A Comparative Study of Extremely Low-Resource Transliteration of the World's Languages

Winston Wu, David Yarowsky
Department of Computer Science, Center for Language and Speech Processing
Johns Hopkins University
{wswu,yarowsky}@jhu.edu

Abstract

Transliteration from low-resource languages is difficult, in large part due to the small amounts of data available for training transliteration systems. In this paper, we evaluate the effectiveness of several translation methods in the task of transliterating around 1000 Bible names from 591 languages into English. In this extremely low-resource task, we found that a phrase-based MT system performs much better than other methods, including a g2p system and a neural MT system. However, by combining the data and training a single neural system, we discovered significant gains over single-language systems. We release the output from each system for comparative analysis.

Keywords: Bible, alignment, named entities, translation, transliteration

1. Introduction

Transliteration is the process of converting text from one script to another. For example, Росси́я is *transliterated* as Rossiya but would be *translated* as Russia. Transliteration is important in the process of borrowing names between languages. In the case of low-resource languages, there is often little training data with which to train transliteration models. Thus, one major obstacle for low-resource languages is the problem of out of vocabulary words (in the transliteration setting, these would be unknown characters). Machine translation systems might use a transliteration step to resolve unknown words, especially names; in transliteration, one can perform a similar process by attempting to resolve unknown characters. In this paper, we compare and contrast multiple methods of transliterating Bible names in over 500 languages into English and evaluate the effectiveness of a pre- and post-transliteration step in transliteration in resolving unknown characters. Our results indicate that a phrase-based machine translation system is the most effective when training on data in the order of hundreds of words, while by simply concatenating multiple languages' training data with some preprocessing, a single neural MT system trained to transliterate multiple languages to a single target language significantly outperforms the single-language systems. We release each system's transliteration output as a dataset for comparative analysis. To our knowledge, this is the first study of such scale that compares such a variety of methods on such small corpora.

2. Related Work

Machine transliteration has been tackled using a variety of methods. For a comprehensive survey, we refer the reader to Karimi et al. (2011). In this paper, we focus on transliteration of named entities across multiple language, especially in a low-resource setting, using the paradigm of monotonic machine translation (Virga and Khudanpur, 2003). Our work is most similar to Irvine et al. (2010), who built character-based machine translation systems using names mined from Wikipedia. Their setting is higher resource, as they use data acquired from the web, and they experiment on a smaller set of languages. Other recent approaches to transliteration of low resource languages include Mayhew et al. (2016), who explore using surrogate languages in place of a language not in Wikipedia, Rosca and Breuel (2016) who showed state of the art transliteration performance using a neural sequence to sequence model, and Jiampojamarn et al. (2010), who explore several methods for language-independent transliteration mining. Qian et al. (2010) also developed a toolkit to extract translation pairs from comparable corpora.

We compare several open-source toolkits for translation: Moses (Koehn et al., 2007), a phrase-based statistical machine translation toolkit; Sequitur (Bisani and Ney, 2008), a grapheme to phoneme system; and OpenNMT (Klein et al., 2017), a neural sequence-to-sequence machine translation system. While Sequitur is not commonly used in MT, Sequitur and Moses have been compared for speech recognition tasks (Schlippe et al., 2014).

3. Data

We use the Bible names translation matrix dataset (Wu and Yarowsky, 2018), which contains 1129 person and place names in the Bible aligned across 519 languages. They constructed this name translation matrix using a combination of distance-based, MT transliteration, and string transduction rules to improve the alignments from a baseline aligner. The scope of this data accentuates the low-resource setting, which is reasonable for many of the world's languages; the Bible may be one of the only bilingual resources available for certain languages.

4. Experiments

We perform three major transliteration experiments: the first, in which we compare several existing machine translation systems in the task of transliteration; the second, in which we evaluate the effectiveness of pre- and post-processing the data with a baseline transliterator; and the third, in which we employ a single NMT system to translate between all pairs of languages. The experiments consisted of training each system to transliterate from a source language into English[1]. We used names from the aforementioned dataset in a 80-10-10 train-dev-test split. All names

were lowercased, and characters were space-separated.

In our experiments, the baseline system was Unidecode,[2] a Python library that provides a language-independent mapping from a Unicode character to a fixed ASCII string. While this is a naive baseline, for languages that do not have much data to train transliteration systems, this may be one of the few transliteration options available.

