Leveraging Orthographic Similarity for Multilingual Neural Transliteration

Anoop Kunchukuttan[1], **Mitesh Khapra**[2], **Gurneet Singh**[2], **Pushpak Bhattacharyya**[1]
{anoopk,pb}@cse.iitb.ac.in, {miteshk,garry}@cse.iitm.ac.in

[1]Department of Computer Science & Engineering
Indian Institute of Technology Bombay, Mumbai, India.

[2]Department of Computer Science & Engineering
Indian Institute of Technology Madras, Chennai, India.

Abstract

We address the task of joint training of transliteration models for multiple language pairs (*multilingual transliteration*). This is an instance of multitask learning, where individual tasks (language pairs) benefit from sharing knowledge with related tasks. We focus on transliteration involving related tasks *i.e.,* languages sharing writing systems and phonetic properties (*orthographically similar languages*). We propose a modified neural encoder-decoder model that maximizes parameter sharing across language pairs in order to effectively leverage orthographic similarity. We show that multilingual transliteration significantly outperforms bilingual transliteration in different scenarios (average increase of 58% across a variety of languages we experimented with). We also show that multilingual transliteration models can generalize well to languages/language pairs not encountered during training and hence perform well on the *zero-shot* transliteration task. We show that further improvements can be achieved by using phonetic feature input.

1 Introduction

Transliteration is a key building block for multilingual and cross-lingual NLP since it is essential for (i) handling of names in applications like machine translation (MT) and cross-lingual information retrieval (CLIR), and (ii) user-friendly input methods. The transliteration problem has been extensively studied in literature for a variety of language pairs (Karimi et al., 2011). Previous work has looked at the most natural setup - training on a single language pair. However, no prior work exists on jointly training multiple language pairs (referred to as **multilingual transliteration** henceforth).

Multilingual transliteration can be seen as an instance of *multi-task learning*, where training each language pair constitutes a task. Multi-task learning works best when the tasks are related to each other, so sharing of knowledge across tasks is beneficial. Thus, multilingual transliteration can be beneficial, if the languages involved are related. We identify such a natural and practically useful scenario: multilingual transliteration involving languages that are related on account of sharing writing systems and phonetic properties. We refer to such languages as **orthographically similar languages**.

We say that two languages are ***orthographically similar*** *if they have: (i) highly overlapping phoneme sets, (ii) mutually compatible orthographic systems, and (iii) similar grapheme to phoneme mappings.* For instance, Indo-Aryan languages largely share the same set of phonemes. They use different Indic scripts, but correspondences can be established between equivalent characters across scripts. For example, the Hindi (Devanagari script) character क (*ka*) maps to the Bengali ক (*ka*) which stands for the consonant sound (IPA: k). The grapheme to phoneme mapping is also consistent for equivalent characters. We can identify two major sources of orthographic similarity: (a) genetic relationship between languages (groups like Romance, Slavic, Indo-Aryan and Turkic languages) (b) prolonged contact between languages over a long period of time, *e.g.* convergence in phonological properties of the Indo-Aryan and Dravidian languages in the Indian subcontinent, most strikingly retroflex consonants (Subbārāo, 2012). Dravidian and Indo-Aryan languages use compatible Indic scripts. Another

Transactions of the Association for Computational Linguistics, vol. 6, pp. 303–316, 2018. Action Editor: Mona Diab.
Submission batch: 8/2017; Revision batch: 12/2017; Published 5/2018.

example is the Nigerian linguistic area comprising Niger-Congo languages like Yoruba, Fula, Igbo and Afro-Asiatic languages like Hausa (the most widely spoken language in Nigeria). Most languages use the Latin script (some use Ajani, a modified Arabic script).

In this work, we explore multilingual transliteration involving orthographically similar languages. To the best of our knowledge, ours is the first work to address the task of multilingual transliteration. We propose that transliteration involving orthographically similar languages is a scenario where multilingual training can be very beneficial. Since these languages share phonological properties, the transliteration tasks are clearly related. *We can utilize this relatedness by sharing the vocabulary across all related languages.* The grapheme-to-grapheme correspondences enable vocabulary sharing. It helps transfer knowledge across languages while training. For instance, if the network learns that the English character *l* maps to the Hindi character ल (*la*), it would also learn that *l* maps to the corresponding Kannada character ಲ (la). Data from both Kannada and Hindi datasets will reinforce the evidence for this mapping. A similar argument can be made when both the source and target languages are related. *The grapheme-grapheme correspondences arise from the underlying phoneme-phoneme correspondences. The consistent grapheme-phoneme mappings help establish the grapheme-grapheme correspondences.*

Due to the utilization of language relatedness, the benefits that are typically ascribed to multi-task learning (Caruana, 1997) may also apply to multilingual transliteration. *Since related languages share characters, it is possible to share representations across languages.* This may help to generalize transliteration models since joint training provides an inductive bias which prefers models that are better at transliterating multiple language pairs. The training may also benefit from *implicit data augmentation* since training data from multiple language pairs is available. From the perspective of a single language pair, data from other language pairs can be seen as additional (noisy) training data. This is particularly beneficial in low-resource scenarios.

Our work adds to the increasing body of work investigating multilingual training for various NLP tasks like POS tagging (Gillick et al., 2016), NER (Yang et al., 2016; Rudramurthy et al., 2016) and machine translation (Dong et al., 2015; Firat et al., 2016; Lee et al., 2017; Zoph et al., 2016; Johnson et al., 2017) with a view to learn models that generalize across languages and make effective use of scarce training data.

