Reusing a Pretrained Language Model on Languages with Limited Corpora for Unsupervised NMT

Alexandra Chronopoulou, Dario Stojanovski, Alexander Fraser

Center for Information and Language Processing, LMU Munich, Germany
{achron, stojanovski, fraser}@cis.lmu.de

Abstract

Using a language model (LM) pretrained on two languages with large monolingual data in order to initialize an unsupervised neural machine translation (UNMT) system yields state-of-the-art results. When limited data is available for one language, however, this method leads to poor translations. We present an effective approach that reuses an LM that is pretrained only on a high-resource language. The monolingual LM is fine-tuned on both languages and is then used to initialize a UNMT model. To reuse the pretrained LM, we have to modify its predefined vocabulary, to account for the new language. We therefore propose a novel vocabulary extension method. Our approach, RE-LM, outperforms a competitive cross-lingual pretraining model (XLM) in English-Macedonian (En-Mk) and English-Albanian (En-Sq), yielding more than $+8.3$ BLEU points for all four translation directions.

1 Introduction

Neural machine translation (NMT) has recently achieved remarkable results (Bahdanau et al., 2015; Vaswani et al., 2017), based on the exploitation of large parallel training corpora. Such corpora are only available for a limited number of languages. UNMT has attempted to address this limitation by training NMT systems using monolingual data *only* (Artetxe et al., 2018; Lample et al., 2018). Top performance is achieved using a bilingual masked language model (Devlin et al., 2019) to initialize a UNMT encoder-decoder system (Lample and Conneau, 2019). The model is then trained using denoising auto-encoding (Vincent et al., 2008) and back-translation (Sennrich et al., 2016a). The approach was mainly evaluated by translating between high-resource languages.

Translating between a high-resource and a low-resource language is a more challenging task. In this setting, the UNMT model can be initialized with a pretrained cross-lingual LM. However, training this UNMT model has been shown to be ineffective when the two languages are not related (Guzmán et al., 2019). Moreover, in order to use a pretrained cross-lingual LM to initialize a UNMT model, the two models must have a shared vocabulary. Thus, a bilingual LM needs to be trained from scratch for each language pair, before being transferred to the UNMT model (e.g. En-De LM for En-De UNMT).

Motivated by these issues, we focus on the question: *how can we accurately and efficiently translate between a high-monolingual-resource (***HMR***) and a low-monolingual-resource (***LMR***) language*? To address this question, we adapt a monolingual LM, pretrained on an HMR language to an LMR language, in order to initialize a UNMT system.

We make the following contributions: (1) We propose REused-LM[1] (**RE-LM**), an effective transfer learning method for UNMT. Our method reuses a pretrained LM on an HMR language, by fine-tuning it on both LMR and HMR languages. The fine-tuned LM is used to initialize a UNMT system that translates the LMR to the HMR language (and vice versa). (2) We introduce a novel vocabulary extension method, which allows fine-tuning a pretrained LM to an unseen language. (3) We show that RE-LM outperforms a competitive transfer learning method (XLM) for UNMT on three language pairs: English-German (En-De) on a synthetic setup, En-Mk and En-Sq. (4) We show that RE-LM is effective in low-resource supervised NMT. (5) We conduct an analysis of fine-tuning schemes for RE-LM and find that including adapters (Houlsby et al., 2019) in the training procedure yields almost the same UNMT results as RE-LM at a lower computational price. We also run experiments to identify the contribution of the vocabulary extension method.

[1]We release the code in https://github.com/alexandra-chron/relm_unmt.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2703–2711,
November 16–20, 2020.

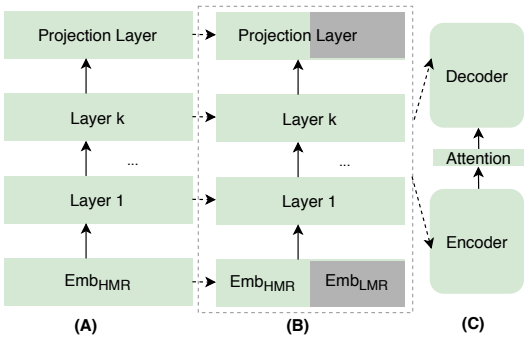

Figure 1: **RE-LM**. **(A)** LM pretraining. **(B)** Fine-tuning. The embedding and the projection layer are extended using §3.2 (dark gray) and **(C)** Transfer to an NMT system. Dashed arrows indicate transfer of weights.

