English-to-Chinese Transliteration with a Phonetic Auxiliary Task

Yuan He[*]
Department of Computer Science
University of Oxford
yuan.he@cs.ox.ac.uk

Shay B. Cohen
School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

Abstract

Approaching named entities transliteration as a Neural Machine Translation (NMT) problem is common practice. While many have applied various NMT techniques to enhance machine transliteration models, few focus on the linguistic features particular to the relevant languages. In this paper, we investigate the effect of incorporating phonetic features for English-to-Chinese transliteration under the multi-task learning (MTL) setting—where we define a phonetic auxiliary task aimed to improve the generalization performance of the main transliteration task. In addition to our system, we also release a new English-to-Chinese dataset and propose a novel evaluation metric which considers multiple possible transliterations given a source name. Our results show that the multi-task model achieves similar performance as the previous state of the art with a model of a much smaller size.[1]

1 Introduction

Transliteration, the act of mapping a name from the orthographic system of one language to another, is directed by the pronunciation in the source and target languages, and often by historical reasons or conventions. It plays an important role in tasks like information retrieval and machine translation (Marton and Zitouni, 2014; Hermjakob et al., 2008).

Over the recent years, many have addressed transliteration using sequence-to-sequence (seq2seq) deep learning models (Rosca and Breuel, 2016; Merhav and Ash, 2018; Grundkiewicz and Heafield, 2018), enhanced with several NMT techniques (Grundkiewicz and Heafield, 2018). However, this recent work neglects the most crucial feature for transliteration, i.e. pronunciation. To bridge this gap, we define a phonetic auxiliary task that shares the sound information with the main transliteration task under the multi-task learning (MTL) setting.

English	IPA	Chinese	Pinyin
A	/ˈeɪ./	艾	ài
my	/mi/	米	mǐ

Table 1: An example of English-to-Chinese transliteration, from *Amy* to 艾米. Each row presents a group of corresponding subsequences in different representations.

Depending on the specific language, the written form of a word reveals its pronunciation to various extents. For alphabetical languages such as English and French, a letter, or a sequence of letters, usually reflects the word pronunciation. For example, the word *Amy* (in the International Phonetic Alphabet, IPA, /ˈeɪ.mi/) has the sub-word *A* corresponding to /ˈeɪ./ and *my* corresponding to /mi/. In contrast, characters in a logographic[2] writing system for languages like Chinese or Japanese do not explicitly indicate sound (Xing et al., 2006).

In this paper, we give a treatment to the problem of transliteration from English (alphabet) to Chinese[3] (logogram) using an RNN-based MTL model with a phonetic auxiliary task. We transform each Chinese character to the alphabetical representation of its pronunciation via the official phonetic writing system, Pinyin,[4] which uses Latin letters with four diacritics denoting tones to represent the sounds.

[*]Work done at The University of Edinburgh.

[1]Our code and data are available at https://github.com/Lawhy/Multi-task-NMTransliteration.

[2]A logogram is an individual character that represents a whole word or phrase.

[3]The Chinese language we mention in this paper refers explicitly to Mandarin, which is the official language originated from the northern dialect in China.

[4]Pinyin is the official romanization system for Standard Chinese (Mandarin) in mainland China and to some extent in Taiwan. It does not apply to other Chinese dialects.

For example, the Chinese transliteration for *Amy* is 艾米 and the associated Pinyin representation is *ài mǐ*. We summarize the correspondences occurring in this example in Table 1.

Due to the similarity between the source name and the Pinyin representation, Jiang et al. (2009) proposed a sequential transliteration model that uses Pinyin as an intermediate representation before transliterating a Chinese name to English. In contrast, our idea is to build a model with a shared encoder and dual decoders, that can learn the mapping from English to Chinese and Pinyin simultaneously. By jointly learning source-to-target and source-to-sound mappings, the encoder is expected to generalize better (Ruder, 2017) and pass more refined information to the decoders.

Transliteration datasets are often extracted from dictionaries, or aligned corpus generated from applying named entity recognition (NER) system to parallel newspaper articles in different languages (Sproat et al., 2006). We use two datasets for our experiments, one taken from NEWS Machine Transliteration Shared Task (Chen et al., 2018) and the other extracted from a large dictionary. We evaluate the transliteration system using both the conventional word accuracy and a novel metric designed for English-to-Chinese transliteration (see Section 5). Our contributions are as follows:

1. We make available a new English-to-Chinese named entities dataset ("DICT") particular to names of people. This dataset is based on the dictionary *A Comprehensive Dictionary of Names in Roman-Chinese* (Xinhua News Agency, 2007).
2. We propose a substitution-based metric called Accuracy with Alternating Character Table (ACC-ACT), which gives a better estimation of the system's quality than the traditional word accuracy (ACC).
3. We propose a multi-task learning transliteration model with a phonetic auxiliary task, and run experiments to demonstrate that it attains better scores than single-main-task or single-auxiliary-task models.