The following are the contributions of our work:
(1) We propose a compact neural encoder-decoder model for multilingual transliteration, that is designed to ensure *maximum sharing of parameters across languages* while providing room for learning language-specific parameters. This allows greater sharing of knowledge across language pairs by leveraging orthographic similarity. We empirically show that models with maximal parameter sharing are beneficial, without increasing the model size.
(2) We show that multilingual transliteration exhibits significant improvement in transliteration accuracy over bilingual transliteration in different scenarios (average improvement of 58%). Our results are backed by extensive experiments on 8 languages across 2 orthographically similar language groups.
(3) We perform an error analysis which suggests that representations learnt by the encoder in multilingual transliteration can reduce transliteration ambiguities. Multilingual transliteration also seems better at learning canonical transliterations instead of alternative, phonetically equivalent transliterations. These could explain the improved performance of multilingual transliteration.
(4) We explore the *zeroshot transliteration task* (*i.e.,* transliteration between languages/language pairs not seen during training) and show that our multilingual model can generalize well to unseen languages/language pairs. Notably, the zeroshot transliteration results mostly outperform the direct bilingual transliteration model.
(5) We have *richer phonetic information* at our disposal for some related languages. We propose a novel method to incorporate phonetic input in the model and show that it provides modest gains for multilingual transliteration.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 formalizes the multilingual transliteration task and describes our proposed solution. Sections 4 and 5 discuss the experimental setup, results and analysis. Section 6 dis-

cusses various zeroshot transliteration scenarios, our solutions and the results of experiments. Section 7 discusses incorporation of phonetic information for multilingual transliteration. Section 8 concludes the work and discusses future directions.

2 Related Work

General Transliteration Methods Previous work on transliteration has focused on the scenario of bilingual training. Until recently, the best-performing solutions were discriminative statistical transliteration methods based on phrase-based statistical machine translation (Bisani and Ney, 2008; Jiampojamarn et al., 2008; Jiampojamarn et al., 2009; Finch and Sumita, 2010). Recent work has explored bilingual neural transliteration using the standard neural encoder-decoder architecture (with attention mechanism) (Bahdanau et al., 2015) or its adaptions (Finch et al., 2015; Finch et al., 2016). Using target bidirectional LSTM with model ensembling, Finch et al. (2016) have outperformed the state-of-the-art phrase-based systems on the NEWS shared task datasets. On the other hand, we focus on multilingual transliteration with the encoder-decoder architecture or its adaptations. The two strands of work are obviously complimentary.

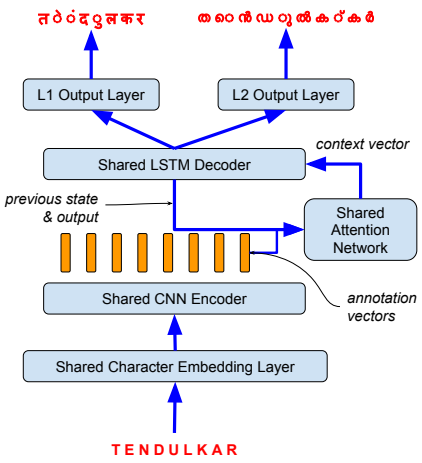
Multilinguality and Transliteration To the best of our knowledge, ours is the first work on multilingual transliteration. Jagarlamudi and Daumé III (2012) have proposed a method for *transliteration mining* (given a name and candidate transliterations, identify the correct transliteration) across multiple languages using grapheme to IPA mappings. Note that their model cannot generate transliterations; it can only rank candidates. Some literature mentions multilingual transliteration (Surana and Singh, 2008; He et al., 2017; Prakash, 2012; Pouliquen et al., 2005) or multilingual transliteration mining (Klementiev and Roth, 2006; Yoon et al., 2007). In these cases, however, *multilingual* refer to methods which work with multiple languages (as opposed to joint training - the sense of the word *multilingual* as we use it).

Multilingual Translation Our work on multilingual transliteration is motivated by recently proposed multilingual neural machine translation architectures (Firat et al., 2016). Broadly, these proposals can be categorized into two groups. One group consists of architectures that specialize parts of the network for particular languages: specialized encoders (Zoph et al., 2016), decoders (Dong et al., 2015) or both (Firat et al., 2016). The other group tries to learn more compact networks with little specialization across languages by using a joint vocabulary (Johnson et al., 2017; Lee et al., 2017). For multilingual transliteration, we adopt an approach that is closer to the latter group since the languages under consideration use compatible scripts resulting in a shared vocabulary. We specialize just the output layer for target languages, but share the encoder, decoder and character embeddings across languages. In this respect, we differ from Johnson et al. (2017). They share all network components across languages, but add an artificial token at the beginning of the input sequence to indicate the target language.

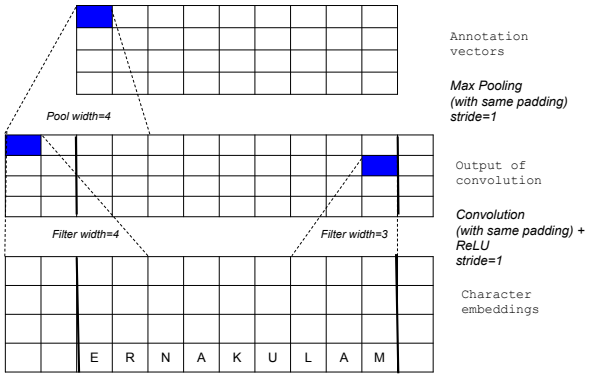
Zeroshot Transliteration We use the multilingual models to address zeroshot transliteration. *Zeroshot* transliteration using bridge/pivot language has been explored for statistical machine transliteration (Khapra et al., 2010) as well as neural machine transliteration (Saha et al., 2016). Unlike previous approaches which pivot over bilingual transliteration models, we propose zeroshot transliteration that pivots over multilingual transliteration models. We also propose a direct zeroshot transliteration method, a scenario which has been explored for machine translation by Johnson et al. (2017), but not investigated previously for transliteration. In our zeroshot model, sequences from multiple source languages are mapped to a common encoder representation without the need for a parallel corpus between the source languages. Another work, the correlational encoder-decoder architecture (Saha et al., 2016), maps source and pivot languages to a common space but requires a source-pivot parallel transliteration corpus.

3 Multilingual Transliteration Learning

We first formalize the multilingual transliteration task and then describe our proposed solution.



(a) Network Architecture



(b) CNN Encoder

Figure 1: Multilingual Neural Transliteration Architecture

3.1 Task Definition

The multilingual transliteration task involves learning transliteration models for l language pairs $(s_i, t_i) \in \mathbf{L}$ $(i = 1 \text{ to } l)$, where $\mathbf{L} \subset S \times T$, and S, T are sets of source and target languages respectively. The languages in each set are orthographically similar. S and T need not be mutually exclusive.

