Compact Personalized Models for Neural Machine Translation

Joern Wuebker, Patrick Simianer, John DeNero
Lilt, Inc.
first_name@lilt.com

Abstract

We propose and compare methods for gradient-based domain adaptation of self-attentive neural machine translation models. We demonstrate that a large proportion of model parameters can be frozen during adaptation with minimal or no reduction in translation quality by encouraging structured sparsity in the set of offset tensors during learning via group lasso regularization. We evaluate this technique for both batch and incremental adaptation across multiple data sets and language pairs. Our system architecture—combining a state-of-the-art self-attentive model with compact domain adaptation—provides high quality personalized machine translation that is both space and time efficient.

1 Introduction

Professional translators typically translate a collection of related documents drawn from a domain for which they have a set of previously translated examples. Domain adaptation is critical to providing high quality suggestions for interactive machine translation and post-editing interfaces. When many translators use the same shared service, the system must train and apply a *personalized* adapted model for each user. We describe a system architecture and training method that achieve high space efficiency, time efficiency, and translation performance by encouraging structured sparsity in the set of offset tensors stored for each user.

Effective model personalization requires both *batch* adaptation to an in-domain training set, as well as *incremental* adaptation to the test set. Batch adaptation is applied when a user uploads relevant translated documents before starting to work. Incremental adaptation is applied when a user provides a correct translation of each segment just after receiving machine translation suggestions, and the system is able to train on that correction before generating suggestions for the next segment. This is referred to as *a posteriori* adaptation by Turchi et al. (2017). Our experiments compare both types of adaptation. There are cases for which incremental adaptation achieves better performance using fewer examples, as examples drawn directly from the test set are often highly relevant to subsequent parts of that test set. There are also cases for which the gains from both types of domain adaptation are additive.

The time required to translate and to adapt both must be minimal in a personalized translation service. Interactive translation requires suggestions to be generated at typing speed, and incremental adaptation must occur within a few hundred milliseconds to keep up with a translator's typical workflow. The service can be expected to store models for a large number of users and dynamically load and adapt models for many active users concurrently. Therefore, minimizing the number of parameters stored for each user's personalized model is important both for reducing storage requirements and latency. We achieve space and time efficiency by representing each user's model as an offset from the unadapted baseline parameters and encouraging most offset tensors to be zero during adaptation.

We show that group lasso regularization can be applied to a self-attentive Transformer model to freeze up to 75% of the parameters with minimal or no loss of adapted translation quality across experiments on four English→German data sets. We confirm these findings for six additional language pairs.

2 Related Work

There is extensive work on incremental adaptation from human post edits or simulated post edits, both for statistical machine translation (Green et al., 2013; Denkowski et al., 2014a,b; Wuebker et al., 2015) and neural machine translation (Peris et al.,

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 881–886
Brussels, Belgium, October 31 - November 4, 2018.

2017; Turchi et al., 2017; Karimova et al., 2017). Both Turchi et al. (2017) and Karimova et al. (2017) apply vanilla fine-tuning algorithms. In addition to fine-tuning towards user corrections, the former applies *a priori* adaptation to retrieved data that is similar to the incoming source sentences. Peris et al. (2017) propose a variant of fine-tuning with passive-aggressive learning algorithms. In contrast to these papers, where all model parameters are possibly altered during training, this work focuses on space efficiency of the adapted models.

Regularization methods that promote or enforce sparsity have been previously used in the context of sparse feature models for SMT: Duh et al. (2010) presented an application of multi-task learning via ℓ_1/ℓ_2 regularization for feature selection in an N-best reranking task. A similar approach, employing ℓ_1/ℓ_2 regularization for feature selection and multi-task learning, was developed by Simianer et al. (2012) and Simianer and Riezler (2013) for tuning of SMT systems. Both works report improvements from regularization.

Techniques for enforcing sparse models using ℓ_1 regularization during stochastic gradient descent optimization were previously developed for linear models (Tsuruoka et al., 2009).

An extremely space efficient method for personalized model adaptation is presented by Michel and Neubig (2018). Here, adaptation is performed solely on the output vocabulary bias vector. Another notable approach for creating compact models is student-teacher-training or knowledge distillation (Kim and Rush, 2016). To the best of our knowledge, this has not been applied in a domain adaptation setting.