2 Related Work

Transfer learning for UNMT. The field of UNMT has recently experienced tremendous progress. Artetxe et al. (2018); Lample et al. (2018) train UNMT models with monolingual data only, using denoising auto-encoding (Vincent et al., 2008) and online back-translation (Sennrich et al., 2016a) as training objectives. This approach is successful for languages with high-quality, large, comparable data. When these conditions are not met, though, UNMT provides near-zero scores (Neubig and Hu, 2018). UNMT is further improved when initialized with a cross-lingual pretrained model, trained on large corpora (Lample and Conneau, 2019; Song et al., 2019). However, many languages have only limited monolingual data available, a setting where UNMT is not effective (Guzmán et al., 2019). Sun et al. (2020), whose work is close to our work in motivation, train a UNMT model for an HMR-LMR language pair. Iteratively, every subset (e.g. 10%) of HMR and all LMR data is backtranslated and the pseudo-parallel corpus is added to the training process. Just like XLM, this training procedure needs to run from scratch for every new language pair. By contrast, our method fine-tunes a monolingual pretrained LM for UNMT, so it is computationally faster and simpler.

Vocabulary. Transferring a pretrained model (*source*) to a new model (*target*) requires the use of a shared vocabulary (Nguyen and Chiang, 2017). Kim et al. (2019) propose a linear alignment of the source and target model embeddings using an unsupervised dictionary. However, when the embeddings of the two models do not have enough overlapping strings, dictionary induction might fail (Søgaard et al., 2018). Lakew et al. (2018) transfer a source NMT model to a target NMT model (e.g. De-En to Nl-En). To enable transfer, they overwrite the source vocabulary with the target vocabulary. By contrast, we keep the union of the two vocabularies. We fine-tune a pretrained monolingual LM to an LMR language, to initialize an NMT model. Thus, we need the vocabularies of both languages.

Adapters. Residual adapters (Houlsby et al., 2019) are feed-forward networks, added to each of to the original model's layers. During fine-tuning, the model parameters are frozen and only the adapters are fine-tuned. This can prevent catastrophic forgetting (Goodfellow et al., 2014; Bapna and Firat, 2019). Adapters show promising results in domain adaptation (Bapna and Firat, 2019) and cross-lingual classification (Artetxe et al., 2020). Motivated by this, we study the use of adapters during LM fine-tuning in our analysis.

3 Proposed Approach

We describe our method for translation between a high-resource (HMR) and a low-resource language (LMR) using monolingual data in this section.

3.1 RE-LM

Our proposed approach consists of three steps, as shown in Figure 1:

(A) We train a monolingual masked LM on the HMR language, using all available HMR corpora. This step needs to be performed only *once* for the HMR language. Note that a publicly available pretrained model could also be used.

(B) To fine-tune the pretrained LM on the LMR language, we first need to overcome the vocabulary mismatch problem. Fine-tuning without extending the vocabulary is detrimental, as we will show later in the analysis. We therefore extend the vocabulary of the pretrained model using our proposed method, described in §3.2.

(C) Finally, we initialize an encoder-decoder UNMT system with the fine-tuned LM. The UNMT model is trained using denoising auto-encoding and online back-translation for the HMR-LMR language pair.

BPE$_{HMR}$	Pro_gram_et e fes_ti_val_it p_ë_r_f_shi_j_n_ë nj_ë rang t_ë g_jer_ë v_ep_rim_tar_ish
BPEjoint	Progra_met e fe_s_ti_val_it përfshijnë një rang të gjerë veprimtari_sh

Figure 2: Segmentations of Albanian (Sq). We observe that splitting Sq using En BPEs (BPE$_{HMR}$) results in heavily segmented tokens. This problem is alleviated using BPE$_{joint}$ tokens, learned on both languages.