We report accuracy and F-score of 0.299 and 0.6799, respectively, on the NEWS dataset, with a model of size 22M parameters, compared to the previous state of the art (Grundkiewicz and Heafield, 2018), which achieves accuracy and F-score of 0.304 and 0.6791, respectively, with a model of size 133M parameters. On the DICT dataset, for the same model sizes, we report accuracy of 0.729 as compared to their 0.732.

Source ($\mathbf{x}$)	**Target** ($\mathbf{y}$)	**Pinyin** ($\mathbf{p}$)
Caleigh	凯莉	kai li

Table 2: An example data point under our multi-task learning setting.

2 Problem Formulation

We use the word *vocabulary* to describe the *set of characters* for the purpose of our task specification. Let V_{src} and V_{tgt} denote the source and target vocabularies, respectively. For a source word $\mathbf{x}$ of length I and a target word $\mathbf{y}$ of length J, we have:

$$\mathbf{x} = (x_1, x_2, ..., x_I) \in V_{src}^I,$$
$$\mathbf{y} = (y_1, y_2, ..., y_J) \in V_{tgt}^J.$$

where the kth element in the vector denotes a character at position k.

We formulate the task of transliteration as a supervised learning problem: given a collection of n training examples, $\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=0}^n$, the objective is to learn a predictor function, $f : \mathbf{x} \rightarrow \mathbf{y}$, of which the parameter space maximizes the following conditional probability:

$$P(\mathbf{y}|\mathbf{x}) \overset{\text{Chain Rule}}{=} \prod_{j=1}^{J} P(y_j|y_1, ..., y_{j-1}, \mathbf{x}).$$

For our multi-task transliteration model, the predictor becomes $f_{\text{MTL}} : \mathbf{x} \rightarrow (\mathbf{y}, \mathbf{p})$, where $\mathbf{p}$ denotes the written representation of the pronunciation of the target word $\mathbf{y}$. For decoding, we maximize the conditional probabilities, $P(\mathbf{p}|\mathbf{x}, \tilde{\mathbf{y}})$ and $P(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{p}})$, where $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{p}}$ refers to the implicit information channeled by one task to the other.

The phonetic information we use for our task refers to the Pinyin version of the name in Chinese, without tone marks,[5] because they are often removed for spelling Chinese names in an alphabetical language. We present an example data point in the form of $(\mathbf{x}, \mathbf{y}, \mathbf{p})$ in Table 2.

3 Dataset Preparation

We experiment with two different English-to-Chinese datasets. For simplicity, we denote the one

[5]For example, the Pinyins, *chī, chí, chǐ* and *chì*, are all transformed to *chi*. Note that this process will decrease the vocabulary size.

taken from NEWS Machine Transliteration Shared Task (Chen et al., 2018) as "NEWS," and the one extracted from the dictionary (Xinhua News Agency, 2007) as "DICT."

3.1 NEWS Dataset

We use the preprocessing script[6] created by Grundkiewicz and Heafield (2018) to construct the NEWS dataset from raw data provided in the Shared Task (Chen et al., 2018). This script merges the raw English-to-Chinese and Chinese-to-English datasets into a single one, then transforms it to uppercase[7] and tokenizes all names into sequences of characters (words are treated as sentences, characters are treated as words). In addition, it takes 513 examples from the training data to form the internal development set and uses the official development set as the internal test set.

To make the final comparison, we download the source-side data of the official test set from the Shared Task's website,[8] and submit the transliteration results (see Section 6.4).

3.2 DICT Dataset

The source dictionary contains approximately 680K name pairs for transliteration from other languages than Chinese. We extracted 58,456 pairs that originated in English and performed the following preprocessing steps:

1. For the source side (English), we remove the inverted commas and white spaces from names that contain them (e.g. *A'Court*, *Le Gresley*).
2. For both sides, we lowercase[9] all the words and tokenize them into sequences of characters.
3. Name pairs with multiple target transliterations are removed from the dataset and saved in a separate file for the construction of the ACT (see next paragraph). As such, every name pair becomes unique in our preprocessed dataset. We randomly divide the rest into the ratio of $8:1:1$, to form training, development and test sets.

We report the final partitions of both datasets in Table 3.

[6]Available at https://github.com/snukky/news-translit-nmt.

