Improving Target-side Lexical Transfer in Multilingual Neural Machine Translation

Luyu Gao, Xinyi Wang, and Graham Neubig
Language Technologies Institute, Carnegie Mellon University
{luyug,xinyiw1,gneubig}@cs.cmu.edu

Abstract

To improve the performance of Neural Machine Translation (NMT) for low-resource languages (LRL), one effective strategy is to leverage parallel data from a related high-resource language (HRL). However, multilingual data has been found more beneficial for NMT models that translate from the LRL to a target language than the ones that translate into the LRLs. In this paper, we aim to improve the effectiveness of multilingual transfer for NMT models that translate *into* the LRL, by designing a better decoder word embedding. Extending upon a general-purpose multilingual encoding method Soft Decoupled Encoding (Wang et al., 2019), we propose DecSDE, an efficient character n-gram based embedding specifically designed for the NMT decoder. Our experiments show that DecSDE leads to consistent gains of up to 1.8 BLEU on translation from English to four different languages.[1]

1 Introduction

The performance of Neural Machine Translation (NMT; Sutskever et al. (2014)) tends to degrade on low-resource languages (LRL) due to a paucity of parallel data (Koehn and Knowles, 2017; Sennrich and Zhang, 2019). One effective strategy to improve translation in LRLs is through multilingual training using parallel data from related high-resource languages (HRL) (Zoph et al., 2016; Neubig and Hu, 2018). The assumption underlying cross-lingual transfer is that by sharing parameters between multiple languages the LRL can benefit from the extra training signal from data in other languages. One of the most popular strategies for multilingual training is to train a single NMT model that translates in many directions by simply appending a flag to each source sentence to indicate which target language to translate into (Ha et al., 2016; Johnson et al., 2017).

Many works focus on using multilingual training to improve *many-to-one* NMT models that translate *from* both an HRL and an LRL to a single target language (Zoph et al., 2016; Neubig and Hu, 2018; Gu et al., 2018). In this situation, sentences from the HRL-target corpus provide an extra training signal for the decoder language model, on top of cross-lingual transfer on the source side. When training an NMT model that translates *into* an LRL, however, multilingual data tends to lead to smaller improvements (Lakew et al., 2019; Arivazhagan et al., 2019; Aharoni et al., 2019).

In this paper, we aim to improve the effectiveness of multilingual training for NMT models that translate *into* LRLs. Prior work has found vocabulary overlap to be an important indicator of whether data from other languages will be effective in improving NMT accuracy (Wang and Neubig, 2019; Lin et al., 2019). Therefore, we hypothesize that one of the main problems limiting multilingual transfer on the target side is that the LRL and the HRL may have limited vocabulary overlap, and standard methods for embedding target words via lookup tables would map corresponding vocabulary from these languages to different representations.

To overcome this problem, we design a target word embedding method for multilingual NMT that encourages similar words from the HRLs and the LRLs to have similar representations, facilitating positive transfer to the LRLs. While there are many methods to embed words from characters (Ling et al., 2015; Kim et al., 2016; Wieting et al., 2016; Ataman and Federico, 2018), we build our model upon Soft Decoupled Encoding (SDE; Wang et al. (2019)), a recently-proposed general-purpose multilingual word embedding method that has demonstrated superior performance to other alternatives. SDE represents a word by combin-

[1] Open-source code is available at https://github.com/luyug/DecSDE

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3560–3566

ing a character-based representation of its spelling and a lookup-based representation of its meaning. We propose DecSDE, an efficient adaptation of SDE to NMT decoders. DecSDE uses a low-rank transformation to assist multilingual transfer, and it precomputes the embeddings for a fixed vocabulary to speedup training and inference. We test our method on translation from English to 4 different low-resource languages, and DecSDE brings consistent gains of up to 1.8 BLEUs.