3.2 Vocabulary Extension

We propose a novel method that enables adapting a pretrained monolingual LM to an unseen language. We consider the case of an LM pretrained on an HMR language. The training data is split using Byte-Pair-Encoding (BPE) (Sennrich et al., 2016b). We denote these BPE tokens as BPE_{HMR} and the resulting vocabulary as V_{HMR}. We aim to fine-tune the trained LM to an unseen LMR language. Splitting the LMR language with BPE_{HMR} tokens would result in heavy segmentation of LMR words (Figure 2). To counter this, we learn BPEs on the joint LMR and HMR corpus (BPE_{joint}). We then use BPE_{joint} tokens to split the LMR data, resulting in a vocabulary V_{LMR}. This technique increases the number of shared tokens and enables cross-lingual transfer of the pretrained LM. The final vocabulary is the union of the V_{HMR} and V_{LMR} vocabularies. We extend the input and output embedding layer to account for the new vocabulary items. The new parameters are then learned during fine-tuning.

4 Experimental Setup

Datasets. We experiment with two setups. In the first *synthetic* setup we use En-De. We sample 8M En sentences from NewsCrawl. To simulate an LMR language, we gradually sample 0.05M, 0.5M and 1M De sentences. We use the WMT dev/test sets (Bojar et al., 2016). The second, *real-world setup* is En-Mk, En-Sq. We use 68M En sentences from NewsCrawl. For Mk and Sq, we use 2.4M Mk and 4M Sq, obtained from OSCAR[2] (Ortiz Suárez et al., 2019) and Wikipedia. We randomly select 3K sentences from SETIMES[3] as dev and 3K as test set. We tokenize data with standard Moses (Koehn et al., 2006) scripts. For the low-resource supervised case, we sample 10K, 100K, and 200K parallel sentences from SETIMES for Mk and Sq.
Preprocessing. We train a standard XLM model (Lample and Conneau, 2019) as a baseline using 32K BPE merge operations, learned on the concatenation of sentences sampled randomly from the corpora of each language pair with $\alpha = 0.5$. For RE-LM, we learn 32K BPEs on the HMR corpus and extract the initial vocabulary (V_{HMR}). Then, we learn 32K BPEs on the joint LMR and HMR corpus (BPE_{joint}). We extend the initial V_{HMR} vocabulary by the amount of LMR vocabulary items that are not already present in V_{HMR}. To identify whether a smaller number of BPE merges would be useful for splitting the LMR language, we conduct experiments varying their number in the analysis.
Model Configuration. RE-LM is built using the XLM codebase[4]. Each masked LM has a Transformer architecture with 1024 hidden units, 6 layers and 8 attention heads. Each NMT model is a 6-layer encoder-decoder Transformer with 1024 hidden units and 8 heads. Each LM is trained using Adam (Kingma and Ba, 2015) with learning rate 10^{-4} and masking follows Devlin et al. (2019). During UNMT and supervised NMT training, Adam with inverse square root scheduling and a learning rate of 10^{-4} is used. We evaluate NMT models on the dev set every 3000 updates using greedy decoding. The En LM and each XLM are trained on 8 NVIDIA GTX 11 GB GPUs for 1 week, with a per-GPU batch size of 32. LM fine-tuning and NMT training models are computationally efficient, using just 1 GPU and 32 batch size. We assume that by fine-tuning the LM on 8 GPUs, we could get even better results. Final translations are generated using beam search of size 5. We report de-tokenized BLEU using SacreBLEU (Post, 2018)[5].
Experiments. For unsupervised translation, we train a randomly initialized UNMT model for each language pair as a first baseline. As a transfer learning baseline, we use XLM (Lample and Conneau, 2019), trained on the two languages and transferred to a UNMT model. The UNMT models are trained using monolingual data. For supervised translation, NMT training is performed using only parallel corpora, without offline back-translation of monolingual data. The first baseline is a randomly initialized NMT system. The second baseline is an NMT model initialized with XLM. We compare them to our proposed approach, RE-LM. Both XLM and RE-LM are trained on the monolingual corpora of both languages of interest. In the analysis, we add adapters (Rebuffi et al., 2018) of hidden size 256 after each self-attention and each feed-forward layer of the pretrained monolingual LM. We freeze the parameters of the pretrained LM and fine-tune only the adapters and the embedding layer.