[7]We lowercase all the words in both NEWS and DICT datasets as evaluating transliteration is case-insensitive.

[8]The official test set with task ID T-EnCh is available at: http://workshop.colips.org/news2018/dataset.html.

[9]Lowercasing does not affect Chinese characters as they are not alphabetical.

Source	Train	Dev	Test
NEWS	81,252	513	1,000
DICT	46,620	5,828	5,828

Table 3: Numbers of data points in training, development and test sets of NEWS and DICT datasets. Dev and Test for the NEWS dataset (first row) refer to the internal development and test set, respectively.

3.3 Alternating Character Table

Chinese characters[10] that sound alike can often replace each other in the transliteration of a name from other languages. Unlike an alphabetical language where a similar pronunciation is bounded to sub-words of various lengths, characters in Chinese have concrete and independent pronunciations. Thus, we can conveniently build the Alternating Character Table (ACT) with each row storing a list of interchangeable characters.

We construct the ACT based on the DICT dataset because it contains less noise after applying significant data cleansing. In total, 449 English names from the DICT dataset have more than one transliterations in Chinese. We purposely removed all these names from the DICT data during the preprocessing so as to ensure that we are not using any knowledge from the test set. The final ACT contains 29 rows (see Appendix) and we use it with our adaptive evaluation metric (see Section 5).

3.4 Pinyin Conversion

In transliteration, the pronunciations of the Chinese characters are often unique (even for a polyphonic character, e.g. 什, that has more than one Pinyins, *shí* and *shén*, only *shí* is commonly used in transliteration). Therefore, we can directly transform each Chinese character into a unique Pinyin, thus forming the target data for the auxiliary task. The procedure is as follows: for each character y_t in the target name $\mathbf{y}$, we use the Python package `pypinyin`[11] to map y_t to the corresponding Pinyin (without the tone mark). The tool will generate the most frequently used Pinyin for each y_t based on dictionary data. We then apply further manual correction on the Pinyins because the most frequent Pinyin is not necessarily the one used in transliteration.

[10]Limited to the set of characters (with size ≈1K out of 80K) commonly used in transliteration.

[11]Available at: https://github.com/mozillazg/python-pinyin. We use the `lazy_pinyin` feature to generate Pinyins without tone marks.

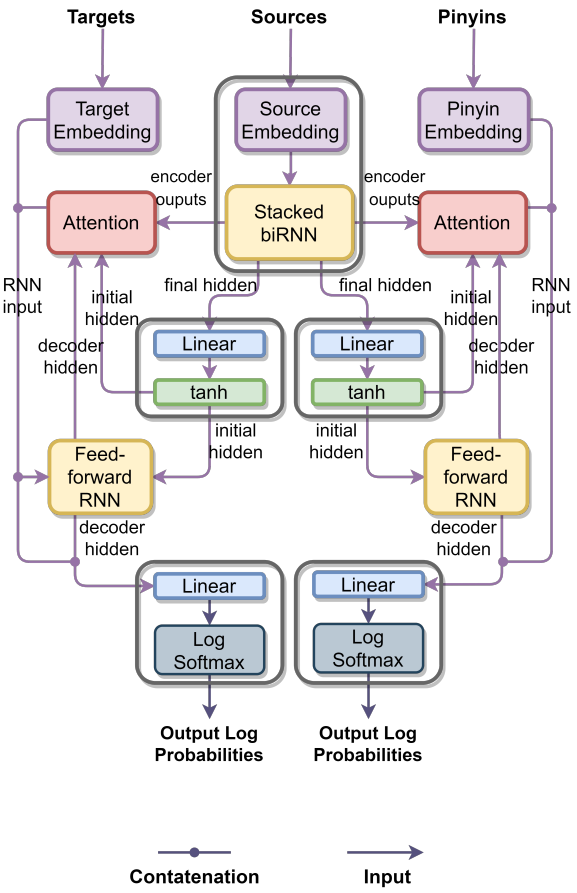

Figure 1: Visualization of the Seq2MultiSeq model. The left half illustrates the components involved in the main task and the right half is for the auxiliary task. The shared part is the encoder that consists of a source embedding layer and a stacked biRNN (top middle).

4 Model

Our model is intent on solving English-to-Chinese transliteration through joint supervised learning of source-to-target (main) and source-to-Pinyin (auxiliary) tasks. Training closely related tasks together can help the model to learn information that is often ignored in single-task learning, thus obtaining a better representation in the shared layers (in our case, encoder). Moreover, the auxiliary task implicitly provides the phonetic information that is not easily learned through the single main task given the characteristics of Chinese (see Section 1). Our model has a sequence-to-multiple-sequence (Seq2MultiSeq) architecture that contains a shared encoder and dual decoders. Between the encoder and decoder is a bridge layer[12] that transforms the encoder's final state into the decoder's initial state (see Figure 1).