2 Translating into Low-resource Languages

Standard NMT training is performed solely on parallel corpora from a source language S to a target language T. However, in the case that T is an LRL, we can use parallel data from S and a related HRL T' to assist learning. The standard look-up embedding in NMT turns words from both the LRL and the HRL into vectors by mapping their indices in the vocabulary to the corresponding entry in the embedding matrix. This is harmful for positive transfer, because different words with similar spellings from the LRL and the HRL are mapped to independent embeddings. For example, "Ola" in Galician and "Olá" in Portuguese both mean "hello", but they would have separate representations through the look-up embedding. We give a demonstration of this embedding (mis-)alignment in § 5.2. Since the target side data is essential for training the decoder's language model, representing lexicons from the LRL and HRL into shared space is especially important to improve positive transfer for NMT models that translate into LRLs.

3 Soft Decoupled Encoding

To address the limitation of the standard word representation for target side multilingual transfer, we turn to Soft Decoupled Encoding (SDE;Wang et al. (2019)), a word embedding method designed for multilingual data. SDE decomposes a word embedding into two components: a character n-gram embedding with a language-specific transformation that represents its spelling, and a semantic embedding that represents its meaning. Given a word w from the target language L_i, SDE embeds the words in three steps.

Character aware embeddings are first used to calculate the lexical representation of w. We extract a bag of n-grams frequency vector from w, denoted as $\text{BoN}(w)$, where each row corresponds to the number of times a character n-gram in the vocabulary appears in w. The character aware embedding of the w is then computed as

$$c(w) = \tanh\left(\mathbf{W}_c \, \text{BoN}(w)\right), \tag{1}$$

where tanh is the activation function and $\mathbf{W}_c \in \mathbb{R}^{d \times n}$ is an embedding matrix of dimension d for the n character n-grams in the vocabulary.

Language-specific transformation is then applied to lexical embedding $c(w)$ to account for the divergence between the HRL and the LRL:

$$c_i(w) = \tanh\left(\mathbf{W}_{L_i} \, c(w)\right), \tag{2}$$

where the matrix $\mathbf{W}_{L_i} \in \mathbb{R}^{d \times d}$ is a linear transformation specific to the language L_i.

Latent semantic embeddings of w are calculated using an embedding matrix $W_s \in \mathbb{R}^{d \times s}$ with s entries, which is shared between the languages. We use $c_i(w)$ as the query vector to perform attention (Luong et al., 2015) over the embeddings

$$s(p) = \mathbf{W}_s \, \text{softmax}\left(\mathbf{W}_s^\top \, c_i(w)\right). \tag{3}$$

The final embedding of w is obtained by summing the lexical and semantic representations

$$e_{\text{SDE}}(w) = c_i(w) + s(w). \tag{4}$$

4 DecSDE for NMT Decoders

In this section, we build upon the previously described SDE, and design a new method for multilingual word representation on the *target* side.

There are two aspects to consider when incorporating character-based representations like SDE in decoders: 1) the embedding method should be efficient during both training and inference time, as it needs to be calculated over the entire vocabulary; 2) it should support popular decoder design decisions, such as weight tying (Press and Wolf, 2017), which allows the decoder to share the parameters of the target embedding matrix and the decoder projection before the softmax operation. With these considerations in mind, we introduce DecSDE, a multilingual target word embedding method based on SDE for NMT decoders.

Fixed Vocabulary and Weight Tying The standard SDE is designed to encode words directly without segmenting them into subwords (Wang et al., 2019). This design choice works well for encoding words on the source side, but it can cause problems for the decoder, which requires a finite vocabulary to generate words for each time step. Therefore, we choose to segment the target sentences into subwords (Kudo and Richardson, 2018), and encode each subword using DecSDE.

The use of a fixed vocabulary also allows us to perform weight tying. Specifically, we construct an embedding matrix for the decoder by precomputing the DecSDE embedding for each subword in the target vocabulary. This embedding matrix can then be used both as the encoder lookup table and as the projection matrix before the decoder softmax.

Efficient Training and Inference One drawback of the standard SDE is that it requires more computation than standard look-up table embeddings because the lexical embedding requires one to extract and embed all character n-grams for each word. This problem is especially important for the decoder, since it needs to embed all target words in the vocabulary for each time step to calculate the probability distribution over the vocabulary.

To make training more efficient, we extract the character n-grams for all words in the target vocabulary, and use an optimized embedding bag layer[2] to parallelize the calculation of lexical embeddings for all words in a batch. For inference, we precompute the DecSDE embedding for all subwords, effectively making inference as fast as the regular look-up table embedding. An analysis of training and inference speed can be found in § 5.2.