Moses was trained using a vanilla setup, with a 4-gram KenLM (Heafield, 2011) language model, tuning with MERT (Och, 2003), and setting the distortion limit to 0 to prevent reordering, which does not occur during transliteration. Sequitur was trained iteratively using the `--ramp-up` flag three times. For OpenNMT, we used the following hyperparameters: a 2 layer GRU for encoder and decoder, optimizer is Adadelta, 0.2 dropout rate, hidden size 200, embedding size 200, no length normalization. We trained each model for 50 epochs and used the model with the lowest validation perplexity.

In the second experiment, we examine how pre- and post-transliteration using a baseline transliterator (Unidecode) affects a Moses-based transliterator. Pre-transliteration is a preprocessing step that uses Unidecode to convert the input data into ASCII letters before it is processed by Moses. In post-transliteration, Unidecode postprocesses the output from Moses to catch foreign (non-ASCII) letters that were not transliterated. We hypothesize that if there is a one-to-one character mapping between the characters of the source language and ASCII letters, then neither pre- nor post-transliteration should help. We expect the preprocessing step to reduce the character set of the source language, which has the effect of losing information if multiple characters can map to a single ASCII character (e.g. in Greek, σ and ς both correspond to the letter s). For postprocessing, we expect that this step will correct characters that were not transliterated, which can occur if they were not seen in the training set, similar to unseen words in MT.

In our final experiment, we exploit the multilinguality of the data for transfer learning. Inspired by the winning system in the SIGMORPHON 2016 shared task (Kann and Schütze, 2016), we train a single neural MT system on the concatenation of the entire training set. This allows the system to learn a joint model from multiple source languages to a single target language. In addition, by concatenating training sets of each of the single language-pair systems, the multi-source approach can circumvent the data scarcity problem. Each training example was split into spaces, with a special source language symbol prepended, as shown in Table 1. No target language symbol was used because the target language was only English. Since the training and test data are different (and substantially larger), this system is not directly comparable to the above approaches but rather represents

[1]Note that the training and test sets are in the same domain, i.e. named entities. While it may be interesting to test on an unrelated set of words, the results are not likely to be encouraging unless these words are orthographically/phonologically similar to their English counterparts (e.g. cognates or borrowed words).

[2]`https://pypi.python.org/pypi/Unidecode`

Source	Target
<ann> p o l ǫ k i s	p o l l u x
<bnp> p o l u k s	p o l l u x
<kwf> p o l a k s	p o l l u x
<msy> p o l l u k s	p o l l u x
<mti> p o r a k u s	p o l l u x
<mto> p ó l u x	p o l l u x
<ncj> p ó l u x	p o l l u x
<rus> п о л л у к с а	p o l l u x

Table 1: Bitext format for the OpenNMT Multi experiment. The target word is the English name Pollux.

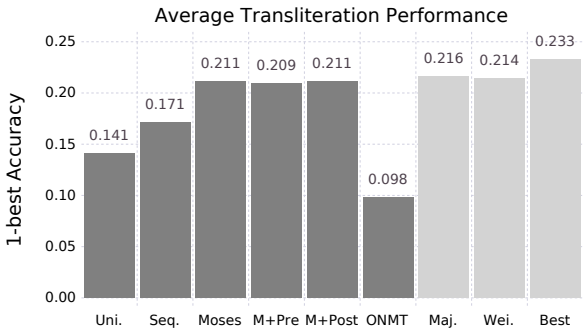

Figure 1: Average transliteration performance for single language pair systems. Light gray indicates system combination (majority, weighted, and best per language).

5. Results, Discussion, and Error Analysis

Each system was evaluated on 1-best exact-match accuracy. Due to the extremely low-resource nature of the data (on the order of a few hundred training examples), the task proved to be quite difficult. On average, as shown in Fig. 1, Sequitur performs better than the baseline Unidecode, and the Moses transliterator performs best overall. Moses with pre- and post-processing perform comparably on average.

Since there are a number of different systems trying to accomplish the same task, a natural question is whether combining the systems' outputs would result in improvements. Out of the six systems, simply choosing the most common hypothesis works better than Moses alone. Using a weighted combination, where each hypothesis is weighted by their respective system's average performance also results in gains over a single system. Finally, if we consider the best performing system on a per-language basis, the average performance sees a small increase.

In the following section, we examine each system in relation to the vanilla Moses system (without pre- or post-transliteration). Code to reproduce these results are available[3]. The three letters that represent a language are ISO 639-3 language codes, and asterisks denote incorrect transliterations.