The encoder has an embedding layer with dropout (Hinton et al., 2012), followed by a 2-layer biLSTM (Schuster and Paliwal, 1997). The bridge layer consists of a linear layer followed by $\tanh$ activation. The shared encoder passes its final state to the main-task decoder and the auxiliary-task decoder via separate bridge layers. In each decoder, we use additive attention (Bahdanau et al., 2015) to compute the context vector (weighted sum of the encoder outputs according to the attention scores), then concatenate it with the target embedding to form the input of the subsequent 2-layer feed-forward LSTM. The prediction is made by feeding the concatenation of the LSTM's output, the context vector and the target embedding into a linear layer followed by log-softmax.

Our model is expected to simultaneously maximize the conditional probabilities mentioned in Section 2. To achieve this goal, we use the linear combination of the main-task decoder's loss[13] (negative log likelihood; $l_{\mathbf{y}}$) and the auxiliary-task decoder's loss ($l_{\mathbf{p}}$) as the model's objective function:

$$l_{\text{MTL}} = \lambda \cdot l_{\mathbf{y}} + (1 - \lambda) \cdot l_{\mathbf{p}},$$

where the subscript MTL stands for multi-task learning and $0 < \lambda < 1$. Note that for $\lambda = 0$ and $\lambda = 1$, it is equivalent to train on a single auxiliary task and a single main task, respectively. The whole system is implemented using the deep learning framework PyTorch (Paszke et al., 2019).[14]

5 Adaptive Evaluation Metrics

We evaluate the transliteration system using word accuracy (ACC) and its variants on the 1-best output:

$$\text{ACC} = \frac{1}{N} \sum_{(y,\hat{y})} \mathbb{I}_{\text{criterion}(\hat{y},y)},$$

where N is the total number of test-set samples, $\mathbb{I}_{\text{criterion}(\hat{y},y)}$ is an indicator function with value 1 if the prediction (top candidate) $\hat{y}$ matches the reference y under certain criterion. The simplest criterion is exact string match between $\hat{y}$ and y. If the test set contains multiple target words for a single source word, we let indicator be 1 if the prediction matches one of the references (Chen et al., 2018).

[12] We call it "bridge" because it connects the shared encoder to each decoder. It allows flexible choices of the hidden sizes of the encoder and decoder and serves as the intermediate "buffer" before passing the encoder final state to each decoder.

[13] We use `nn.NLLLoss()` from the PyTorch library.

[14] Available at `https://pytorch.org/`.

Source	Target (F)	Target (M)	MED
Mona	莫娜	莫纳	1
Colina	科莉娜	科利纳	2

Table 4: Examples of a single source name with more than one target transliterations, with (F) and (M) indicating female and male, respectively.

We use ACC and ACC+ to denote the original accuracy and its variant with multiple references.

The drawback of ACC is that it may underestimate the quality of the system because it neglects the possibility of having more than one transliteration for a given source name, as is the case for English-to-Chinese transliteration. For example in Table 4, if the test set only includes Target (F) for a Source while the model predicts Target (M), ACC will mistakenly count it as wrong. Although ACC+ considers the alternatives appearing in the dataset, it is unrealistic to expect the dataset to contain all possible references. To resolve this issue, we propose a new variant of word accuracy specific to English-to-Chinese transliteration.

Based on the knowledge of a native Chinese speaker, we analyze the English-to-Chinese dataset and summarize the key observations for source names with multiple target transliterations as follows: the minimum edit distance (MED) between any two target names ≤ 2, and the lengths are the same; for any two such target names, distinct characters occur in the same position, and they often indicate the *gender* of the name (see Table 4).

To use ACT in accord with the above observations, we propose the following criterion for the accuracy indicator function (we refer to it as ACC-ACT). Let subscript t denote the position of a character, then $\mathbb{I}_{\text{criterion}(\hat{y},y)} = 1$ if either $\text{MED}(\hat{y}, y) = 0$ (which covers all the cases for ACC) or the following conditions are met **in order**:

1. $\hat{y}$ and y are of the same length, L;
2. $\text{MED}(\hat{y}, y) \leq 2$ and distinct characters of $\hat{y}$ and y must occur in the same position(s);
3. If $\hat{y}_t \neq y_t$ for $1 \leq t \leq L$, replace $\hat{y}_t$ by looking up the ACT and this condition will be satisfied if any of the modified $\hat{y}$(s) can match y exactly.