Low-rank Language-Specific Transformation The language-specific transform in the standard SDE used on the encoder side sometimes hurts the model performance (Wang et al., 2019). Our experiments confirm that this phenomenon also happens on the decoder side. We hypothesize that this is because the full-rank transformation matrix, that is $\mathbf{W}_{L_i}$ in Eq. 2 might overfit the training data and project the lexical embeddings from different languages too far from each other, which could hurt multilingual transfer. Therefore, we introduce a novel low-rank language-specific transformation for DecSDE: We upper-bound the rank of the transformation matrix so that it is less complex, which can encourage generalization. Specifically, we replace language-specific transformation matrix $\mathbf{W}_{L_i}$ in Eq. 2 with two components: an identity matrix and a low-rank factorized matrix,

$$\mathbf{W}_{L_i} = \mathbf{I} + \mathbf{U}_{L_i}\mathbf{V}_{L_i} \tag{5}$$

where $\mathbf{U}_{L_i} \in \mathbb{R}^{d\times u}$, $\mathbf{V}_{L_i} \in \mathbb{R}^{u\times d}$ are the low-rank matrices with dimension $u < d$. Thus, the identity matrix $\mathbf{I}$ passes through the lexical embedding as-is, and the low-rank matrix performs a simple transformation to account for the divergence between languages without amplifying the difference.

Extension to Multiple Target Language Note that though in this work we focus on HRL and LRL pairs, one can easily extend the framework to multiple (> 2) target languages. In particular, the only language dependent component of DecSDE is the matrices W_{L_i}, while the rest of DecSDE parameters as well as transformer encoder-decoder parameters are shared. We can add and train W_{L_j} for each of additional language L_j.

5 Experiments

5.1 Setup

Datasets To validate our method, we use the 58-language-to-English TED corpus for experiments (Qi et al., 2018). We use three LRL datasets: Azerbaijani (aze), Belarusian (bel), Galician (glg) to English, and a slightly higher-resource dataset Slovak (slk). Each LRL is paired with a related HRL: Turkish (tur), Russian (rus), Portuguese (por), and Czech (ces) respectively. We translate from English to each of the four LRLs, and train together with the corresponding HRL. For simplicity, as a research setup, we do not use back-translation with mono-lingual data which is also hard to come by for languages low in resource we experiment with.

Implementation We implement our method using the fairseq (Ott et al., 2019) toolkit. We use the Transformer (Vaswani et al., 2017) NMT model with 6 encoder and decoder layers and 4 attention heads. Other details of model architecture can be found in § A.1. For all experiments, we use SentencePiece (Kudo and Richardson, 2018) with a vocabulary size of 16K.

Compared Systems We compare with two systems: 1) LookUp-piece: we use SentencePiece separately on each language to get subword vocabularies. Both encoder and decoder use look-up based

[2]Implementation with torch.nn.functional.embedding_bag

embeddings. 2) LookUp-word: We concatenate the training data together and extract the most frequent 64K tokens as the shared vocabulary. Both encoder and decoder use look-up based embeddings. Both systems employ vanilla weight-tying.

5.2 Experiment Results

Model	aze	bel	glg	slk
LookUp-word	0.26	2.65	5.91	6.7
LookUp-piece	5.18	9.81	21.86	21.34
DecSDE	**6.66**	**11.56**	**23.68**	**22.55**
- weight tying	5.75	9.5	22.22	21.2
- transform	6.26	10.18	23.68	22.4
- low-rank transform	5.65	11.36	22.1	22.2

Table 1: Model performance and ablations. DecSDE outperforms the best baseline for all four languages.

Performance We measure model performance using SacreBLEU (Post, 2018) and summarize the results in Tab. 1. DecSDE consistently improves over the best baseline for all languages, outperforming LookUp-piece by up to 1.8 BLEU. Meanwhile, we see word-level baseline has inferior performance, likely due to little word-level overlap between HRL and LRL.