Baseline. As a language-independent baseline, we expected that Unidecode would perform worse in most cases. Indeed, the baseline obtained 0% accuracy for 86 languages. This is due to the presence of morphological affixes. Con-

[3]`github.org/wswu/trabina`

sider the following:

Apurinã (apu)	English	Unidecode	Moses
épeso	ephesos	*epeso	*ephesus[4]
xório	julius	*xorio	julius
nikorao	nicolas	*nikorao	*nicolaus

The -o ending in Apurinã, which likely denotes a nominative, is translated to -us by Moses. Unidecode cannot handle this phenomenon, since it operates only on single characters; here it merely stripped off accents from each character. In addition, since Unidecode is not a translation system, it will never convert x → j or k → c, since these are already ASCII letters. Many other languages exhibit similar morphological patterns that could not be handled by Unidecode. However, the baseline actually performs better than Moses in 42 languages, which is a testament to the difficulty of our task. In these cases, we believe that Moses may have learned an incorrect translation model from the tiny amounts of training data, when simply passing the character through the system unchanged or applying a script conversion (e.g. Cyrillic to Roman) would have resulted in the correct answer. The following are some instances where Unidecode transliterates the correct name, while Moses does not.

Lang	Unidecode	Moses
Siyin (csy)	enoch	*enoc
Guahibo (guh)	jordan	*jordam
Ukranian (ukr)	puteoli	*putheoli
Murrinh-patha (mwf)	moses	*mouseus

Sequitur. While Sequitur is more commonly used in the speech community as a grapheme-to-phoneme software, we considered using Sequitur because transliteration is performed on sequences of characters rather than entire words. Sequitur's average performance was only below Moses' by a few percentage points, and it actually outperformed Moses in 145 languages. Some examples of Sequitur's successes over Moses follow:

Lang	Sequitur	Moses
Amele (aey)	elam	*ilam
	abilene	*abylene
Balinese (ban)	cleopas	*clopas
Bukawa (buk)	bartimaeus	*batimeas
Hawaiian Pidgin	castor	*casthor
(hwc)	phrygia	*phirygia
Hote (hot)	miletus	*miretus
	philetus	*piletus
	troas	*troaz

We observe that the mistakes are natural looking mistakes: mixing up letters that are phonologically similar (*l* and *r*, *s* and *z*, and some vowels), and occasionally adding or removing an *h*. Our hypothesis is that, in these cases, Moses' language model is biasing the system away from the correct answer. While the language model is trained on same-domain data (i.e. Bible named entities), it is possible that the language model would give a low score to unusual letter sequences like "aeus" in Bartimaeus or "phry" in Phrygia. Further investigation is needed to determine the role the language model plays in the transliteration process.

OpenNMT. Neural machine translation is state-of-the-art for many language pairs (Bojar et al., 2016). In this transliteration task, however, the neural MT system performed the worst overall. For 22 languages, OpenNMT performed better than Moses, but for most of these languages, the accuracies were under 10%, representing very small gains.
While it is possible that the parameters we used were not optimal for character-based transliteration, it is likely that the size of the data was just too small for a neural model to effectively learn from. This corroborates a finding from Koehn and Knowles (2017) that neural MT models tend to perform better than phrase-based models only past a certain threshold of data size (a corpus size of over 10^7 words). We observed that the neural MT system often prefers shorter words compared to the phrase-based Moses, and perplexity on the development set was generally lowest after around 20–25 epochs training, after which overfitting was evident. Below are some example transliterations from Moses and OpenNMT:

Lang	Source	Moses	OpenNMT
Qaqet (byx)	aleksandria	alexandria	*alandria
Frafra (gur)	metusela	*methushelah	*metusel
	alekzander	*alechzander	alexander
Hiri Motu	eparona	*epharon	ephron
(hmo)	mikaela	*micael	michael

5.1. Resolving Unknown Characters

Analogous to OOVs in machine translation, handling unknown characters is vital in achieving high transliteration accuracy. Since the target language in our experiments is always English, we utilized Unidecode to transliterate characters into ASCII letters.

Preprocessing. While the average accuracy of Moses + preprocessing was almost the same as without preprocessing, preprocessing helped in 195 languages but hurt in 206 languages, suggesting that preprocessing is largely dependent on the language.