5 Results and Analysis

5.1 Unsupervised Translation

Table 1 presents our UNMT results, comparing random initialization, XLM and RE-LM.

[2] https://oscar-corpus.com/
[3] http://opus.nlpl.eu/SETIMES.php
[4] github.com/facebookresearch/XLM/
[5] Signature "BLEU+c.mixed+#.1+s.exp+tok.13a+v.1.4.9"

HMR-LMR language pair size of LMR language	**En-De** **0.05M**		**En-De** **0.5M**		**En-De** **1M**		**En-Mk** **2.4M**		**En-Sq** **4M**	
	←	→	←	→	←	→	←	→	←	→
random	3.9	4.9	3.4	2.6	4.2	4.1	3.5	3.0	6.6	5.6
XLM	8.1	6.4	19.8	16.0	21.7	18.1	12.2	12.8	16.3	18.8
RE-LM	**10.7**	**7.5**	**22.6**	**19.0**	**24.3**	**21.9**	**22.0**	**21.1**	**27.6**	**28.1**

Table 1: UNMT BLEU scores. The first column indicates the pretraining method used. Left arrow (←) refers to translation from the LMR language to En, while right arrow (→) refers to translation from En to the LMR language.

Synthetic setup. We observe that RE-LM consistently outperforms XLM. Using 50K De sentences, RE-LM has small gains over XLM (+1.1 BLEU in En→De). However, when we scale to slightly more data (500K), the performance of RE-LM is clearly better than the one of XLM, with +3 En→De BLEU gains. With 1M De data, our model surpasses the XLM by more than 2.6 BLEU in both directions.

Real-world setup. Our approach surpasses XLM in both language pairs. We observe that RE-LM achieves at least +8.3 BLEU over XLM for En-Mk. Our model was first pretrained on En and then fine-tuned on both En and Mk. Therefore, it has processed *all* En and Mk sentences, obtaining a good cross-lingual representation. However, XLM is jointly trained on En and Mk. As a result, it overfits Mk before processing all En data. RE-LM is similarly effective for En-Sq, achieving an improvement of at least +9.3 BLEU over XLM.

Synthetic vs Real-world setup. The effectiveness of RE-LM is pronounced in the real-world setup. We identify two potential reasons. First, for En-De, 8M En is used for LM pretraining, while for En-Mk and En-Sq, 68M En is used. When XLM is trained on imbalanced HMR-LMR data, it overfits the LMR language. This is more evident for the En-Mk (or En-Sq) than for the En-De XLM, perhaps due to the larger data imbalance. Second, in En-De, we use high-quality corpora for both languages (NewsCrawl), whereas Mk and Sq are trained on low-quality CommonCrawl data. The fact that RE-MLM outperforms XLM for Mk and Sq shows that it is more robust to noisy data than the XLM.

5.2 Low-Resource Supervised Translation

We sample 10K, 100K and 200K of En-Mk and En-Sq bi-text and train supervised NMT systems. We compare XLM, RE-LM and *random*, an NMT model trained from scratch. We observe (Table 2) that RE-LM consistently outperforms the baselines when trained on 100K or less for En-Mk and En-Sq. Using 200K, though, RE-LM yields the same results as XLM. We hypothesize that this happens because SETIMES is a homogeneous domain. Thus, training an NMT model with 200K is sufficient for competitive results, so both pretraining models provide similar improvements over *random*.

parallel	languages direction	En-Mk		En-Sq	
		←	→	←	→
10K	random	23.4	23.7	25.5	18.9
	XLM	38.7	38.7	44.7	41.4
	RE-LM	**40.1**	38.9	**45.7**	**42.8**
100K	random	48.4	48.2	51.8	37.4
	XLM	53.7	53.2	57.1	52.0
	RE-LM	**54.8**	53.4	**58.1**	**52.9**
200K	random	51.3	51.2	55.6	51.4
	XLM	55.0	55.5	60.9	55.1
	RE-LM	55.2	55.3	61.1	54.8

Table 2: BLEU scores on the dev set using increasing amounts of parallel data. We show in bold the models that achieve at least +1 BLEU compared to XLM.