We are provided with parallel transliteration corpora for these l language pairs $(D_i, \ \forall i = 1 \text{ to } l)$. The goal is to learn a joint transliteration model for all language pairs which minimizes an appropriate loss function over all the transliteration corpora.

$$M^* = \arg\min_{M} \mathcal{L}(M, \mathcal{D}) \qquad (1)$$

where M is the candidate joint transliteration model and $\mathcal{D}$=$(D_1, D_2, ..., D_l)$ is training data for all language pairs, $\mathcal{L}$ is the training loss function given the model and the training data.

We focus on 3 practical training scenarios:

Similar source languages: We have multiple orthographically similar source languages and a single target language which is not similar to the source languages. This is an instance of many-to-one learning, *e.g.*, Indic languages to English.

Similar target languages: We have multiple orthographically similar target languages and a single source language which is not similar to the target languages. This is an instance of one-to-many learning, *e.g.*, English to Indic languages.

All similar languages: We have multiple source languages as well as target languages, which are all orthographically similar. This is an instance of many-to-many learning, *e.g.*, Indic-Indic languages.

3.2 Proposed Solution

We propose a neural encoder-decoder model for multilingual transliteration. For each source-target language pair (s, t), the network models $\mathbf{P^{s,t}} = p(y_j^t | y_{j-1}^t ... y_1^t, \mathbf{x}^s)$, where $\mathbf{x}^s$ is the input character sequence and y_j^t is j^{th} element of the output character sequence $\mathbf{y}^t$. Note that we design a single network to represent all the $\mathbf{P^{s,t}}$ distributions corresponding to the set of language pairs $\mathbf{L}$. Our network is an adaptation of the standard encoder-decoder model with attention (Bahdanau et al., 2015). We describe only the salient aspects of our network and refer the reader to Bahdanau et al. (2015) for the basic encoder-decoder architecture. Figure 1a shows the network architecture of our multilingual transliteration system.

Encoder & Decoder: We used a CNN encoder to encode the character sequence. It consists of a single convolutional layer (stride size = 1 and SAME padding), followed by ReLU units and max pooling. We use filters of different sizes and concatenate their output to produce the encoder output. Figure 1b shows a schematic of the encoder. We chose CNN over the conventional bidirectional

LSTM layer since the temporal dependencies for transliteration are mostly local, which can be handled by the CNN encoder. We observed that training and decoding are significantly faster, with little impact on accuracy. The decoder contains a layer of LSTM cells and their start state is the average of the encoder's output vectors (Sennrich et al., 2017).

Parameter Sharing: The vocabulary of the orthographically similar languages (at input and/or output) is comprised of the union of character sets of all these languages. Since the character set of these languages overlaps to a large extent, we share their character embeddings too. The encoder is shared across all source languages and the decoder is shared across all target languages.

The network uses a shared attention mechanism. The attention network is comprised of a single feedforward layer, which predicts the attention score given the previous decoder output, previous decoder state and encoder annotation vector.

The output layer (a fully connected feedforward layer) transforms the decoder LSTM layer's output to the size of the output vocabulary, and a softmax function is applied to convert the output scores to probabilities. Each target language has its own set of output layer parameters.

Barring the output layer, all network parameters (input embeddings, output embeddings, attention layer, encoder and decoder) are shared across all similar languages. This allows maximum transfer of information for multilingual learning, while the output layer alone specializes for the specific target language. Compared to using a target language tag in the input sequence (Johnson et al., 2017), we believe our approach allows the language-specific parameters to directly influence the output characters.

Training Objective and Model Selection: We minimize the average negative likelihood of parallel training corpora across all language pairs. We determined the hyperparameters which gave best results on a validation set. After training the model for a fixed number of iterations (sufficient for convergence), we select the model with the maximum accuracy on the validation set for each language pair. For instance, if the model corresponding to the 32^{nd} epoch reported maximum accuracy on the validation set for English-Hindi, this model was used for reporting test set results for English-Hindi. We observed that this criterion performed better than choosing the model with least validation set loss over all language pairs.

4 Experimental Setup

We describe our experimental setup.

Network Details: The CNN encoder has 4 filters (widths 1 to 4) of 128 hidden units each in the convolutional layer (encoder output size=512). We use a stride size of 1 and the SAME padding for the convolutional and max-pooling layers. The decoder is a single layer of 512 LSTM units. We used the same configuration for both bilingual and multilingual experiments across all datasets for convenience after exploration on some language pairs. We apply dropout (Srivastava et al., 2014) (probability=0.5) at the output of the encoder and decoder, and SGD with the ADAM optimizer (Kingma and Ba, 2014) (learning rate=0.001). We trained our models for a maximum of 40 epochs (which we found sufficient for our models to converge) and a batch size of 32. In each training epoch, we cycle through the parallel corpora of each language pair. The parallel corpora are roughly of the same size. Better training schedules could be explored in future.

Languages: We experimented with two sets of orthographically similar languages:
Indian languages: (i) Hindi (*hi*), Bengali (*bn*) from the Indo-Aryan branch of Indo-European family (ii) Kannada (*kn*), Tamil (*ta*) from the Dravidian family. We studied Indic-Indic transliteration and transliteration involving a non-Indian language (English↔Indic). We mapped equivalent characters in different Indic scripts in order to build a common vocabulary based on the common offsets of the Unicode codepoints (Kunchukuttan et al., 2015).
Slavic languages: Czech (*cs*), Polish (*pl*), Slovenian (*sl*) and Slovak (*sk*). We studied Arabic↔Slavic transliteration. Arabic is a non-Slavic language (Semitic branch of Afro-Asiatic) and uses an *abjad* script in which vowel diacritics are omitted in general usage.