There is no guarantee that characters that are interchangeable according to ACT can replace each other in every scenario. But since we only apply substitution on the output predictions rather than the references, we are not manipulating the test set by creating any new instance. This new metric (ACC-ACT) will ensure cases like in Table 4 are captured without requiring extra data in the test set, thus giving a more reasonable estimate of the system's quality than both ACC and ACC+.

		Enc	Dec-M	Dec-A
Emb.	h	256	256	128
	δ	0.1	0.1	0.1
RNN	h	512	512	128
	δ	0.2	0.2	0.1

Table 5: Illustration of the model settings, where Emb. and RNN stand for the embedding layers and RNN units in each part (column) of the model, h and δ are the hidden size and dropout value, respectively. The column names (from left to right) stand for encoder, main-task decoder and auxiliary-task decoder.

6 Experimental Setup

Recall from Section 4 that we use λ to denote the weighting of the two tasks we train. We set the single-main-task ($\lambda = 1$) and the single-auxiliary-task ($\lambda = 0$) models as the baselines, and compare the multi-task models of different weightings ($\lambda \in \{\frac{1}{6}, \frac{1}{4}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, \frac{8}{9}\}$) against them. We conduct experiments on both the NEWS and DICT datasets and select the best model for each of them to compare to the previous state of the art.

6.1 Model and Training Settings

The configurations of hidden sizes and dropout values of embedding layers and RNN units are presented in Table 5. The type of all RNN units is LSTM and the number of layers is set to 2. Besides the bridge layer that transforms the encoder's final hidden state to the decoder's initial hidden state, we add another one to carry the final cell state for using LSTM (in total, we have 4 "bridges").

We use the Adam optimizer (Kingma and Ba, 2015) with the batch size set to 64. Evaluation of the development set is carried out on every 500 batches. We record the validation score (ACC) and decrease the learning rate (initially set to 0.003) by 90% if the score does not surpass the previous best. We pick the final model that attains the highest validation score within 100 training epochs.

For decoding in the training phase, we apply teacher forcing (Williams and Zipser, 1989) with

	NEWS				DICT		
	Main			Auxiliary	Main		Auxiliary
λ	ACC	ACC+	ACC-ACT	ACC	ACC	ACC-ACT	ACC
1	0.723	0.731	0.746	NA	0.725	0.748	NA
1/6	0.666	0.672	0.688	0.698	0.728	0.750	0.744
1/4	0.734	0.743	0.751	0.755	0.725	0.747	**0.746**
1/2	0.724	0.733	0.740	0.738	0.723	0.748	0.739
2/3	0.698	0.707	0.715	0.705	0.722	0.746	0.739
5/6	**0.739**	**0.749**	**0.760**	**0.757**	**0.729**	**0.752**	**0.746**
8/9	0.670	0.679	0.686	0.705	0.722	0.746	0.734
0	NA	NA	NA	0.743	NA	NA	0.743

Table 6: Experiment results on NEWS internal test set and DICT development set, where $\lambda = 1$ and $\lambda = 0$ are baselines of main task and auxiliary task, respectively. Maximum score in each metric is is bold.

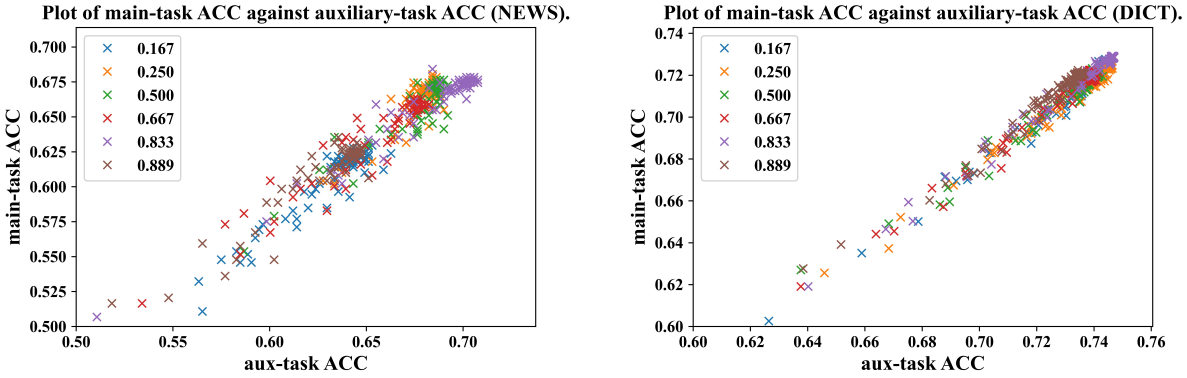

Figure 2: The plots of main-task ACC against auxiliary-task ACC on the NEWS (left) and DICT (right) development sets. Colors indicate which multi-task model (by λ value) the evaluation points belong to. To highlight the dense regions, we set the minimum of the x-axis to 0.5 and 0.6 for NEWS and DICT datasets, respectively.