5.3 Analysis

We experiment with different fine-tuning schemes and show results in Table 3. Then, we vary the number of BPE merges used to split the LMR language using the vocabulary extension method and also show experiments where this method is not used at all. The results are presented in Table 4.

RE-LM. In Table 3, we compare fine-tuning an LM *only* on the LMR language to fine-tuning it on *both* the HMR and LMR language (rows 1 and 2). Fine-tuning only on the LMR language provides worse BLEU scores because of catastrophic forgetting. The negative effect is clear for Mk and Sq, where fine-tuning only on the LMR results in worse BLEU scores than random initialization, shown in Table 1. For De, the effect is smaller, perhaps because En and De are very similar languages.

Adapters. We insert adapters to the pretrained LM and fine-tune only the adapter and embedding layer. We use the fine-tuned LM to initialize a UNMT system. Adapters are used for both translation directions during UNMT training. Results are presented in Table 3. Fine-tuning the LM only on the LMR language yields at least +3.9 BLEU for En-Sq com-

	HMR-LMR language pair size of LMR language	**En-De 0.05M** ←	→	**En-De 0.5M** ←	→	**En-De 1M** ←	→	**En-Mk 2.4M** ←	→	**En-Sq 4M** ←	→
LM	ft on LMR	9.4	7.3	20.4	16.8	20.6	17.8	2.7	2.4	4.7	4.7
	ft on LMR & HMR (**RE-LM**)	**10.7**	**7.5**	**22.6**	**19.0**	**24.3**	**21.9**	**22.0**	**21.1**	27.6	28.1
	+ adapters ft on LMR (**adapter RE-LM**)	9.8	**7.5**	21.3	18.3	23.7	20.0	21.6	19.0	**30.2**	**29.4**
	+ adapters ft on LMR & HMR	9.2	7.1	20.6	18.0	23.4	19.9	21.6	20.3	24.6	25.5

Table 3: Comparison of UNMT BLEU scores obtained using different fine-tuning schemes of the pretrained monolingual LM. *LM* refers to the pretrained LM (on HMR data), while *ft* refers to fine-tuning.

pared to fine-tuning on both (rows 3, 4). En and Sq are not similar languages and their embeddings also differ. Thus, fine-tuning on both is not helpful. By contrast, fine-tuning only on Sq preserves the pretrained model's knowledge, while adapters are trained to encode Sq. For En-De and En-Mk, both approaches provide similar results. En and Mk do not share an alphabet, so their embeddings do not overlap and both fine-tuning methods are equally effective. In En-De, fine-tuning only on De is marginally better than fine-tuning on both. We highlight that adapters allow parameter-efficient fine-tuning. Adapter RE-LM reaches almost the same results as RE-LM, using just a fraction of the RE-LM parameters while fine-tuning. Details can be found in the appendix.

BPE_{joint} merges	**En-De 0.5M** →	←	**En-Mk 2.4M** →	←	**En-Sq 4M** →	←
-	8.1	8.0	6.1	6.4	7.2	7.6
8K	8.3	10.2	14.3	17.3	18.1	16.4
16K	8.7	14.6	14.9	20.2	27.1	25.5
32K	22.6	19.0	22.0	21.1	27.6	28.1

Table 4: UNMT BLEU scores obtained with RE-LM, **with** (rows 2-4) and **without** (row 1) extending the vocabulary of the pretrained LM (V_{HMR}). When extending the vocabulary, we vary the number of BPE_{joint} merges used to split the LMR data. We note that 32K BPEs are used to split the HMR data (BPE_{HMR}).