The languages chosen are representative of languages spoken by some major groups of peoples

en-Indic		Indic-en		Indic-Indic			ar-Slavic	
en-hi	12K	hi-en	18K		bn	kn	ta	ar-cs 15K
en-bn	13K	bn-en	12K	hi	3620	5085	5290	ar-pl 15K
en-kn	10K	kn-en	15K	bn		2720	2901	ar-sl 10K
en-ta	10K	ta-en	15K	kn			4216	ar-sk 10K

Table 1: Training set statistics for different datasets (number of word pairs). *Validation set*: 1K (en→Indic & ar↔Slavic), 500 (Indic→en, Indic-Indic). *Test set*: 1K (all pairs).

Pair	Src	Tgt
en-hi	KANAKLATA	कनकलता (*kanakalatA*)
en-kn	LEHMANN	ಲೆಹಮನ್ (*l.ehaman*)

(a) English-Indic

Pair	Src	Tgt
pl-ar	DUMITRESCU	د ومـيـترسـكـو (*dwmytrskw*)
cs-ar	MAURICE	مـو ريـس (*mwrys*)

(b) Slavic-Arabic

Table 2: Examples of transliteration pairs from our datasets

which exhibit orthographic similarity: Indic, Romance, Germanic, Slavic, *etc.* These languages are spoken by around 2 billion people. So our approach addresses a major chunk of the world's people.

Datasets: (See Table 1 for statistics of datasets).

We used the official *NEWS 2015* shared task dataset (Banchs et al., 2015) for English to Indic transliteration. This dataset has been used for many editions of the NEWS shared tasks. We split the *NEWS 2015* training dataset as the train and validation data for Indic-English transliteration. For testing, we used the *NEWS 2015* dev-test set. We created the Indian-Indian parallel transliteration corpora from the English to Indian language training corpora of the NEWS 2015 dataset by mining name pairs which have English names in common.

We mined the Arabic-Slavic dataset from *Wikidata* (Vrandečić and Krötzsch, 2014), a structured knowledge base containing *items* (roughly entities of interest). Each item has a *label* (title of item page) which is available in multiple languages. We extracted labels from selected items referring to named entities (persons, organizations and locations) to ensure that we extract parallel transliterations (as opposed to translations).

Pair	P	B	M	Pair	P	B	M
Similar Source and Target Languages							
Indic-Indic (45.5%)							
bn-hi	**29.74**	19.08	27.69	kn-bn	28.59	24.04	**37.47**
bn-kn	17.62	18.14	**27.74**	kn-ta	34.89	30.85	**38.30**
hi-bn	29.92	25.46	**39.15**	ta-hi	**29.07**	19.24	28.97
hi-ta	25.15	28.62	**38.70**	ta-kn	26.99	19.86	**29.06**
Similar Source Languages							
Slavic-Arabic (55.8%)				*Indic-English (24.2%)*			
cs-ar	38.91	37.10	**59.17**	bn-en	**55.23**	48.93	54.01
pl-ar	34.70	34.80	**44.83**	hi-en	49.19	38.26	**51.11**
sk-ar	43.26	37.49	**62.21**	kn-en	42.79	33.77	**47.70**
sl-ar	41.90	36.74	**62.04**	ta-en	**33.93**	23.22	25.93
Similar Target Languages							
Arabic-Slavic (176.8%)				*English-Indic (1.1%)*			
ar-cs	15.41	12.08	**36.76**	en-bn	42.90	41.70	**46.10**
ar-pl	13.68	12.26	**24.21**	en-hi	60.50	**64.10**	60.70
ar-sk	15.24	13.82	**38.72**	en-kn	48.70	52.00	**53.90**
ar-sl	18.31	13.63	**44.35**	en-ta	52.90	**57.80**	55.30

Table 3: Comparison of bilingual (B) and multilingual (M) neural models as well as bilingual PBSMT (P) models (top-1 accuracy %). Figure in brackets for each dataset shows average increase in transliteration accuracy for multilingual neural model over bilingual neural model. Best accuracies for each language pair in **bold**.

Evaluation: We use top-1 exact match accuracy as the evaluation metric (Banchs et al., 2015). This is one of the metrics in the NEWS shared tasks on transliteration.

5 Results and Discussion

We discuss and analyze the results of our experiments.

5.1 Quantitative Observations

Table 3 compares results of bilingual (B) and multilingual (M) neural models as well as a bilingual transliteration system (P) based on phrase-based statistical machine transliteration (PBSMT). The PBSMT system was trained using *Moses* (Koehn et al., 2007) with no lexicalized reordering and uses monotonic decoding. We used a 5-gram character language model trained with Witten-Bell smoothing.

We observe that multilingual training substantially improves the accuracy over bilingual training in all datasets (an average increase of 58.2% over all

language pairs). Transliteration accuracy increases in all scenarios: (i) similar sources (ii) similar targets and (iii) similar sources & targets.

If we look at results for various language groups, transliteration involving Slavic languages and Arabic benefits more than transliteration involving Indic languages and English. Arabic→Slavic transliteration shows maximum improvement (average: 176.8%) while English→Indic pairs show minimum improvement (average: 1.1%).

We also see that the multilingual model shows significant improvements over a bilingual transliteration system based on phrase-based SMT. The PBSMT system is better than the bilingual neural transliteration system in most cases. This is consistent with previous work (Finch et al., 2015), where the standard encoder-decoder architecture (with attention mechanism) (Bahdanau et al., 2015) could not outperform PBSMT approaches. However, a model using target bidirectional LSTM with model ensembling (Finch et al., 2016) outperforms PBSMT models. These improvements are orthogonal to our work and could be used to further improve the bilingual as well as multilingual systems. The bilingual neural network models are not able to outperform the PBSMT models possibly due to the small size of the datasets and the limited depth of the network (single layer encoder and decoder).

5.2 Qualitative Observations

We see that multilingual transliteration is better than bilingual transliteration in the following aspects:

• Vowels are generally a major source of transliteration errors (Kumaran et al., 2010; Kunchukuttan and Bhattacharyya, 2015) because of ambiguities in vowel mappings. We see a major decrease in vowel errors due to multilingual training (average decrease of ∼20%). We observe substantial decrease in long-short vowel confusion errors (Indic languages as target languages) and insertion/deletion of *A* (English/Slavic as target). We also see a major improvement in Arabic→Slavic transliteration. The Arabic script does not represent vowels, hence the transliteration system needs to correctly generate vowels. The multilingual model is better at generating vowels compared to the bilingual model.