the following empirical decay function:

$$\mathbf{tfr} = \max\left(1 - \frac{10 + \mathbf{epoch} \times 1.5}{50}, 0.2\right),$$

where $\mathbf{tfr}$ refers to the teacher forcing ratio, i.e. the probability of feeding the true reference instead of the predicted token. We use beam search decoding with beam size 10 and length normalization (Wu et al., 2016) for evaluation.

6.2 Evaluation

We use ACC and ACC-ACT to evaluate the performance on the main task and ACC on the auxiliary task. Note that since the only data portion we have that contains multiple references given a source word is the internal test set of NEWS data, we apply ACC+ on this particular set exclusively.

6.3 Model Selection

In the experiments in this section, we tune λ on the NEWS internal test set and DICT development set, and select the model with the highest ACC on the main task.

The experiment results in Table 6 show that $\lambda = \frac{5}{6}$ yields the best models on both datasets. We observe a significant improvement against the baselines on NEWS while a less noticeable increase on DICT. Besides, the models are more sensitive to λ on NEWS than DICT (with standard deviation 0.03 and 0.003 on ACC, respectively).

Furthermore, we investigate the relationship between the main and the auxiliary tasks based on the evaluation points of the development set. In Figure 2, we observe a nearly-total positive linear correlation between the main-task ACC and auxiliary-task ACC, and this is further evident in the Pearson cor-

	Internal Test				Official Test
	Main			Auxiliary	Main
System	ACC	ACC+	ACC-ACT	ACC	ACC+
Baseline	0.724	0.733	0.742	0.736	NA
Multi-task	0.739	0.749	0.760	0.757	0.299
BiDeep	0.731	0.739	0.746	0.740	NA
BiDeep+	NA	0.765	NA	NA	0.304

Table 7: Experiment results on the NEWS internal test (official development) set and official test set, where "Baseline" refers to the single-task model and "BiDeep+" refers to the best system Grundkiewicz and Heafield (2018) submitted to the NEWS workshop, and the corresponding scores are taken from their paper.

	Main		Auxiliary
System	ACC	ACC-ACT	ACC
Baseline	0.726	0.748	0.738
Multi-task	0.729	0.751	0.749
BiDeep	0.732	0.755	0.760

Table 8: Experiment results on the DICT test set, where Baseline refers to the single-task model.

User	**ACC+**	**F-score**
romang	0.3040 (1)	0.6791 (2)
Ours	0.2990 (2)	0.6799 (1)
saeednajafi	0.2820 (3)	0.6680 (3)
soumyadeep	0.2610 (4)	0.6603 (4)

Table 9: Table of the NEWS leaderboard (available at https://competitions.codalab.org/competitions/18905#results, accessed 19 June 2020). User "romang" refers to Grundkiewicz and Heafield (2018).

relation coefficients[15], which are 0.982 and 0.992 for NEWS and DICT, respectively. This means the multi-task model improves the performance on both tasks simultaneously.

6.4 Test-set Results and System Comparison

We submit our 1-best transliteration results on the NEWS official test set through the CodaLab link provided by the Shared Task's Committee and we present the leaderboard partially in Table 9. Note that in addition to ACC+, the leaderboard also records mean F-score[16] on which we rank first.

We report the test-set performance of our best multi-task model on NEWS in Table 7 and DICT in Table 8, in comparison to the system built by Grundkiewicz and Heafield (2018). The baseline model of their work employs the RNN-based BiDeep[17] architecture (Miceli Barone et al., 2017) which consists of 4 bidirectional alternating stacked encoder, each with a 2-layer transition RNN cell, and 4 stacked decoders with base RNN of depth 2 and higher RNN of depth 4 (Zhou et al., 2016; Pascanu et al., 2014; Wu et al., 2016). Besides, they strengthen the model by applying layer normalization (Ba et al., 2016), skip connections (Zhang et al., 2016) and parameter tying (Press and Wolf, 2017). We reproduce their model without changing any configurations in their paper (Grundkiewicz and Heafield, 2018), and train it on both tasks separately.