BPE_{joint} merges	new vocabulary items Mk	Sq	De
8K	5K	5K	0.6K
16K	10K	10K	2K
32K	19K	20K	19K

Table 5: Statistics of the vocabulary extension method. We split the LMR corpus using 8K, 16K, or 32K BPE merges and report the number of new vocabulary items.

Vocabulary Extension. In order to use RE-LM, we extend the vocabulary of each language, as described in §3.2. The intuition is that, since the pretrained monolingual LM uses BPEs learned exclusively on the HMR language, these BPEs would not split the LMR corpus in a meaningful way. We conduct experiments to clarify the contribution of the vocabulary extension, presented in Table 4. In Table 5, we present the amount of vocabulary items added for each of our experimental setups.

Without vocabulary extension, the results are poor. This is expected, as in the case of Mk for example, the HMR language (En) uses Latin alphabet, whereas Mk uses Cyrillic. If the vocabulary of Mk is not taken into account, the UNMT model cannot provide accurate results. The same applies for Sq and De. We hypothesize that, even though these languages use Latin script, a lot of their words do not appear in En, therefore extending the initial vocabulary to include them is crucial. Using vocabulary extension, we experiment with learning 8K, 16K or 32K BPEs on the joint corpus. We then use them to split the LMR data. We observe in Table 4 that even using only 8K BPEs, there is a large improvement in Mk and Sq (more than +8 BLEU). For En-De, the improvement is negligible. This might be the case because, as Table 5 shows, using 8K merges, only 600 items are added to the initial vocabulary, which are not sufficient for representing De language. This setup for En-De is in fact very similar to not employing vocabulary extension. We notice that adding more vocabulary items (using more BPE merge operations) is helpful for all language pairs, providing improved BLEU scores.

6 Conclusions

Training competitive unsupervised NMT models for HMR-LMR scenarios is important for many real low-resource languages. We proposed RE-LM, a novel approach that fine-tunes a high-resource LM on a low-resource language and initializes an NMT model. RE-LM outperformed a strong baseline in UNMT, while also improving translations on a low-resource supervised setup. In future work, we will apply our method to languages with corpora from diverse domains and also to other languages.

Acknowledgments

This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement № 640550). This work was also supported by DFG (grant FR 2829/4-1). We thank Katerina Margatina and Giorgos Vernikos for their valuable comments and help with the first draft of this paper.

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018. Unsupervised neural machine translation. In *International Conference on Learning Representations*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

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

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 1538–1548.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the conference on machine translation. In *Proceedings of the Conference on Machine Translation*, pages 131–198.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. 2014. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations*.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 6100–6113.

Dan Hendrycks and Kevin Gimpel. 2017. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of the International Conference on Machine Learning*.

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

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Philipp Koehn, Marcello Federico, Wade Shen, Nicola Bertoldi, Ondrej Bojar, Chris Callison-Burch, Brooke Cowan, Chris Dyer, Hieu Hoang, Richard Zens, et al. 2006. Open source toolkit for statistical machine translation: Factored translation models and confusion network decoding. In *Final Report of the 2006 JHU Summer Workshop*.

Surafel Melaku Lakew, Aliia Erofeeva, Matteo Negri, Marcello Federico, and Marco Turchi. 2018. Transfer learning in multilingual neural machine translation with dynamic vocabulary. In *International Workshop on Spoken Language Translation*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, page 7057–7067.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 875–880.

Toan Q. Nguyen and David Chiang. 2017. Transfer learning across low-resource, related languages for

neural machine translation. In *Proceedings of the International Joint Conference on Natural Language Processing*, pages 296–301.

Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. 2019. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In *Workshop on the Challenges in the Management of Large Corpora*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

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

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 157–163.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2018. Efficient parametrization of multi-domain deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Anders Søgaard, Sebastian Ruder, and Ivan Vulić. 2018. On the limitations of unsupervised bilingual dictionary induction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 778–788.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. Mass: Masked sequence to sequence pre-training for language generation. In *Proceedings of the International Conference on Machine learning*, pages 5926–5936.