Lang	Source	Moses	+Pre
Ankave (aak)	segɨria	*cenria	cenchrea
Greek (ell)	εῦα	*eῦa	eva
Ukranian (ukr)	марта	*marta	martha
Armenian (hye)	սողոմոն	solomon	*solomone
Russian (rus)	косам	cosam	*kosam
Ossetian (oss)	тимейы	timaeus	*timee

While pretransliterating helps in some cases, in other cases it appears to conflate character mappings, thereby removing information crucial for transliteration. Interestingly, for the name "Martha", although Moses with preprocessing gives the correct English name, Moses without preprocessing actually produced a closer representation of how the name would be pronounced in Ukranian[5]. For "Cosam", preprocessing the к → k seems to have prevented the correct letter

[4] "Ephesus" is also a valid spelling of this ancient Greek city, but the dataset contains only one correct gold.

[5] Martha originated from the Aramaic Martâ, which was borrowed into Greek as Μάρθα, which was transliterated into Latin as Martha.

c from appearing in the output instead. From our experiments, it is not clear in what cases one should apply pre-transliteration.

Postprocessing. We found that postprocessing the output never hurts performance, but improves performance only for four source languages: Acehnese (ace), German (deu), Korean (kor), and Munduruku (myu). For Acehnese, German, and Munduruku, the improvements largely consisted of removing a diacritic on a letter (e.g. Joël → Joel). The interesting case however is Korean.

English	malchus	felix	crete
Moses	* 말 chus	*pel 릭 s	* 크 re 테
+Post	malchus	*pelrigs[6]	*keurete

Due to Korean's system of composing characters based on sound, Korean words may contain characters not seen during training time. In these cases, Moses treats the character as an unknown word and outputs the same character untranslated. This is a prime example of a situation in which we expect post-processing to help. In the case of Malchus, post-processing correctly transliterates the unknown Korean character, resulting in the correct English name, while for Felix and Crete, the post-transliterated name is not an exact match but is recognizably close.

5.2. Analyses by Category

We additionally examine transliteration performance not by language, but by features of the words themselves. By aggregating all words and stratifying based on word length, we can gain additional insights.

English word length. Roughly two thirds of English names are longer than 5 characters. When separating these names into short (length $\leq$ 5) and long (length $>$ 5), we see that all systems had an easier time transliterating shorter words.

System	$\leq$ 5 chars	$>$ 5 chars
Moses	3079/10092 (.31)	4536/24314 (.19)
Sequitur	2492/10092 (.25)	3756/24314 (.15)
Unidecode	2623/10092 (.26)	2437/24314 (.10)

Edit distance between source and target. We see that around two thirds of the names in the entire set have an edit distance of 3 or less to the English. This is not too surprising, especially since most languages use Roman script. We see that the performance of transliteration systems degrades as the difference between source and target increases.

System	Dist $\leq$ 3	Dist $>$ 3
Moses	6866/21746 (.32)	749/12660 (.06)
Sequitur	5867/21746 (.27)	381/12660 (.03)
Unidecode	4995/21746 (.23)	65/12660 (.01)

Roman vs non-roman characters. There are surprisingly few languages in the world that do not use a Roman character set; the data set contains 35 such languages. These include the Arabic and Cyrillic scripts, which are used in several languages, as well as other scripts, including Hangul and the family of Brahmic scripts, that are specific to a single language. We find that on average, the transliteration of languages in Roman script performs better that that of non-Roman script languages. A surprising result is that Sequitur does not transliterate any non-Roman words correctly, which may be due to encoding issues.

System	Roman	Non-Roman
Moses	7316/32129 (.23)	304/2277 (.13)
Sequitur	6248/32129 (.19)	0/2277 (.00)
Unidecode	4988/32129 (.16)	72/2277 (.03)

For the non-Roman script languages, we performed the same analysis as above. The following table shows average accuracy on languages written in non-Roman scripts.

Model	Accuracy
Baseline	0.031
Moses	0.125
+pre	0.119
+post	0.125

The baseline performs poorly as expected. Pretransliteration improves over Moses in 11 languages but underperforms in 17. Even on this subset of non-Roman script languages, there seems to be no pattern as to whether preprocessing is effective or not, which reiterates our findings for the entire dataset. Even within a language family, for example, transliterating Kannada and Marathi has slightly higher performance with preprocessing, while Tamil suffers. Preprocessing does slightly better overall for Greek, but when examining the transliterated names, there is evidence both for and against preprocessing:

Greek	Moses	+Pre
ιαρεδ	*jered	jared
ιεριμωθ	jerimoth	*jeremoth

Transfer learning. The single neural MT system trained on the concatenation of the training data for all languages performed much better than the other systems in our experiments, achieving a 69% one-best accuracy on the concatenation of the test sets. This massive gain stems from the combination of the 1000x increase in training data and the neural architecture's ability to effectively leverage the commonality between languages. This result indicates that this transfer learning technique works well when combining low-resource languages, even when each individual language pair may only have a miniscule amount of data.