• We also observe that consonants with similar phonetic properties are a major source of transliteration errors, and these show a substantial decrease with multilingual training. For Indic-English transliteration, we see substantial error reduction in the following character pairs *K-C*, *T-D*, *P-B*. We also observe a decrease in confusion between aspirated and unaspirated sounds. For Arabic→Slavic transliteration, we see substantial error reduction for the following character pairs *K-C*, *F-V* and *P-B*.

Source	B	M
بـاستـور (*bAstwr*)	bastor	pastor
كـيـلـين (*kylyn*)	kelen	kailin

(a) Arabic-Czech examples

Source	B	M
वर्जिल (*varjila*)	vergill	virgil
एलिसन (*elisana*)	elissan	ellison

(b) Hindi-English examples

Table 4: Examples of bi- *vs.* multi-lingual outputs. *ar* and *hi* text are also shown using Buckwalter and ITRANS romanization schemes respectively

• For Slavic→Arabic, we observed a significant reduction in the number of errors related to characters representing fricative sounds like *j,s,z,g* (Buckwalter romanization).

• The multilingual system seems to prefer the canonical spellings, even when other alternative spellings seem faithful to the source language phonetics. The system is thus able to learn conventional usage better than the bilingual models. *e.g. morisa* (Hindi, romanized text shown) is transliterated incorrectly to the phonetically acceptable English word *moris* by the bilingual model. The multilingual model generates the correct system *Maurice*.

• Since Indic scripts are very phonetic, very few non-canonical spellings are possible. As a consequence, vowel error reduction was also minimum for English-Indic transliteration (10%). This may partly explain why multilingual training provides minimal benefit for English-Indic transliteration.

Table 4 shows some examples where multilingual output is better than the bilingual output.

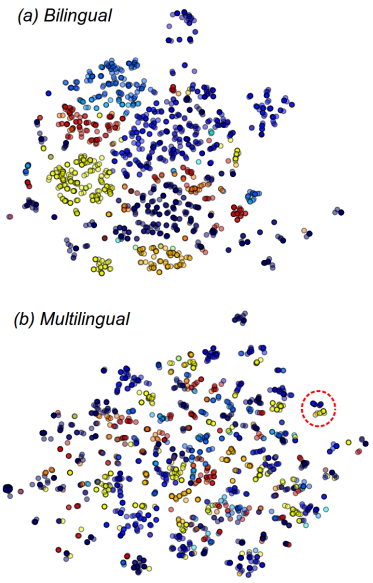


Figure 2: Visualization of contextual representations of vowels for *hi-en* transliteration. Each colour represents a different vowel.

5.3 Analysis

We investigated a few hypotheses to understand why multilingual models are better:

Better contextual representations for vowels: We hypothesize that the encoder learns better contextual representations for vowels. To test this hypothesis, we studied 3 character long sequences from the test set with a vowel in the middle (*i.e.,* 1-char window around vowel). We processed these sequences through the encoder of the bilingual and multilingual transliteration systems to generate the encoder output corresponding to the vowels. For instance, for the vowel *a* in the word *part*, we encode the 3 character sequence *par* using the encoder. The encoder output corresponding to the character *a* is considered the contextual representation of the character *a* in this word. Figure 2 shows a visualization of these contextual representations of the vowels using t-SNE (van der Maaten and Hinton, 2008). For the bilingual model, we observe that the contextual representations of same vowels tend to cluster together. For the multilingual model, the clustering is more specialized. The representations are grouped by the vowel along with the context. For instance, the region highlighted in the plot shows representations of Hindi vowels *e* (yellow) and *i* (blue) followed by the consonant *v*. Other vowels with the same context are seen in the same region too. This suggests that the multilingual model is able to learn specialized representations of vowels in different contexts and this helps the decoder generate correct transliterations.

More monolingual data: In the many-one scenario, more monolingual data is available for the target language since target words from all training language pairs are available. We hypothesize that this may help the decoder to better model the target language sequence. To test this, we decoded the test data using the bilingual models along with a larger target RNN LM (with LSTM units) using shallow fusion (Gulcehre et al., 2017). The RNN LM was trained on all the target language words across all parallel corpora. These experiments were performed for Indic-English and Slavic-Arabic pairs. We did not observe any major change in the transliteration accuracies of bilingual models due to integration of a larger LM. Thus, larger target side data does not explain the improvement in transliteration accuracy due to multilingual transliteration.

More parallel data: Multilingual training pools together parallel corpora from multiple orthographically similar languages, which effectively increases the data available for training. To test if this explains the improved performance, we compared multilingual and bilingual models under similar data size conditions, *i.e.,* the total parallel corpora size across all language pairs used to train the multilingual model is equivalent to the size of the bilingual corpus used to train the bilingual model. Specifically we compared the following under similar data size conditions: (a) *{bn,hi,kn,ta}-en* multilingual model (50% *hi-en* training pairs) with *hi-en* bilingual model (b) *{cs,pl,sk,sl}-ar* multilingual model (30% *pl-ar* training pairs) with *pl-ar* bilingual model. Table 5 shows the results of these experiments. We observed that the multilingual system showed significantly higher transliteration accuracy compared to the bilingual model for Polish-Arabic. For Hindi-English, the bilingual model was better than the multilingual model. So, we cannot decisively conclude that the performance improvement of multilingual transliteration can be attributed to the effective increase in training data size. In a multilingual training scenario, data from other languages act as noisy

Pair	B	M
pl-ar	14.64	44.83
hi-en	38.26	35.93

Table 5: Results of experiments under balanced data conditions

Pair	sep_{dec}	sep_{out}	sep_{none}	Pair	sep_{dec}	sep_{out}	sep_{none}
Indic-Indic							
bn-hi	27.28	27.69	**28.72**	kn-bn	32.22	**37.47**	35.76
bn-kn	25.86	**27.74**	27.11	kn-ta	40.06	38.30	**40.37**
hi-bn	37.22	39.15	**39.35**	ta-hi	27.74	**28.97**	28.45
hi-ta	35.74	**38.70**	35.54	ta-kn	28.13	**29.06**	28.65
Arabic-Slavic				*English-Indic*			
ar-cs	**39.68**	36.76	35.95	en-bn	41.00	**46.10**	44.10
ar-pl	**27.25**	24.21	26.85	en-hi	56.00	60.70	**61.60**
ar-sk	**41.36**	38.72	40.14	en-kn	49.30	53.90	**54.30**
ar-sl	**49.14**	44.35	45.88	en-ta	53.00	**55.30**	52.90

Table 6: Comparison of multilingual architectures

version of data from the original language and supplement the available bilingual data, but they cannot necessarily substitute data from the original language pair.