Haipeng Sun, Rui Wang, Kehai Chen, Masao Utiyama, Eiichiro Sumita, and Tiejun Zhao. 2020. Self-training for unsupervised neural machine translation in unbalanced training data scenarios. *arXiv preprint arXiv:2004.04507*.

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 *Advances in Neural Information Processing Systems*, page 5998–6008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 1096–1103.

A Appendix

A.1 Vocabulary Extension

We provide more examples of different segmentations of Sq, De and Mk using either the BPE_{HMR} or the BPE_{joint} tokens in Figure 3. We observe that, as expected, the Mk sentence is split to the character level, as it uses a different alphabet (Cyrillic) than the one that the BPE_{HMR} tokens were learned on (Latin).

BPE_{hmr}	Pro_gram_et e fes_ti_val_it p_ë_r_f_shi_j_n_ë nj_ë rang_t_ë g_jer_ë v_ep_rim_tar_ish
BPEjoint	Progra_met e fe_s_ti_val_it përfshijnë një rang të gjerë veprimtari_sh

BPE_{hmr}	S_ie hab_en ein ein_z_ig_arti_ges Pro_j_ek_t real_is_ier_t
BPEjoint	Sie haben ein einzig_artiges Projekt realisiert

BPE_{hmr}	П_р_о_е_к_т_о_т б_е_ш_е о_д_о_б_р_е_н о_д в_п_а_д_а_т_а в_о м_а_ј
BPEjoint	Проектот беше одоб_рен од впадата во мај

Figure 3: Segmentation of Sq, De and Mk using BPE_{HMR} or BPE_{joint} tokens. Using BPE_{HMR} tokens results in heavily split words.

A.2 Datasets

We report that we remove sentences longer than 100 words after BPE splitting. We split the data using the fastBPE codebase[6].

A.3 Model Configuration

We tie the embedding and output (projection) layers of both LM and NMT models (Press and Wolf, 2017). We use a dropout rate of 0.1 and GELU activations (Hendrycks and Gimpel, 2017). We use the default parameters of Lample and Conneau (2019) in order to train our models unless otherwise specified. We do not tune the hyperparameters. The code was built with PyTorch (Paszke et al., 2019) on top of the XLM implementation[7]. This code was used for LM pretraining, LM fine-tuning, UNMT training, and NMT training.

LM configuration and training details. RE-LM approach pretrains a **monolingual** language model whereas the XLM approach pretrains a **bilingual** language model. We obtain a checkpoint every 200K sentences processed by the model. We train each LM using as criterion the validation perplexity on the LMR language, with a patience of 10.

The training details of the two *pretraining* methods are presented here:

- The monolingual LM pretraining required 1 week, 8 GPUs and had 137M parameters.
- The XLM pretraining required 1 week, in 8 GPUs. The total number of trainable parameters is 138M.

Our approach also requires an *LM fine-tuning* step. The runtimes, parameters and GPU details are shown in Table 6 under RE-LM *ft* column. The runtimes mentioned refer to the En-Mk language pair. We note that the *LM fine-tuning* step is a lot faster than performing *XLM pretraining* for each language pair (note that pretraining ran on 8 GPUs, whereas fine-tuning on a single GPU).

NMT configuration and training details. The parameters and runtimes of the UNMT models we used are shown in Table 6 under UNMT columns. Likewise, the details of supervised NMT models are shown under sup NMT columns. We get a checkpoint every 50K sentences processed by the model. Regarding the adapter RE-LM training procedure, we note that, different from Houlsby et al. (2019); Bapna and Firat (2019), we also freeze the layer normalization (Ba et al., 2016) parameters, without introducing new ones.

A.4 Validation Scores of Results

In Tables 7 and 8 we show the dev scores of the main results of our proposed approach (RE-LM) compared to the baselines. These Tables extend Table 1 of the main paper.

In Tables 9 and 10, we show the dev scores of the extra fine-tuning experiments we did for the analysis. The Tables correspond to Table 3 of the main paper.

