On Romanization for Model Transfer Between Scripts in Neural Machine Translation

Chantal Amrhein[1] and **Rico Sennrich**[1,2]
[1]Department of Computational Linguistics, University of Zurich
[2]School of Informatics, University of Edinburgh
{amrhein,sennrich}@cl.uzh.ch

Abstract

Transfer learning is a popular strategy to improve the quality of low-resource machine translation. For an optimal transfer of the embedding layer, the child and parent model should share a substantial part of the vocabulary. This is not the case when transferring to languages with a different script. We explore the benefit of romanization in this scenario. Our results show that romanization entails information loss and is thus not always superior to simpler vocabulary transfer methods, but can improve the transfer between related languages with different scripts. We compare two romanization tools and find that they exhibit different degrees of information loss, which affects translation quality. Finally, we extend romanization to the target side, showing that this can be a successful strategy when coupled with a simple deromanization model.

1 Introduction

Neural Machine Translation (NMT) has opened up new opportunities in transfer learning from high-resource to low-resource language pairs (Zoph et al., 2016; Kocmi and Bojar, 2018; Lakew et al., 2018). While transfer learning has shown great promise, the transfer between languages with different scripts brings additional challenges. For a successful transfer of the embedding layer, both the parent and the child model should use the same or a partially overlapping vocabulary (Aji et al., 2020). It is common to merge the two vocabularies by aligning identical subwords and randomly assigning the remaining subwords from the child vocabulary to positions in the parent vocabulary (Lakew et al., 2018, 2019; Kocmi and Bojar, 2020).

This works well for transfer between languages that use the same script, but if the child language is written in an unseen script, most vocabulary positions are replaced by random subwords. This significantly reduces the transfer from the embedding layer. Gheini and May (2019) argue that romanization can improve transfer to languages with unseen scripts. However, romanization can also introduce information loss that might hurt translation quality. In our work, we study the usefulness of romanization for transfer from many-to-many multilingual MT models to low-resource languages with different scripts. Our contributions are the following:

- We show that romanized MT is not generally optimal, but can improve transfer between related languages that use different scripts.
- We study information loss from different romanization tools and its effect on MT quality.
- We demonstrate that romanization on the target side can also be effective when combined with a learned deromanization model.

2 Related Work

Initial work on transfer learning for NMT has assumed that the child language is known in advance and that the parent and child model can use a shared vocabulary (Nguyen and Chiang, 2017; Kocmi and Bojar, 2018). Lakew et al. (2018) argue that this is not feasible in most real-life scenarios and propose using a dynamic vocabulary. Most studies have since opted to replace unused parts of the parent vocabulary with unseen subwords from the child vocabulary (Lakew et al., 2019; Kocmi and Bojar, 2020); others use various methods to align embedding spaces (Gu et al., 2018; Kim et al., 2019). Recently, Aji et al. (2020) showed that transfer of the embedding layer is only beneficial if there is an overlap between the parent and child vocabulary such that embeddings for identical subwords can be aligned. Such alignments are very rare if the child language uses an unseen script.

Gheini and May (2019) train a universal vocabulary on multiple languages by romanizing languages written in a non-Latin script. Their many-to-one parent model can be transferred to new source languages without exchanging the vocabulary. In our work, we extend this idea to many-to-many translation settings using subsequent deromanization of the output. We study the trade-off between a greater vocabulary overlap and information loss as a result of romanization. Based on experiments on a diverse set of low-resource languages, we show that romanization is helpful for model transfer to related languages with different scripts.

3 Romanization

Romanization describes the process of mapping characters in various scripts to Latin script. This mapping is not always reversible. The goal is to approximate the pronunciation of the text in the original script. However, depending on the romanization tool, more or less information encoded in the original script is lost. We compare two tools for mapping our translation input to Latin script:

`uroman`[1] (Hermjakob et al., 2018) is a tool for universal romanization that can romanize almost all character sets. It is unidirectional; mappings from Latin script back to other scripts are not available.

`uconv`[2] is a command-line tool similar to `iconv` that can be used for transliteration. It preserves more information from the original script, which is expressed with diacritics. `uconv` is bi-directional for a limited number of script pairs.

Below is an example of the same Chinese sentence romanized with `uroman` and `uconv`:

她到塔皓湖去了
`uroman`: ta dao ta hao hu qu le
`uconv`: tā dào tǎ hào hú qù le
"She went to Lake Tahoe."