3 Self-Attentive Translation Model

The neural machine translation systems used in this work are based on the Transformer model introduced by Vaswani et al. (2017), which uses self-attention rather than recurrent or convolutional layers to aggregate information across words. In addition to its superior performance, its main practical advantage over recurrent models is faster training.

The Transformer follows the encoder-decoder paradigm. Source word vectors $x_1, \ldots, x_m$ are chosen from an embedding matrix X_e. A series of stacked encoder layers generate intermediate representations $z_1, \ldots, z_m$. Each layer of the encoder consists of two sub-layers: a multi-head *self-attention* layer that uses scaled dot-product attention over all source positions, followed by a feed-forward *filter* layer. Layer normalization (Ba et al., 2016), dropout (Srivastava et al., 2014), and residual connections (He et al., 2016) are applied to each sub-layer.

A series of stacked decoder layers produces a sequence of target word vectors $y_1, \ldots, y_n$. Each decoder layer has three sub-layers: self-attention, encoder-attention, and a filter. For target position j, the *self-attention* layer can attend to any previous target position $j' \in [1, j]$, with target words offset by one so that representations at j can observe word $j-1$, but not word j. The *encoder-attention* layer can attend to the final encoder state z_i for any source position $i \in [1, m]$. Observed target word vectors are chosen from an embedding matrix Y_e, and target word j is predicted from y_j via a soft-max layer parameterized by an output projection matrix Y_o.

The encoders in this work have six layers that have a *self-attention* sub-layer size of 256 and a *filter* sub-layer size of 512. Each filter performs two linear transformations and a ReLU activation:

$$f(x) = \max(0, xW_1 + b_1)W_2 + b_2.$$

The decoders in this work have three layers, and all sub-layer sizes are 256. The decoder sub-layers are simplified versions of those described in Vaswani et al. (2017): The *filter* sub-layers perform only a single linear transformation, and layer normalization is only applied once per decoder layer after the *filter* sub-layer.

Unlike in Vaswani et al. (2017), none of X_e, Y_e, or Y_o share parameters in our TensorFlow[1] implementation. Baseline models are optimized with Adam (Kingma and Ba, 2015).

4 Compact Adaptation

4.1 Fine Tuning

Personalized machine translation requires batch adaptation to a domain-relevant bitext (such as a user provided translation memory) as well as incremental adaptation to the test set. We apply fine-tuning, which involves continuing to train model parameters with a gradient-based method on domain-relevant data, as a simple and effective method for neural translation domain adaptation (Luong and Manning, 2015). The fine-tuned model without regularization and clipping is denoted as the *Full Model*. Confirming previous work, we found that

[1] https://www.tensorflow.org/

stochastic gradient descent (SGD) is the most effective optimizer for fine tuning (Turchi et al., 2017). In our experiments, batch adaptation uses a batch size of 7000 words for 10 Epochs and a fixed learning rate of 0.1, dropout of 0.1, and label smoothing with $\epsilon_{ls} = 0.1$ (Szegedy et al., 2016).

Incremental adaptation uses a batch size of one and a learning rate of 0.01. To ensure a strong adaptation effect within a single document, we set dropout and label smoothing to zero and perform up to three SGD updates on each segment. After each update, we measure the model perplexity on the current training example and continue with another update if the perplexity is still above 1.5.

4.2 Offset Tensors

In a personalized translation service, adapted models need to be loaded quickly, so a space-efficient representation is critical for time efficiency as well. Production speed requirements using contemporary cloud hardware limit model sizes to roughly 10M parameters per user, while a high-quality baseline Transformer model typically requires 35M parameters or more. We propose to store the parameters of an adapted model as an offset from the baseline model. Each tensor is a sum $W = W_b + W_u$, where W_b is from the baseline model and is shared across all adapted models, while the offset W_u is specific to an individual user domain. Space efficiency is achieved by only storing W_u for a subset of tensors and setting the rest of the offset tensors to zero.

One approach to achieving model sparsity is to manually partition the network into a small number of regions and systematically evaluate translation performance when storing offsets for only one region. We define five distinct regions, which are evaluated in isolation: Outer layers (the first and last layers of both encoder and decoder), inner layers (all the remaining layers), the two embedding matrices X_e and Y_e, and the output projection matrix Y_o. The latter three are each stored as a single matrix and each contributes 10.3M parameters to the full model size in English→German. During adaptation, the embedding matrices are only updated for vocabulary present in the training examples, and so the offsets can be stored efficiently as a sparse collection of columns. The same principle can be applied to the output projection matrix by only updating parameters corresponding to vocabulary items that appears in the adaptation examples (denoted *Sparse Output Proj.* in Table 1).