We note that the dev scores are obtained using greedy decoding, while the test scores are obtained with beam search of size 5. We clarify that we train each NMT model using as training criterion the validation BLEU score of the LMR→HMR direction, with a patience of 10. We specifically use `multi-bleu.perl` script from Moses.

[6] https://github.com/glample/fastBPE
[7] https://github.com/facebookresearch/XLM/

	XLM		RE-LM			adapter RE-LM		random	
	UNMT	sup NMT	ft	UNMT	sup NMT	ft	UNMT	UNMT	sup NMT
params	223M	223M	156M	258M	258M	88M	270M	258M	258M
runtime	48h	10h	60h	72h	10h	44h	20h	18h	15h

Table 6: Parameters and training runtimes used for each experiment. We note that each of the experiments ran on a single GPU. *ft* refers to the fine-tuning of the pretrained monolingual LM. Adapter RE-LM refers to the addition of adapters to the LM and the UNMT model.

languages	En-De											
size of LMR	0.05M				0.5M				1M			
	←		→		←		→		←		→	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
random	3.2	3.9	4.1	4.9	2.5	3.4	2.3	2.6	3.7	4.2	3.5	4.1
XLM	5.6	8.1	4.8	6.4	14.5	19.8	12.0	16.0	17.4	21.7	14.6	18.1
RE-LM	7.4	10.7	4.1	7.5	16.2	22.6	13.8	19.0	17.8	24.3	16.3	21.9

Table 7: Unsupervised NMT results with dev scores. The first column indicates the pretraining method used. *Random* refers to random initialization, while XLM refers to the method of Lample and Conneau (2019) and RE-LM to our proposed approach.

size of LMR	2.4M				4M			
	Mk→En		En→Mk		Sq→En		En→Sq	
	dev	test	dev	test	dev	test	dev	test
random	3.1	3.5	3.0	3.0	5.8	6.6	5.6	5.6
XLM	11.8	12.2	12.6	12.8	15.5	16.3	17.3	18.8
RE-LM	22.0	22.0	19.5	21.1	27.2	27.6	27.6	28.1

Table 8: Unsupervised NMT BLEU scores with corresponding dev scores for En-Mk, En-Sq.

	languages	En-De											
	size of LMR	0.05M				0.5M				1M			
		←		→		←		→		←		→	
		dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
LM	ft LMR	6.8	9.4	5.2	7.3	15.1	20.4	12.9	16.8	15.3	20.6	13.3	17.8
	ft both (**RE-LM**)	7.4	10.7	4.1	7.5	16.2	22.6	13.8	19.0	17.8	24.3	16.3	21.9
	+ adapter RE-LM	6.8	9.8	4.8	7.5	15.1	21.3	13.4	18.3	16.9	23.7	15.2	20.0
	+ adapters ft both	6.7	9.2	4.1	7.1	14.8	20.6	13.0	18.0	17.1	23.4	15.0	19.9

Table 9: Comparison of UNMT BLEU scores obtained using different fine-tuning schemes of the pretrained monolingual LM with corresponding dev scores for En-De. *LM* refers to the pretrained LM, trained on HMR data, while *ft* refers to fine-tuning. *ft both* means fine-tuning on the LMR and the HMR language.

	size of LMR	2.4M				4M			
		Mk→En		En→Mk		Sq→En		En→Sq	
		dev	test	dev	test	dev	test	dev	test
LM	ft LMR	2.6	2.7	2.3	2.4	4.4	4.7	4.2	4.7
	ft both (**RE-LM**)	22.0	22.0	19.5	21.1	27.2	27.6	27.6	28.1
	+ adapter RE-LM	21.4	21.6	20.0	19.0	29.8	30.2	29.3	29.4
	+ adapters ft both	22.7	21.6	22.2	20.3	24.4	24.6	25.4	25.5

Table 10: Comparison of UNMT BLEU scores obtained using different fine-tuning schemes of the pretrained monolingual LM with corresponding dev scores for En-Mk and En-Sq. *LM* refers to the pretrained LM, trained on HMR data, while *ft* refers to fine-tuning. *ft both* means fine-tuning on the LMR and the HMR language.