The two tools exhibit different degrees of information loss. `uroman` ignores tonal information and consequently collapses the representations of 塔 (Pinyin *tǎ*; 'tower') and 她 (Pinyin *tā*; 'she'). Romanization with `uconv` retains this distinction but it still adds ambiguity and loses the distinction between 她 (Pinyin *tā*; 'she') and 他 (Pinyin *tā*; 'he'), among others. While `uconv` exhibits less information loss, its use of diacritics limits subword sharing between languages. We measure character-level overlap between English and romanized Arabic, Russian and Chinese with chrF scores (Popović, 2015) and find they are much higher for `uroman` (9.6, 18.8 and 13.3) compared to `uconv` (6.8, 18.1 and 7.2 respectively).

4 Deromanization

Romanization is not necessarily reversible with simple rules due to information loss. Therefore, previous work on romanized machine translation has focused on source-side romanization only (Du and Way, 2017; Wang et al., 2018; Aqlan et al., 2019; Briakou and Carpuat, 2019; Gheini and May, 2019). We argue that romanization can also be applied on the target side, followed by an additional deromanization step. This step can be performed by a character-based Transformer (Vaswani et al., 2017) that takes data romanized with `uroman` or `uconv` as input and is trained to map it back to the original script. We provide more details on our deromanization systems in Appendix A.2.

5 Experimental Setup

5.1 Data

We use OPUS-100 (Zhang et al., 2020)[3], an English-centric dataset that includes parallel data for 100 languages. It provides up to 1 million sentence pairs for every X-EN language pair as well as 2,000 sentence pairs for development and testing each. There is no overlap between any of the data splits across any of the languages, i.e. every English sentence occurs only once.

We pretrain our multilingual models on 5 high-resource languages that cover a range of different scripts $\{AR, DE, FR, RU, ZH\} \leftrightarrow EN$. For our transfer learning experiments, we choose 7 additional languages that are either:

(a) **Not closely related** to any of the pretraining languages and written in an **unseen script**, e.g. Marathi is not related to any of our pretraining languages and written in Devanagari script.

(b) **Closely related** to a pretraining language and written in an **unseen script**, e.g. Yiddish is related to German and written in Hebrew script.

[1] https://github.com/isi-nlp/uroman
[2] https://linux.die.net/man/1/uconv
[3] https://github.com/EdinburghNLP/opus-100-corpus

		script	related to	# sentence pairs
pretraining languages	Arabic (ar)	Arabic	he, mt	1,000,000
	German (de)	Latin	yi	1,000,000
	French (fr)	Latin	-	1,000,000
	Russian (ru)	Cyrillic	sh	1,000,000
	Chinese (zh)	Simpl. Han	-	1,000,000
(a)	Amharic (am)	Ge'ez	-	71,222
	Marathi (mr)	Devanagari	-	21,985
	Tamil (ta)	Tamil	-	198,927
(b)	Hebrew (he)*	Hebrew	ar	50,000
	Yiddish (yi)	Hebrew	de	7,718
(c)	Maltese (mt)*	Latin	ar	100,000
	Serbo-Croatian (sh)*	Latin	ru	98,421

Table 1: Overview of all languages, the script they are written in, other languages in this set they are closely related to (considering lexical similarity) and number of X↔EN sentence pairs. (*) means artificial low-resource settings were created.

(c) Written in **Latin script** but **closely related to a pretraining language in non-Latin script**, e.g. Maltese is related to Arabic and written in Latin script.

Our selection of low-resource languages covers a wide range of language families and training data sizes. Table 1 gives an overview of the selected languages.

5.2 Model Descriptions

We use `nematus`[4] (Sennrich et al., 2017) to train our models and SacreBLEU[5] (Post, 2018) to evaluate them. We compute statistical significance with paired bootstrap resampling (Koehn, 2004) using a significance level of 0.05 (sampling 1,000 times with replacement from our 2,000 test sentences). Our subword vocabularies are computed with byte pair encoding (Sennrich et al., 2016) using the SentencePiece implementation (Kudo and Richardson, 2018). We use a character coverage of 0.9995 to ensure the resulting models do not consist of mostly single characters.

Bilingual Baselines: We follow the recommended setup for low-resource translation in Sennrich and Zhang (2019) to train our bilingual baselines for the low-resource pairs (original script). For our bilingual low-resource models, we use language-specific vocabularies of size 2,000.