Ablation We examine the effect of DecSDE components by removing each of them, as in Tab. 1. First, we can see that removing weight tying degrades the model performance by a large margin for all four languages. Next, comparing the standard linear transformation (- low-rank transform), and the method without the entire language-specific transform component (- transform), we can see that using the regular transform without low-rank factorization actually degrades the model performance for three out of the four languages, indicating that a full linear transformation might hinder multilingual transfer. Using the low-rank transform achieves the best performance for all four languages.

Speed We measure the training time for one epoch, and the decoding time of the whole test set for aze. The results are in Tab. 2. DecSDE incurs a reasonable training overhead, and has similar inference speed as the regular lookup embedding.

Effect of Vocabulary Size We compared DecSDE and LookUp-piece with different vocabulary sizes to study the impact of subword segmentation and show results in Tab. 3. DecSDE consistently outperforms LookUp-piece, but both methods tend to demonstrate decreasing accuracy as the vocabulary size gets larger.

Model	Train	Decode
LookUp-Piece	152 sec	13.2 sec
DecSDE	341 sec	11.5 sec

Table 2: Train/inference speed. DecSDE has similar inference speed as standard look-up embeddings.

Method	# Vocab	aze	bel	glg	slk
LookUp-piece	8K	6.18	9.2	22.02	21.92
	16K	5.18	9.81	21.86	21.34
	32K	5.03	8.75	21.27	20.37
DecSDE	8K	6.43	11.57	23.81	22.92
	16K	6.26	11.36	23.68	22.4
	32K	5.36	10.65	22.85	20.16

Table 3: Performance with Different Vocab Size.

Effect of N-gram Size DecSDE builds up its character n-gram vocabulary by extracting n-grams of lengths from 1 up to n from the input vocabulary. Using a larger n makes the model more expressive, but it might adds more parameters the model which could lead to overfitting. In this section, we examine the effect of different n values on DecSDE. The results are listed in Tab. 4.

N-gram	aze	bel	glg	slk
3	5.24	11.07	21.45	21.48
4	6.16	11.36	23.35	22.4
5	6.26	10.86	23.68	21.62

Table 4: N-gram Size

We observe that using upto 4-gram give a huge performance improvement, while using 5-gram leads to small improve in aze and glg but small decrease in bel, slk. This suggests using character n-grams up to size 4 is enough to provide enough discriminative power for our model.

Latent	aze	bel	glg	slk
5K	5.73	11.1	23.87	22.65
10K	6.26	11.36	23.68	22.4
20K	5.92	11.06	22.94	22.25

Table 5: Latent Size

Effect of Latent Size We train DecSDE with different latent embedding sizes of 5K, 10K and 20K and record BLEU in Tab. 5. We observe small differences among them for each LRL. We do find a trend that increasing the size too high will hurt performance, indicating a latent size of around 10K

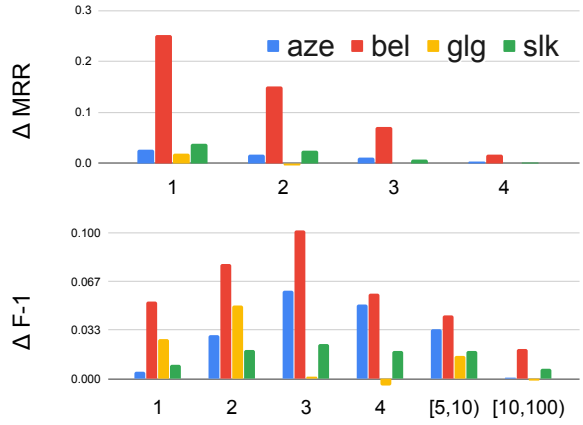

Figure 1: *Top*: Gain in embedding similarity for similarly spelled HRL, LRL word pairs. *Bottom*: Gain in word accuracy F-1 over rare words in the LRLs.

is sufficient while going larger is likely to incur over-fitting problem.

Embedding Analysis One main advantage of DecSDE is its ability to capture spelling similarity between LRL and HRL. To show this, we pick word pairs from HRL and LRL with edit distance from 1 to 4, and compare their embeddings. For each word pair word pair, we take the LRL word and use the cosine similarity between embeddings to retrieve words from the HRL. Retrieval success is measured by mean reciprocal rank (MRR, the higher the better). The gain of DecSDE over LookUp-piece with respect to edit distance is plotted in the top of Fig. 1, which shows that DecSDE embed similar spelling words closer in the embedding space.