A second approach to achieving model sparsity is to use a procedure to select the subset of offset tensors that are stored. For example, we evaluate a simple policy that stores an offset for all tensors whose average change in parameter values is higher than a threshold. This set is selected on a development domain and held fixed for all other domains. We refer to this method as *fixed* adaptation.

4.3 Tensor Selection via Group Lasso

A group sparse regularization penalty such as group lasso can be applied to the offset tensors for simultaneous regularization and tensor selection. This penalty drives entire offset tensors to zero, so that they do not need to be stored or loaded. We add the following regularization term to the loss function (Scardapane et al., 2017):

$$R_{\ell_{1,2}}(\mathcal{T}) = \sum_{T \in \mathcal{T}} \sqrt{|T|} \|\Delta T\|_2 \quad (1)$$

$$\|\Delta T\|_2 = \sum_{\tau \in T} \Delta\tau^2 \quad (2)$$

Here, each tensor corresponds to one group. $\mathcal{T}$ denotes the set of all tensors in the model, $\tau \in T$ the set of all weights within a single tensor and $\Delta\tau$ the size of the offset for τ. Note that we are regularizing the *difference* between the parameters of the adapted model and the baseline model, rather than regularizing the full network parameters directly. In this way, we maintain the expressive power of the full network while minimizing the size of the adapted models. Group lasso regularization is equivalent to ℓ_1 regularization when the group size is 1. Sparsity among groups is encouraged because the ℓ_1 norm serves as a convex proxy for the ℓ_0 norm, which would explicitly penalize the number of non-zero elements (Yuan and Lin, 2006). To facilitate tensor selection, we define a threshold ϑ to clip offset tensors ΔT with average weight $\frac{1}{|T|}\sum_{\tau \in T} \Delta\tau < \vartheta$ to zero. Both the threshold ϑ and the regularization weight λ were manually tuned on a development domain and set to $\vartheta = 10^{-4}$ and $\lambda = 10^{-6}$. We apply clipping to all tensors except the embedding and output projection matrices X_e, Y_e and Y_o. As our production constraints allow us to retain only one of the three, we pre-select the sparse output projection as part of the model and exclude the embedding matrices from adaptation. This method will be denoted as *Lasso*.

	# Param.	User1 Batch	User1 Incr.	User2 Batch	User2 Incr.	Autodesk Batch	Autodesk Incr.	IWSLT Batch	IWSLT Incr.
Baseline	36.2M	35.7		32.7		40.3		25.9	
Full Model	25.8M	47.5	48.2	44.2	34.8	47.7	46.6	27.5	26.3
Outer Layers	2.2M	45.0	47.9	36.4	33.1	45.5	44.7	27.3	26.1
Inner Layers	2.7M	45.4	47.2	36.7	33.6	45.5	43.9	27.8	26.5
Encoder Embed.	5.0M	41.7	41.8	33.6	32.9	42.5	41.6	27.4	26.4
Decoder Embed.	5.5M	36.6	37.6	33.0	33.0	40.8	40.5	26.2	25.9
Output Proj.	10.3M	44.2	46.0	38.1	34.5	45.2	42.9	27.1	26.5
Sparse Output Proj. (*)	5.5M	43.5	46.7	39.7	34.7	45.5	43.3	27.1	26.7
(*) + Fixed	6.9M	46.4	47.8	42.3	30.9	47.6	43.7	27.3	26.0
(*) + Lasso	6.7M	47.6	46.6	43.1	33.2	47.9	41.5	27.5	27.0
Full Model Batch+Incr.	25.9M	50.6		41.8		52.6		27.0	
(*) + Lasso Batch+Incr.	9.2M	51.3		39.1		51.1		27.6	
Repetition Rate Source		11.0		8.8		18.3		9.2	
Repetition Rate Target		9.4		8.7		17.5		7.5	

Table 1: Experimental results in BLEU (%) on the English→German data. We evaluate changes to each region of the network separately. In combination with sparse output projection, we also evaluate a *fixed* selection of parameters chosen by thresholding and a set selected dynamically for each data set using group *lasso*. The two bottom rows show repetition rates in % for the source and target sides of the test data.