[4]https://github.com/EdinburghNLP/nematus

[5]BLEU+case.mixed+lang.XX-XX+numrefs.1 +smooth.exp+tok.13a+version.1.4.2

Pretrained multilingual models: We pretrain three multilingual standard Transformer Base machine translation models (Vaswani et al., 2017): One keeps the original, non-Latin script for Arabic, Russian and Chinese (orig). The others (`uroman` and `uconv`) apply the respective romanization to these parent languages. We follow Johnson et al. (2017) for multilingual training by prepending a target language indicator token to the source input. For our pretrained models, we use a shared vocabulary of size 32,000. An overview of our model hyperparameters is given in Appendix A.1.

Finetuning: We finetune our pretrained models independently for every low-resource language X. For finetuning on a child X↔EN pair, we use the same preprocessing as for the respective parent, i.e. we keep original script, use `uroman`, or use `uconv` for romanization. We reuse 250,000 sentence pairs from the original pretraining data and oversample the X↔EN data for a total of around 650,000 parallel sentences for finetuning. This corresponds roughly to a 3:2 ratio which helps to prevent overfitting. We early stop on the respective X↔EN development set. For finetuning, we use a constant learning rate of 0.001. The remaining hyperparameters are identical to pretraining.

5.3 Vocabulary Transfer

For our transfer baseline without romanization, we merge our bilingual baseline vocabulary with that of the parent model following previous work (Aji et al., 2020; Kocmi and Bojar, 2020). First, we

	orig	uroman	uconv
ar-en	**37.4**	36.3	**37.4**
ru-en	33.3	33.5	**34.1**
zh-en	**39.5**	37.0	**39.2**

Table 2: X→EN BLEU scores (Papineni et al., 2002) of the multilingual pretrained models trained on original scripts (orig), romanized with `uroman` and `uconv`. Best systems (no other being statistically significantly better) marked in bold.

align subwords that occur in both vocabularies. Next, we assign the remaining subwords from the bilingual baseline vocabulary to random unused positions in the parent vocabulary. With `uroman`, we can reuse the parent vocabulary as is. `uconv`, however, may produce unseen diacritics, which can result in a small number of unseen subwords. If that is the case, we perform the same vocabulary replacement for these subwords as for the vocabulary with the original script.

6 Results

6.1 Does Romanization Hurt Translation?

To study the effects of information loss from romanization, we compare the translation quality of our three pretrained multilingual models. To minimize the impact of deromanization, we only discuss X→EN directions for languages with non-Latin scripts. The results are presented in Table 2. Whether romanization hurts the translation quality depends largely on the language pair. For example, for ZH→EN, both romanization tools perform worse than the model trained on original scripts. This is in line with our previous discussion: Even though `uconv` keeps tonal information, there is still more ambiguity compared to using Chinese characters. The model trained with `uconv` romanization consistently outperforms `uroman`. This indicates that it is more important to minimize information loss than to maximize subword sharing.

An additional effect of using romanization, and thus being able to reuse the subword segmentation model during transfer, is that compression rates are worse than for dedicated segmentation models (see Table 3). The resulting longer sequences with potentially suboptimal subword splits may also have a negative influence on translation quality.

	orig	uroman (%)	uconv (%)
ar	67.7	+ 2.2	+ 9.9
de	97.8	- 0.5	- 0.8
fr	131.7	- 0.4	- 0.6
ru	91.5	+ 3.3	- 0.2
zh	54.1	+ 98.9	+ 156.6
am	113.0	+ 70.4	+ 83.1
he	40.3	+ 17.6	+ 20.1
mr	42.4	+ 36.8	+ 35.4
mt	176.5	- 1.9	- 1.4
sh	168.0	- 4.7	- 5.5
ta	138.3	+ 20.1	+ 22.3
yi	54.2	+ 12.4	+ 39.5

Table 3: Average number of subwords per sentence with original script data (orig) and % relative change after romanization (`uroman` and `uconv`). Original script data is segmented with a shared subword segmentation model for {AR,DE,EN,FR,RU,ZH} and language-specific models for low-resource languages. For `uroman` and `uconv`, all languages are segmented using a shared model for {AR,DE,EN,FR,RU,ZH}, romanized with the respective tool.