In Table 7, we can see that the multi-task model performs significantly better than both the single-task baseline and the BiDeep model in all metrics on NEWS. Note that the BiDeep model we reproduce achieves the same ACC+ as reported in the work of Grundkiewicz and Heafield (2018) and ACC+ is the only evaluation metric used in their paper. "BiDeep+" in the third row refers to the final system they submitted to the Shared Task, on which they adopted additional NMT techniques including ensemble modeling for re-ranking and synthetic data generated from back translation (Sennrich et al., 2017). Our ACC+ score on

[15]Computed by `pearsonr()` from `Scipy` library, which is available at: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.pearsonr.html.

[16]The F-score metric measures the similarity between the target prediction and reference. Precision and Recall in this particular F-score are computed based on the length of the Longest Common Subsequence. See details in the NEWS whitepaper (Chen et al., 2018).

[17]Implemented with the Marian toolkit available at https://marian-nmt.github.io/docs/.

Source	Output (ST)	Output (MT)
ocallaghan	奥卡拉根	奥卡拉汉 ✓
holleran	霍尔伦	霍勒伦 ✓
ajemian	阿赫米安	阿杰米安

Table 10: Example outputs and the corresponding source words of our systems, where "ST" and "MT" refer to "single-task" and "multi-task" models. The tick symbols indicate which outputs match the references.

the anonymized official test set is 0.299 which is slightly worse than their 0.304. However, we attain a better F-score (0.6799) than them (0.6791) as shown in Table 9. Moreover, our model is of size 22M parameters, which is much smaller than their baseline BiDeep of size 133M parameters,[18] and we do not apply as many NMT techniques as they did. Nevertheless, on the DICT test set, there is no prominent difference among the single-task baseline, multi-task and BiDeep model, possibly because the noise pattern in the DICT dataset is not complex enough to reflect the learning ability of these models.

7 Discussion

In our experiments, a system has ACC-ACT>ACC+>ACC because both ACC-ACT and ACC+ consider the cases of ACC but ACC-ACT can capture more acceptable transliterations. Despite a consistent ranking given by the three metrics, ACC-ACT reveals different information from ACC and ACC+. For example, in Table 6, the model of $\lambda = \frac{5}{6}$ outperforms $\lambda = \frac{1}{2}$ by 0.015 and 0.016 in ACC and ACC+, respectively, but the difference is 0.020 in ACC-ACT, on the NEWS dataset. This suggests a more prominent gap between these two models. In contrast, by looking at the same two rows but on the DICT dataset, ACC-ACT indicates a smaller gap (0.004) than ACC (0.006). If we conduct experiments on another dataset, the disagreement among the metrics might be significant enough to render an inconsistent ranking.

Furthermore, we present some typical examples in which the multi-task model generates better predictions than the single-task in Table 10. In the first example, the single-task model wrongly maps the sub-word *ghan* to 根 (emphasizing on the character *g*) while the multi-task model correctly maps *han* to 汉. The erroneous grouping of the English characters also occurs in the second example where the single-task model maps *er* to 尔 instead of more reasonably *ler* to 勒. Even in the third example where both outputs are mismatched, the multi-task model predicts the character 杰, which is closer to the source sub-word *je* than the single-task model's 赫 in terms of pronunciation. Overall, it seems that the multi-task model can capture the source-word pronunciation better than the single-task one.

Still, the multi-task model does not consistently handle all names better than the single-task model–especially for exceptional names that do not have a regular transliteration. For instance, the name *Fyleman* is transliterated into 法伊尔曼, but the character 伊 does not have any source-word correspondence if we consider the pronunciation of the source name.

Finally, our model can be generalized to other transliteration tasks by replacing Pinyin with other phonetic representations such as IPA for English and rōmaji for Japanese. In addition, ACC-ACT can be extended to alphabetical languages by, for instance, constructing the Alternating Sub-word Table which stores lists of interchangeable subsequences. Another possible future work is to redesign the objective function by treating λ as a trainable parameter or including the correlation information (Papasarantopoulos et al., 2019).

8 Related Work

Previous work has demonstrated the effectiveness of using MTL on models through joint learning of various NLP tasks such as machine translation, syntactic and dependency parsing (Luong et al., 2016; Dong et al., 2015; Li et al., 2014). In most of this work, underlies a similar idea to create a unified training setting for several tasks by sharing the core parameters. Besides, machine transliteration has a long history of using phonetic information, for example, by mapping a phrase to its pronunciation in the source language and then convert the sound to the target word (Knight and Graehl, 1997). There is also relevant work that uses both graphemes and phonemes to various extents for transliteration, such as the correspondence-based (Oh et al., 2006) and G2P-based (Le and Sadat, 2018) approaches. Our work is inspired by the intu-

[18] We compute the size of our multi-task model by counting the number of trainable parameters extracted from `model.parameters()`; For the BiDeep model, we use the `numpy` package to load the model in `.npz` format and calculate the number of parameters via a simple for-loop.

itive understanding that pronunciation is essential for transliteration, and the success of incorporating phonetic information such as Pinyin (Jiang et al., 2009) and IPA (Salam et al., 2011), in the model design.