5.4 Comparison with variant architectures

We compare three variants of the encoder-decoder architecture: (a) our model: target language specific output layer (and parameters) (sep_{out}), (b) every target language has its own decoder and output layer (sep_{dec}) (Lee et al., 2017), and (c) all languages share the same decoder and output layer, but the first token of the input sequence is a special token to specify target language (sep_{none}) (Johnson et al., 2017). These architectures differ in the degree of parameter sharing. sep_{dec} has fewer shared parameters than our model and sep_{none} has more shared parameters than our model. In all three cases, the encoder is shared across all source languages. Table 6 shows the results of this comparison. We cannot definitively conclude if one model is better than the other. Except for Arabic-Slavic transliteration, the trend seems to indicate that models with greater parameter sharing (sep_{out}/sep_{none}) may perform better. In any case, given the comparable results we prefer models with fewer model parameters (sep_{out}/sep_{none}).

6 Zeroshot Transliteration

In the previous sections, we have shown that multilingual training is beneficial for language pairs observed during training. In addition, the encoder-decoder architecture opens up the possibility of *zeroshot transliteration i.e.,* transliteration between language pairs that have not been seen during training. The encoder-decoder architecture decouples the source and target language network components and makes the architecture more modular. As a consequence, we can consider the encoder output (for the source language) to be embedded in a language-neutral, common subspace - a sort of interlingua. The decoder proceeds to generate the target word from the language neutral representation of the source word. Hence, training on just a few language pairs is sufficient to learn all language-specific parameters - making zeroshot transliteration possible.

Before describing different zeroshot transliteration scenarios, we introduce a few terms. A language that is the source in any language pair seen during training is said to be *source-covered*. We can define *target-covered* languages analogously. Now, we can envisage the following zeroshot transliteration scenarios: (a) *unseen language pairs*: both the source and target languages are covered, but the pair was not observed during training, (b) *unseen source language*: the source language is not covered, but it is orthographically similar to other source-covered languages, (c) *unseen target language*: can be defined analogously. Next, we describe our proposed solutions to these scenarios.

6.1 Unseen Language Pair

We investigated the following solutions:

Multilingual Zeroshot-Direct: Since source and target languages are covered, we use the trained multilingual model discussed in previous sections for source-target transliteration.

Model selection can be an issue for this approach. As discussed earlier, model selection using validation set accuracy for each language pair is better than average validation set loss. For an unseen language pair, we cannot use validation set accuracy/loss for model selection (since validation data is not available). So we explored the following model selection criterion: maximum average validation set accuracy

across all the trained language pairs (*sc_acc*). We also compared *sc_acc* to the model with least average validation set loss over all training language pairs (*sc_loss*).

Nevertheless, there are inherent limitations to model selection by averaging validation accuracy or loss for trained language pairs. Irrespective of the model selection method used, the chosen model may still be suboptimal for unseen language pairs since the network is not optimized for such pairs.

Multilingual Zeroshot-Pivot: To address the limitations with zeroshot-direct transliteration, we propose transliteration from source to target using a pivot language, and pipelining the best source-pivot and pivot-target transliteration models. We choose a pivot language such that the network has been trained for source-pivot and pivot-target pairs. Since the network has been trained for the source-pivot and target-pivot pairs, we can expect optimal performance in each stage of the pipeline. *Note that we use the multilingual model for source-pivot and pivot-target transliteration* (we found it better than using the bilingual models). To reduce cascading errors due to pipelining, we consider the top-k source-pivot transliterations in the next stage of the pipeline. The probability for a target word **y** given the source word **x** is computed as:

$$p(\mathbf{y}|\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{y}|\mathbf{z}^i)p(\mathbf{z}^i|\mathbf{x}) \quad (2)$$

$\mathbf{z}^i$: i^{th} best source-pivot transliteration. We used *k=10*.

6.2 Unseen Source Language

An unseen source language can be easily handled in our architecture. Though the language has not been source-covered, a source word can be directly processed through the network since all source languages share the encoder and character embeddings.

6.3 Unseen Target Language

Handling an unseen target language is tricky since the output layer is specific to each target language. Hence, parameters for the unseen language's output layer cannot be learned during training. Note that even architectures where the entire network is completely shared between all language pairs (Johnson et al., 2017) cannot handle an unseen target language - the embedding for target language indicator tokens are not learned during training for unseen target languages. We found that simple approaches like assigning parameters for unseen target languages by averaging the parameters of the trained target languages do not work.

Hence, we use a target-covered language as proxy for the unseen target language. The simplest approach considers the output of the source to proxy language transliteration system as the target language's output. However, this doesn't take into account phonotactic characteristics of the target language. We propose to incorporate this information by using an RNN (using LSTM units) character-level language model of the target language words. While predicting the next output character during decoding, we combine the scores from the multilingual transliteration model and the target LM using shallow fusion of the transliteration model and the language model (Gulcehre et al., 2017).