6.2 Can We Restore the Original Script?

Table 4 compares our character-based Transformers to `uconv`'s built-in, rule-based deromanization. Relying on `uconv`'s built-in deromanization is not optimal. First, it does not support mappings back into all scripts. Second, the performance of built-in `uconv` deromanization varies with the amount of "script code-switching", e.g. due to hyperlinks or email addresses. Character-based Transformers can learn to handle mixed script and outperform `uconv`'s built-in deromanization.

Our models can reconstruct the original script much better from `uconv` data than from `uroman`. This is not surprising considering that `uroman` causes more information loss and ambiguity. As a shallow measure of the ambiguity introduced, we can compare the vocabulary size (before subword segmentation): With romanization, the total number of types in our training sets decreases on average by 10% for `uconv` and by 14% for `uroman`.

Preliminary experiments with artificial low-resource settings (Appendix B.1) showed that additional training data can improve deromanization but it performs well even with very small amounts of training data (10,000 sentences). This shows that our proposed character-based Transformer models are powerful enough to learn a mapping back to

	built-in	learned	
	uconv	uconv	uroman
ar	92.7	**98.1**	94.6
ru	94.9	**99.2**	98.8
zh	-	**96.7**	94.0
am	-	**99.7**	97.8
he	98.7	**99.7**	96.9
mr	79.1	**99.3**	97.6
ta	72.9	**98.3**	98.0
yi	49.6	**96.9**	89.6

Table 4: chrF scores of the deromanization to the original script. Best systems marked in bold.

			transfer from multilingual parent		
		base	orig	uroman	uconv
(a)	am-en	14.4	**16.2**	**16.5**	**16.0**
	en-am	12.7	13.7	6.5	**14.3**
	mr-en	34.3	**45.0**	43.4	42.8
	en-mr	25.7	**33.4**	**33.2**	**33.0**
	ta-en	21.9	**29.3**	**29.0**	**29.2**
	en-ta	13.5	**21.5**	21.0	**22.4**
	avg imp	-	**+ 6.1**	+ 4.5	**+ 5.9**
(b)	yi-en	6.9	22.5	24.9	**28.9**
	en-yi	9.5	12.0	**20.7**	**19.7**
	he-en	22.8	**28.6**	**28.5**	**29.0**
	en-he	21.1	24.5	25.2	**26.6**
	avg imp	-	+ 6.8	+ 9.8	**+ 11.0**
(c)	mt-en	46.5	**59.1**	**59.5**	**59.5**
	en-mt	35.6	**45.0**	**45.2**	**45.3**
	sh-en	40.1	55.5	**56.3**	**56.7**
	en-sh	33.8	52.1	52.3	**53.7**
	avg imp	-	+ 13.9	+ 14.3	**+ 14.8**

Table 5: BLEU scores of the bilingual baselines (no transfer learning) and finetuned models using original scripts (orig), romanized with `uroman` and `uconv`. Average improvement over bilingual baseline is shown per group of languages. Best systems (no other being statistically significantly better) marked in bold.

the original script as much as this is possible, given the increased ambiguity. This finding is supported by concurrent work showing that character-based Transformers are well-suited to a range of string transduction tasks (Wu et al., 2020).

6.3 Transfer to Low-Resource Languages

Table 5 shows the results from our experiments on transfer learning with romanization. Romanizing non-Latin scripts is not always useful. For low-resource languages that use an unseen script but are not related to any of the pretraining languages (a), the performance degrades for `uroman` and is not statistically significantly different for `uconv`. The extremely low BLEU score for EN$\rightarrow$AM shows another problem with `uroman` romanization: `uroman` ignores the Ethiopic word space character which increases the distance between translation and reference.

However, for languages that are related to a pretraining language with a different script (groups (b) and (c)), there is an added benefit of using romanization. The statistically significant improvement of `uconv` over `uroman` strengthens our claim that it is important to keep as much information as possible from the original script when mapping to Latin script. Despite potential information loss from romanization and error propagation from deromanization, our results show that romanization has merit when applied to related languages that can profit from a greater vocabulary overlap.

7 Conclusion

We analyzed the value of romanization for transferring multilingual models to low-resource languages with different scripts. While we cannot recommend romanization as the default strategy for multilingual models and transfer learning across scripts because of the information loss inherent to it, we find that it benefits transfer between related languages that use different scripts. The `uconv` romanization tool outperforms `uroman` because it preserves more information encoded in the original script and consequently causes less information loss. Furthermore, we demonstrated that romanization can also be successful on the target side if followed by an additional, learned deromanization step. We hope that our results provide valuable insights for future work in transfer learning and practical applications for low-resource languages with unseen scripts.