6. Conclusion

We have performed an extensive comparison of several machine translation methods adapted for transliteration of 591 languages into English. By evaluating the performance of Unidecode, Sequitur, Moses, and OpenNMT across most of the world's languages, we observed that the phrase-based machine translation paradigm was the most effective for training character-based transliteration systems on tiny amounts of data. Performing a pre-processing

[6]Note that Korean does not have an 'f' sound.

Foreign	Lang	English	Unidecode	Sequitur	Moses	M+Pre	M+Post	OpenNMT
filipi'de	tur	philippi	filipi'de		philipia	philipia	philipia	philip
poluks	mnx	pollux	poluks	poluk	polucs	polucs	polucs	polug
timotio	aak	timothy	timotio	**timothy**	**timothy**	**timothy**	**timothy**	timothe
ibrahima	bam	abraham	ibrahima	ibraham	**abraham**	**abraham**	**abraham**	ibraham
alejandría	caa	alexandria	alejandria		**alexandria**	**alexandria**	**alexandria**	al
gebeliela	dob	gabriel	gebeliela	gebriel	geberiel	geberiel	geberiel	gebbrla
гедеоне	rus	gideon	gedeone		gedeone	gedeon	gedeone	gahedon
filaistus	mfi	philetus	filaistus	philestus	phlestus	phlestus	phlestus	fylestus
		Avg Accuracy:	.14	.17	.2129	.2134	.2130	.10

Table 2: Example transliterations from each system. Average accuracy is over all languages, not just the ones listed in the table. Correct transliterations are bolded. Note that Sequitur is not guaranteed to transliterate a word.

or post-processing transliteration step using a language-independent transliterator to deal with unknown characters yielded inconclusive (statistically insignificant) results as to whether preprocessing is effective, though we found evidence that post-transliterating can help with unseen characters. Standard methods of system combination slightly boosted performance. In addition, we found that exploiting the multilinguality of the data allows for effective transfer learning in a single neural machine translation model that can act as a universal transliterator. A dataset of names transliterated by each system is available for research purposes. With recent advances in neural models, we believe that approaches leveraging multiple languages are worth exploring in the future.

7. Acknowledgments

This work was supported in part by the DARPA LORELEI program. The findings, conclusions, and opinions found in this work are those of the authors and do not necessarily represent the views of the funding agency.

8. Bibliographical References

Bisani, M. and Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. *Speech communication*, 50(5):434–451.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany, August. Association for Computational Linguistics.

Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Irvine, A., Callison-Burch, C., and Klementiev, A. (2010). Transliterating from all languages. In *Proceedings of the Conference of the Association for Machine Translation in the Americas (AMTA)*.

Jiampojamarn, S., Dwyer, K., Bergsma, S., Bhargava, A., Dou, Q., Kim, M.-Y., and Kondrak, G. (2010). Transliteration generation and mining with limited training resources. In *Proceedings of the 2010 Named Entities Workshop*, pages 39–47. Association for Computational Linguistics.

Kann, K. and Schütze, H. (2016). Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 555–560. Association for Computational Linguistics.

Karimi, S., Scholer, F., and Turpin, A. (2011). Machine transliteration survey. *ACM Computing Surveys (CSUR)*, 43(3):17.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Koehn, P. and Knowles, R. (2017). Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al. (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*, pages 177–180. Association for Computational Linguistics.

Mayhew, S., Christodoulopoulos, C., and Roth, D. (2016). Transliteration in any language with surrogate languages. *arXiv preprint*.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Qian, T., Hollingshead, K., Yoon, S.-y., Kim, K.-y., and Sproat, R. (2010). A python toolkit for universal transliteration. In *LREC*.

Rosca, M. and Breuel, T. (2016). Sequence-to-sequence neural network models for transliteration. *arXiv preprint*.

Schlippe, T., Quaschningk, W., and Schultz, T. (2014). Combining grapheme-to-phoneme converter outputs for enhanced pronunciation generation in low-resource scenarios. In *The 4th Workshop on Spoken Language Tech-*

nologies for Under-resourced Languages, St. Petersburg, Russia. SLTU 2014.

Virga, P. and Khudanpur, S. (2003). Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the ACL 2003 workshop on Multilingual and mixed-language named entity recognition-Volume 15*, pages 57–64. Association for Computational Linguistics.

Wu, W. and Yarowsky, D. (2018). Creating a translation matrix of the Bible's names across 591 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. European Language Resources Association (ELRA).