Pair	**Biling Direct**	**Zeroshot Pivoting** †	**Zeroshot Direct**		
			sc_acc	*sc_loss*	*sc_ora*
bn-ta	16.20	**36.12** (hi)	31.79	27.45	34.47
ta-bn	18.48	**47.01** (hi)	24.87	17.97	25.28
hi-kn	34.66	**47.14** (ta)	44.48	42.13	43.05
kn-hi	34.12	**39.02** (ta)	40.45	37.28	40.96

† *best pivot for multilingual pivoting in brackets*

(a) Unseen language pair

Method	**Slavic-ar** *(cs-ar)*	**Indic-en** *(hi-en)*
Bilingual	37.10	38.26
Zeroshot	60.48	45.24
Multilingual	59.17	51.11

(b) Unseen source language

Method	**ar-Slavic** *(ar-cs)*		**en-Indic** *(en-hi)*	
	proxy	**acc**	**proxy**	**acc**
Bilingual	none	12.08	none	64.10
Proxy	sk	42.09	kn	9.00
	sl	41.89	bn	18.30
	pl	38.27	ta	2.90
Proxy + LMFusion	sk	42.20	kn	9.70
	sl	40.99	bn	18.10
	pl	36.35	ta	3.10

(c) Unseen target language

Table 7: Results for zeroshot transliteration

6.4 Results and Discussion

We discuss the results for each scenario below.

6.4.1 Unseen language pair

We experimented with transliteration between four Indic languages *viz., hi,bn,ta,kn*. We trained a multilingual model on 8 out of the 12 possible language pairs, covering all the 4 languages. The remaining 4 language pairs (*ta-bn, bn-ta, hi-kn* and *hi-kn*) are the unseen language pairs (results in Table 7a).

Zeroshot *vs.* Direct Bilingual: For all unseen language pairs, all zeroshot systems (pivot as well different direct configurations) are better than the direct bilingual system. Note that unlike the zeroshot systems, the bilingual systems were directly trained on the unseen language pairs. Yet the zeroshot systems outperform direct bilingual systems since the underlying multilingual models are significantly better than the bilingual models (as seen in Section 5). These results show that the multilingual model generalizes well to unseen language pairs.

Direct *vs.* Pivot Zeroshot: We also observe that the pivot zeroshot system is better than both of the direct zeroshot systems (*sc_acc* and *sc_loss*). To verify if the limitations of the model selection criterion explain the direct system's relatively lesser accuracy, we also considered an oracle direct zeroshot system (*sc_ora*). The oracle system selects the model with the best accuracy using a parallel validation corpus for the unseen language pair. This oracle system is also inferior to the pivot transliteration model. So, we can conclude that the network is better tuned for transliteration of language pairs it has been directly trained on. Hence, multilingual pivoting works better than direct transliteration in spite of the cascading errors involved in pipelining.

Model Selection Criteria: For the direct zeroshot system, the average accuracy (*sc_acc*) is a better model selection criterion than average loss (*sc_loss*).

6.4.2 Unseen source language

We conducted 2 experiments: (a) train on *(bn,kn,ta)-en* pairs and test on *hi-en* pair, and; (b) train on *(pl,sk,sl)-ar* pairs and test on *cs-ar* pair. See Table 7b for results. In this scenario, too, we observe that zeroshot transliteration performs better than direct bilingual transliteration. In fact, zeroshot transliteration is competitive with multilingual transliteration too (accuracy is $>$ 90% of multilingual transliteration). Though the network has not seen the source language, the encoder is able to generate source language representations that are useful for decoding.

6.4.3 Unseen target language

We conducted 2 experiments: (a) train on *ar-(pl,sk,sl)* pairs and test on *ar-cs* pair, and;(b) train on *en-(bn,kn,ta)* pairs and test on *en-hi* pair. The target languages used in training were the proxy languages. See Table 7c for results.

We observe contradictory results for the experiments. For *ar-cs*, the use of proxy language gives transliteration results better than bilingual transliteration. But, the proxy language is not a good substitute for Hindi. Shallow fusion of target LM with the transliteration model makes little difference.

We see that the transliteration performance of proxy languages is correlated to its orthographic similarity to the target language. Thus, it is preferable to choose a proxy with high orthographic similarity to the target language. We see one anomaly to this trend. Kannada as proxy performs badly compared to Bengali, though Kannada is orthographically more similar to Hindi. One reason could be the orthographic convention in Hindi and Bengali that the terminal vowel is automatically suppressed. In Kannada, the vowel has to be explicitly suppressed with a terminal *halanta* character. Simply deleting the terminal *halanta* in Kannada output to conform to Hindi conventions increases accuracy to 24.1% (better than Bengali). Clearly, shallow fusion is not sufficient to adapt a proxy language's output to the target language, and further investigations are required. If a proxy-target corpus is available, we can generate better transliterations via pivoting.

7 Incorporating Phonetic Information

So far, we considered characters as atomic units. We have thus relied on correspondences between characters for multilingual learning. In addition, for some languages, we can find an almost one-one correspondence from the characters to phonemes (a basic unit of sound). Each phoneme can be factorized into a set of articulatory properties like place of articulation, nasalization, voicing, aspiration, *etc.* If the input for transliteration incorporates these phonetic properties, it may learn better character representations

	Pair	O	P^h	Pair	O	P^h
Indic-English	bn-en	54.01	**55.53**	kn-en	47.70	**53.31**
	hi-en	51.11	**54.86**	ta-en	25.93	**29.63**
Indic-Indic	bn-hi	27.69	**28.51**	kn-bn	37.47	**39.19**
	bn-kn	27.74	**29.72**	kn-ta	38.30	**41.93**
	hi-bn	**39.15**	37.73	ta-hi	28.97	**29.17**
	hi-ta	38.70	**39.00**	ta-kn	29.06	**31.54**

Table 8: Onehot (O) *vs.* phonetic (P^h) input

across languages by bringing together similar characters. *e.g.* the Kannada character ಳ (*La*), has no Hindi equivalent character, but the Hindi character ल (*la*) is the closest character. The two characters differ in terms of one phonetic feature (the *retroflex* property), which can be represented in the phonetic input and can serve to indicate the similarity between the two characters.

We incorporated phonetic features in our model by using feature-rich input vectors instead of the conventional onehot vector input for characters. Our phonetic feature input vector is a bitvector encoding the phonetic properties of the character, one bit for each value of every property. The multiplication of the phonetic feature vector with the weight matrix in the first layer generates phonetic embeddings for each character. These are inputs to the encoder. Apart from this input change, the rest of the network architecture is the same as described in Section 3.2.