Acknowledgements

We thank our colleagues Anne, Annette, Duygu, Jannis, Mathias, Noëmi and the anonymous reviewers for their helpful feedback. This work was funded by the Swiss National Science Foundation (project MUTAMUR; no. 176727).

References

Alham Fikri Aji, Nikolay Bogoychev, Kenneth Heafield, and Rico Sennrich. 2020. In neural machine translation, what does transfer learning transfer? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7701–7710, Online. Association for Computational Linguistics.

Fares Aqlan, Xiaoping Fan, Abdullah Alqwbani, and Akram Al-Mansoub. 2019. Arabic–Chinese Neural Machine Translation: Romanized Arabic as Subword Unit for Arabic-sourced Translation. *IEEE Access*, 7:133122–133135.

Eleftheria Briakou and Marine Carpuat. 2019. The university of Maryland's Kazakh-English neural machine translation system at WMT19. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 134–140, Florence, Italy. Association for Computational Linguistics.

Jinhua Du and Andy Way. 2017. Pinyin as Subword Unit for Chinese-Sourced Neural Machine Translation. In *Proceedings of the 25th Irish Conference on Artificial Intelligence and Cognitive Science, Dublin, Ireland, December 7 - 8, 2017*, volume 2086 of *CEUR Workshop Proceedings*, pages 89–101. CEUR-WS.org.

Mozhdeh Gheini and Jonathan May. 2019. A universal parent model for low-resource neural machine translation transfer.

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor O.K. Li. 2018. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, New Orleans, Louisiana. Association for Computational Linguistics.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda 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*, 5:339–351.

Yunsu Kim, Yingbo Gao, and Hermann Ney. 2019. Effective cross-lingual transfer of neural machine translation models without shared vocabularies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1246–1257, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Tom Kocmi and Ondřej Bojar. 2018. Trivial transfer learning for low-resource neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 244–252, Belgium, Brussels. Association for Computational Linguistics.

Tom Kocmi and Ondřej Bojar. 2020. Efficiently reusing old models across languages via transfer learning. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 19–28, Lisboa, Portugal. European Association for Machine Translation.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Surafel M Lakew, Aliia Erofeeva, Matteo Negri, Marcello Federico, and Marco Turchi. 2018. Transfer Learning in Multilingual Neural Machine Translation with Dynamic Vocabulary. In *15th International Workshop on Spoken Language Translation (IWSLT)*.

Surafel M Lakew, Alina Karakanta, Marcello Federico, Matteo Negri, and Marco Turchi. 2019. Adapting Multilingual Neural Machine Translation to Unseen Languages. *16th International Workshop on Spoken Language Translation (IWSLT)*.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for neural machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 296–301, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Maja Popović. 2015. chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for 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 *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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Boli Wang, Jinming Hu, Yidong Chen, and Xiaodong Shi. 2018. XMU neural machine translation systems for WAT2018 Myanmar-English translation task. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation: 5th Workshop on Asian Translation: 5th Workshop on Asian Translation*, Hong Kong. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the Transformer to Character-level Transduction.

Biao Zhang, Philip Williams, Ivan Titov, and Rico Sennrich. 2020. Improving massively multilingual neural machine translation and zero-shot translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1628–1639, Online. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A Model Details

A.1 Multilingual Pretrained Models

We train multilingual Transformer Base machine translation models (Vaswani et al., 2017) with 6 encoder layers, 6 decoder layers, 8 heads, an embedding and hidden state dimension of 512 and a feed-forward network dimension of 2048. We regularize our models with a dropout of 0.1 for the embeddings, the residual connections, in the feed-forward sub-layers and for the attention weights. Furthermore, we apply exponential smoothing of 0.0001 and label smoothing of 0.1. We tie both our encoder and decoder input embeddings as well as the decoder input and output embeddings (Press and Wolf, 2017). All of our multilingual machine translation models are trained with a maximum token length of 200 and a vocabulary of size 32,000.

