using a Dynamic Memory of Feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States.

Yuanhe Tian, Guimin Chen, and Yan Song. 2021. Enhancing Aspect-level Sentiment Analysis with Word Dependencies. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3726–3739, Online.

Yuanhe Tian, Han Qin, Fei Xia, and Yan Song. 2022. Syntax-driven Approach for Semantic Role Labeling. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7129–7139.

Yuanhe Tian, Wang Shen, Yan Song, Fei Xia, Min He, and Kenli Li. 2020a. Improving Biomedical Named Entity Recognition with Syntactic Information. *BMC Bioinformatics*, 21:1471–2105.

Yuanhe Tian, Yan Song, and Fei Xia. 2020b. Joint Chinese Word Segmentation and Part-of-speech Tagging via Multi-channel Attention of Character N-grams. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2073–2084.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. 2018. Bootstrapping Transliteration with Constrained Discovery for Low-resource Languages. *arXiv preprint arXiv:1809.07807*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. *Advances in neural information processing systems*, 30.