9 Conclusion

We argue in this paper that language-specific features should be used when solving transliteration in a neural setting, and we exemplify a way of using phonetic information as the transferred knowledge to improve a neural machine transliteration system. Our results demonstrate that the main transliteration task and the auxiliary phonetic task are indeed mutually beneficial in English-to-Chinese transliteration, and we discuss the possibility of applying this idea on other language pairs.

Acknowledgements

We thank the anonymous reviewers for their insightful feedback. We would also like to thank Zheng Zhao, Zhijiang Guo, Waylon Li and Pinzhen Chen for their help and comments.

References

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *ArXiv*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Nancy Chen, Rafael E. Banchs, Xiangyu Duan, Min Zhang, and Haizhou Li, editors. 2018. *Proceedings of the Seventh Named Entities Workshop*. Association for Computational Linguistics, Melbourne, Australia.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Ulf Hermjakob, Kevin Knight, and Hal Daumé. 2008. Name translation in statistical machine translation - learning when to transliterate. In *ACL*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580.

Xue Jiang, Le Sun, and Dakun Zhang. 2009. A syllable-based name transliteration system. In *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration (NEWS 2009)*, pages 96–99, Suntec, Singapore. Association for Computational Linguistics.

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

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, page 128–135, USA. Association for Computational Linguistics.

Ngoc Tan Le and Fatiha Sadat. 2018. Low-resource machine transliteration using recurrent neural networks of Asian languages. In *Proceedings of the Seventh Named Entities Workshop*, pages 95–100, Melbourne, Australia. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, and Wenliang Chen. 2014. Joint optimization for chinese pos tagging and dependency parsing. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22:274–286.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. *CoRR*, abs/1511.06114.

Yuval Marton and Imed Zitouni. 2014. Transliteration normalization for information extraction and machine translation. *Journal of King Saud University - Computer and Information Sciences*, 26(4):379 – 387. Special Issue on Arabic NLP.

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, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 99–107, Copenhagen, Denmark. Association for Computational Linguistics.

Jong-Hoon Oh, Key-Sun Choi, and Hitoshi Isahara. 2006. A machine transliteration model based on correspondence between graphemes and phonemes. *ACM Trans. Asian Lang. Inf. Process.*, 5:185–208.

Nikos Papasarantopoulos, Lea Frermann, Mirella Lapata, and Shay B. Cohen. 2019. Partners in crime: Multi-view sequential inference for movie understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2057–2067, Hong Kong, China. Association for Computational Linguistics.

Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2014. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. *ArXiv*, abs/1608.05859.

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

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098.

Khan Md. Anwarus Salam, Yamada Setsuo, and Tetsuro Nishino. 2011. Translating unknown words using wordnet and ipa-based-transliteration. *14th International Conference on Computer and Information Technology (ICCIT 2011)*, pages 481–486.

Mike Schuster and Kuldip Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 73–80, Sydney, Australia. Association for Computational Linguistics.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Hongbing Xing, Hua Shu, and Ping Li. 2006. The acquisition of chinese characters : Corpus analyses and connectionist simulations.

Xinhua News Agency. 2007. *Names of the World's Peoples: a comprehensive dictionary of names in Roman-Chinese*. China Translation & Publishing Corporation.

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Ruslan Salakhutdinov, and Yoshua Bengio. 2016. Architectural complexity measures of recurrent neural networks. *ArXiv*, abs/1602.08210.

Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. 2016. Deep recurrent models with fast-forward connections for neural machine translation. *Transactions of the Association for Computational Linguistics*, 4:371–383.

A Alternating Character Table in Full

Alternating Characters
莉,利,里,丽
弗,夫,芙
思,斯,丝
妮,内,娜,纳,尼
萨,沙,莎
亚,娅
玛,马,穆
琳,林
芭,巴
茜,西,锡
萝,罗
滕,坦
莱,来,勒
代,黛,戴
瓦,沃,娃
吉,姬,基
雷,蕾
薇,维,威
鲁,卢,露
塔,特
尤,于
安,阿
菲,费
纽,努
范,文
蒙,莫
查,恰
保,葆
柯,科

Table 11: The Alternating Character Table in full.