For optimization, we use Adam (Kingma and Ba, 2015) with standard hyperparameters and a learning rate of 0.0001. We follow the Transformer learning schedule described in (Vaswani et al., 2017) with a linear warmup over 4,000 steps. Our token batch size is set to 16,348 and we train on 4 NVIDIA Tesla V100 GPUs. All models were trained using the implementation provided in `nematus` (Sennrich et al., 2017) using early stopping on a development set with patience 5.

A.2 Character-Based Deromanization

We train character-based Transformer Base machine translation models (Vaswani et al., 2017). To achieve character-level deromanization, we do not make any changes to the architecture. We simply change the input format such that every character is separated by spaces. The original space characters are replaced by another character that does not occur in the training data (∅). The following example shows the parallel training data for learned deromanization:

`uroman` source: C H t o ∅ t a m ∅ d a l s h e ?
`uconv` source: Č t o ∅ t a m ∅ d a l ' š e ?
target: Ч т о ∅ т а м ∅ д а л ь ш е ?
"What's next?"

We use a maximum sequence length of 1,200 since character-level sequences are much longer than subword-level sequences. Our vocabularies are made up of all characters that occur in the respective training data. All other parameters are set as for multilingual pretraining described in Appendix A.1.

B Supplementary Results

B.1 Effect of Data Size on Deromanization

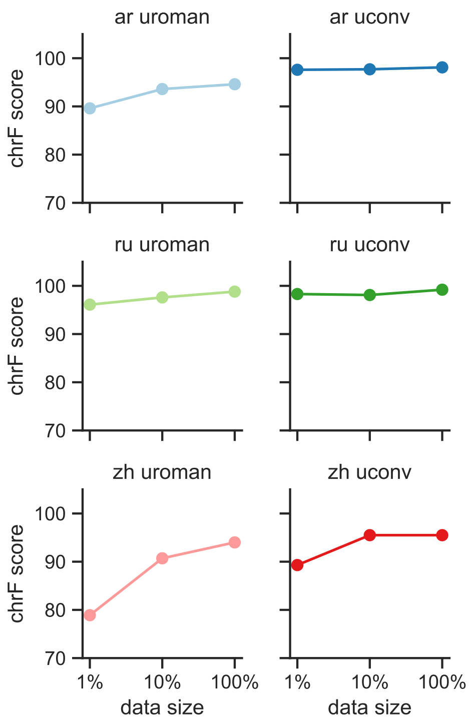

Figure 1: chrF scores of deromanization models trained on 1%, 10% and 100% of the total data (corresponding to 10,000, 100,000 and 1,000,000 parallel sentences). Results compare romanization with `uroman` and `uconv` for Arabic, Russian and Chinese.

Figure 1 shows the influence of the training data size on the chrF score between the deromanized test set and the original script test set. Additional data can improve deromanization models, especially for languages such as Chinese, where a mapping back to the original script is difficult to learn due to the information loss from romanization.

We analyze how deromanization quality affects the BLEU score of deromanized translations. This is shown in Table 6. We find that the deromanization models for `uroman` are more affected by an extreme low-resource setting. For `uconv`, deromanization models trained on smaller data sets show less performance loss compared to using full data. It is notable that training `uconv` deromanization models only on 100,000 sentences has almost no effect on the BLEU score for EN→AR and EN→ZH. For EN→RU, there is a loss of 1.1 BLEU points compared to training on 100% of the data. Looking at the deromanization outputs for EN→RU, we found that deromanization models trained on less data could not handle "script code-

	uroman		
	1%	10%	100%
en-ar	20.3	**21.6**	**21.7**
en-ru	26.5	27.9	**28.5**
en-zh	38.8	**41.6**	**41.9**

	uconv		
	1%	10%	100%
en-ar	21.2	**21.8**	**21.7**
en-ru	27.9	28.2	**29.3**
en-zh	40.2	**41.8**	**41.9**

Table 6: EN→X BLEU scores of the multilingual pre-trained models after deromanization. Deromanization models were trained on 1%, 10% and 100% of the total data (corresponding to 10,000, 100,000 and 1,000,000). Best systems (no other being statistically significantly better) marked in bold.

switching" as well as the models trained on full data.

While these results show that additional training material can improve deromanization, they do not mean that romanization on the target side cannot be used in low-resource machine translation settings. First, our results in Section 6.3 have shown that romanization on the target side can bring improvements even if deromanization models cannot perfectly reconstruct the original script. Second, it will often be possible to find additional monolingual data to improve deromanization models.