Next, we examine performance of DecSDE for rare words in the LRLs. We calculate word F-1 of rare words for DecSDE and LookUp-piece using compare-mt (Neubig et al., 2019), and plot word frequency vs. gain in word F-1 of in the bottom of Fig. 1. DecSDE brings more significant gains for less frequent words, likely because it encodes similar words in HRL and LRL to closer space, thus assisting positive transfer.

6 Implications and Future Work

In this paper, we have demonstrated that DecSDE, a multilingual character-sensitive embedding method, improves translation accuracy into low resource languages. This implies, on a higher level, that looking into the character-level structure of the target-side vocabulary when creating word or subword embeddings is a promising way to improve cross-lingual transfer. While ablations have shown that the proposed design decisions (such as Low-rank Language-specific transformation, weight tying, etc.) are reasonable ones, this is just a first step in this direction. Future work could examine even more effective methods for target-side lexical sharing in MT or other language generation tasks.

Acknowledgments

This work was supported in part by an Apple PhD Fellowship to XW. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

References

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. In *NAACL*.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges. In *arxiv*.

Duygu Ataman and Marcello Federico. 2018. Compositional representation of morphologically-rich input for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 305–311, Melbourne, Australia. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Thanh-Le Ha, Jan Niehues, and Alexander H. Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *Arxiv*.

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.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. *AAAI*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *ACL*, abs/1808.06226.

Surafel M. Lakew, Marcello Federico, Matteo Negri, and Marco Turchi. 2019. Multilingual neural machine translation for low-resource languages. *IJCoL*.

Yu-Hsiang Lin, Chian-Yu Chen, Jean Lee, Zirui Li, Yuyan Zhang, Mengzhou Xia, Shruti Rijhwani, Junxian He, Zhisong Zhang, Xuezhe Ma, Antonios Anastasopoulos, Patrick Littell, and Graham Neubig. 2019. Choosing transfer languages for cross-lingual learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3125–3135, Florence, Italy. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based neural machine translation. *arxiv*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Graham Neubig, Zi-Yi Dou, Junjie Hu, Paul Michel, Danish Pruthi, Xinyi Wang, and John Wieting. 2019. compare-mt: A tool for holistic comparison of language generation systems. *CoRR*, abs/1903.07926.

Graham Neubig and Junjie Hu. 2018. Rapid adaptation of neural machine translation to new languages. In *EMNLP*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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. *ArXiv*, abs/1608.05859.

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana. 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.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.

Xinyi Wang and Graham Neubig. 2019. Target conditioned sampling: Optimizing data selection for multilingual neural machine translation. In *ACL*.

Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. 2019. Multilingual neural machine translation with soft decoupled encoding. *ICLR*, abs/1902.03499.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515, Austin, Texas. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *EMNLP*.

A Appendix

A.1 Training Details

For DecSDE and both baseline models, we use the transformer architecture. Both the transformer encoder and decoder have six layers, four attention heads, 512 embedding dimension and 1024 FFN dimension. All models are trained with a stochastic gradient descent with Adam optimizer, with a learning rate of 5e-4 with a inverse square root scheduler, for a maximum of 50 epochs. Dropout of 0.3, label smoothing of 0.1 are used. These are inherited from fairseq (Ott et al., 2019)'s low resource IWSLT'14 German to English (Transformer) example[3]. For DecSDE, we have $u = 16$ for aze, $u = 80$ for bel, $u = 0$ for glg and $u = 48$ for slk. We select u by manual search based on dev set perplexity. A latent size of 10K is used unless specified otherwise following the original SDE paper. Charater n-gram up to size of 5 are used for aze and glg, and up to 4 for bel and slk. We pick this among 3, 4 an 5 by dev set perplexity. With DecSDE, the models have approximately 60M parameters. In comparison, the baseline LookUp-piece have roughly 52M parameters.

A.2 Datasets

TED dataset and preprocessing tools we used are available at https://github.com/neulab/word-embeddings-for-nmt. Futher word segementations are done with SentencePiece running the uni-gram sub-word algorithm.

[3] https://github.com/pytorch/fairseq/tree/master/examples/translation