Experiments: We experimented with Indian languages (Indic→English and Indic-Indic transliteration). Indic scripts generally have a one-one correspondence from characters to phonemes. Hence, we use phonetic features described by Kunchukuttan et al. (2016) to generate phonetic feature vectors for characters (available via the *Indic NLP Library*[1]). These Indic languages are spoken by nearly a billion people and hence the use of phonetic features is useful for many of the world's most widely spoken languages.

Results and Discussion: Table 8 shows the results. We observe that phonetic feature input improves transliteration accuracy for Indic-English transliteration. The improvements are primarily due to reduction in errors related to similar consonants like (T,D), (P,B), (C,K) and the use of H for aspiration.

[1] https://github.com/anoopkunchukuttan/indic_nlp_library

For Indic-Indic transliteration, we see moderate improvement in transliteration accuracy due to phonetic feature input. Since the source as well as target scripts are largely phonetic, phonetic representation may not be useful in resolving ambiguities (unlike Indic-English transliteration). Again, we see improvements due to reduction of errors related to similar consonants.

8 Conclusion and Future Work

We show that multilingual training using a neural encoder-decoder architecture significantly improves transliteration involving orthographically similar languages compared to bilingual training. Our key idea is maximal sharing of network components in order to utilize high task relatedness on account of orthographic similarity. The primary reasons for the improvements could be: (a) learning of specialized representations by the shared encoder and; (b) ability to learn canonical spellings. We also show that the multilingual transliteration models can generalize well to language pairs not encountered during training and observe that *zeroshot* transliteration can outperform direct bilingual transliteration in many cases. Moreover, multilingual transliteration can be further improved by shared phonetic input.

Transliteration is an example of a sequence to sequence task which is characterized by the following properties: (a) small vocabulary (b) short sequences (c) monotonic transformation (d) unequal source and target sequence length (e) significant vocabulary overlap across languages. *Given the benefits we have shown for multilingual transliteration, other NLP tasks that can be characterized similarly (viz., grapheme to phoneme conversion, translation of short text like tweets and headlines between related languages at subword level, and possibly speech recognition as well as TTS) could also benefit from multilingual training.*

9 Acknowledgements

We would like to thank Rudramurthy V for making available code for parsing Wikidata and extracting multilingual named entities. We would also like to thank the action editor and reviewers for their valuable comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Rafael E. Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A. Kumaran. 2015. Report of NEWS 2015 Machine Transliteration Shared Task. In *Proceedings of the Fifth Named Entities Workshop*.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*.

Rich Caruana. 1997. Multitask learning. *Machine learning*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-Task learning for multiple language translation. In *Annual Meeting of the Association for Computational Linguistics*.

Andrew Finch and Eiichiro Sumita. 2010. A Bayesian model of bilingual segmentation for transliteration. In *International Workshop on Spoken Language Translation*.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2015. Neural network transduction models in transliteration generation. In *Proceedings of the Fifth Named Entities Workshop*.

Andrew Finch, Lemao Liu, Xiaolin Wang, and Eiichiro Sumita. 2016. Target-Bidirectional neural models for machine transliteration. In *Proceedings of The Sixth Named Entities Workshop*.

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. 2016. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into Neural Machine Translation. *Computer Speech and Language*.

Junqing He, Long Wu, Xuemin Zhao, and Yonghong Yan. 2017. HCCL at SemEval-2017 Task 2: Combining multilingual word embeddings and transliteration model for semantic similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation*.

Jagadeesh Jagarlamudi and Hal Daumé III. 2012. Regularized interlingual projections: Evaluation on multilingual transliteration. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Annual Meeting of the Association for Computational Linguistics*.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. DirecTL: A language-independent approach to transliteration. In *Proceedings of the 2009 Named Entities Workshop: Shared task on transliteration*.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda B. Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*.

Sarvnaz Karimi, Falk Scholer, and Andrew Turpin. 2011. Machine transliteration survey. *ACM Computing Surveys*.

Mitesh M. Khapra, A. Kumaran, and Pushpak Bhattacharyya. 2010. Everybody loves a rich cousin: An empirical study of transliteration through bridge languages. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*.

A. Kumaran, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2010. Compositional machine transliteration. *ACM Transactions on Asian Language Information Processing*.

Anoop Kunchukuttan and Pushpak Bhattacharyya. 2015. Data representation methods and use of mined corpora for Indian language transliteration. In *Named Entities Workshop*.

Anoop Kunchukuttan, Ratish Puduppully, and Pushpak Bhattacharyya. 2015. Brahmi-Net: A transliteration and script conversion system for languages of the Indian subcontinent. In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies: System Demonstrations*.

Anoop Kunchukuttan, Pushpak Bhattacharyya, and Mitesh M. Khapra. 2016. Substring-based unsupervised transliteration with phonetic and contextual knowledge. In *SIGNLL Conference on Computational Natural Language Learning*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level Neural Machine Translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*.

Bruno Pouliquen, Ralf Steinberger, Camelia Ignat, Temnikova Irina, and Anna Widiger. 2005. Multilingual person name recognition and transliteration. *Corela*.

Ram Prakash. 2012. Quillpad multilingual predictive transliteration system. In *Proceedings of the Second Workshop on Advances in Text Input Methods*.

V. Rudramurthy, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2016. Sharing network parameters for crosslingual Named Entity Recognition. *arXiv preprint arXiv:1607.00198*.

Amrita Saha, Mitesh M. Khapra, Sarath Chandar, Janarthanan Rajendran, and Kyunghyun Cho. 2016. A correlational encoder-decoder architecture for pivot-based sequence generation. In *International Conference on Computational Linguistics*.

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 Nadejde. 2017. Nematus: A toolkit for neural machine translation. In *Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.

Kārumūri V Subbārāo. 2012. *South Asian languages: A syntactic typology*. Cambridge University Press.

Harshit Surana and Anil Kumar Singh. 2008. A more discerning and adaptable multilingual transliteration mechanism for Indian languages. In *Proceedings of the Third International Joint Conference on Natural Language Processing*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Su-Youn Yoon, Kyoung-Young Kim, and Richard Sproat. 2007. Multilingual transliteration using feature based phonetic method. In *Annual Meeting-Association for Computational Linguistics*.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource Neural Machine Translation.