5 Experiments

5.1 Data

We first evaluate all techniques on an English→German Transformer network trained on 98M parallel sentence pairs. We apply byte pair encoding (Sennrich et al., 2016) separately to each language and obtain vocabularies with 40K unique tokens each. We refer to the unadapted model as *Baseline*. We evaluate on four domains. For development, we use a data set labeled *User1* that was gathered from a user of the browser-based CAT (computer-aided translation) tool Lilt[2] and contains documents from the financial domain with 48K segments for batch adaptation and 1790 segments for testing and incremental adaptation. We further evaluate on a second user test set *User2* (technical support, 31k batch adaptation, 1000 test segments); the public Autodesk corpus[3], where we select the first 20k segments for batch adaptation and the next 1000 segments for testing; and the IWSLT corpus[4] (semi-technical talks), where we use all provided 206K sentences for batch adaptation and the *dev2010* set (888 sentences) for testing. The overall best performing compact adaptation technique, group lasso regularization, is further evaluated on six other language pairs trained using production data sets collected from Lilt's user base: English↔French, English↔Russian and English↔Chinese. Adaptation is performed on user data from various domains (technical manuals, finance, legal), each with 8k-10k segments for batch adaptation and 2000 segments for testing and incremental adaptation. Translation quality is evaluated using the cased BLEU (Papineni et al., 2002) measure.

5.2 Results

Table 1 shows English→German results. Full model adaptation, where all offsets are stored, improves over the baseline in all cases to various degrees. This full model contains only 25.8M parameters, as offsets for both embedding matrices are stored as sparse collections of columns for the vocabulary present in the adaptation data. Next, we evaluate the impact of storing offsets only for one region at a time. We observe that among the three vocabulary matrices, the output projection Y_o has the strongest impact on quality, which is not dimin-

[2] https://lilt.com

[3] https://autodesk.app.box.com/Autodesk-PostEditing

[4] http://workshop2017.iwslt.org/

	en→fr	fr→en	en→ru	ru→en	en→zh	zh→en	Avg.
Baseline	28.8	35.8	10.7	29.2	19.9	18.9	23.9
Full Model Batch+Incr.	36.6	49.6	21.0	42.1	40.6	46.6	39.4
(*) + Lasso Batch+Incr.	36.0	46.3	19.8	42.7	39.3	45.0	38.2

Table 2: Experimental results in BLEU (%) on six production language pairs. We compare the unadapted baseline model with a full model and the model with sparse output projection and group lasso, both with application of batch and incremental adaptation.

ished by storing a sparse variant that is restricted only to observed vocabulary.

In addition, we evaluate two methods of choosing a subset of tensors procedurally. We first experiment with a fixed subset of tensor offsets that was chosen by selecting all tensors for which parameters were offset by more than 0.002 on average after batch adaptation on the *User1* data set. This simple procedure approaches the performance of full model adaptation, but stores only 27% of its parameters. Dynamically selecting tensor offsets for each data set using group lasso regularization improves performance on 6 out of 8 data conditions.

The combination of batch and incremental adaptation yields further improvements, with the exception of the *User2* and *IWSLT* tasks, where incremental adaptation overall performs not as well as batch adaptation. For these tasks, both tests sets exhibit lower repetition rates[5] (Cettolo et al., 2014) than the test sets for the two other tasks (see the two bottom lines in Table 1). The *User2* test set is furthermore a random sample of non-consecutive text from a translation memory, which is suboptimal for incremental learning.

Altogether, we are able to achieve translation performance similar to full model adaptation with 25% of the total network parameters. Note that due to the selection of entire tensors with groupwise regularization, there is nearly zero space overhead incurred by storing a sparse set of offset tensors.

Table 2 confirms our main findings on six other language pairs. We observe average improvements of 14.3 BLEU with our final compact model, which compares to 15.5 BLEU for full model adaptation.

6 Conclusion

We describe an efficient approach to personalized machine translation that stores a sparse set of tensor offsets for each user domain. Group lasso regularization applied to the offsets during adaptation achieves high space and time efficiency while yielding translation performance close to a full adapted model, for both batch and incremental adaptation and their combination.

[5]Repetition rates have been confirmed to be a suitable indicator for gains through incremental adaptation in numerous works (Wuebker et al., 2015; Bertoldi et al., 2014).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Nicola Bertoldi, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. 2014. Online adaptation to post-edits for phrase-based statistical machine translation. *Machine Translation*, 28(3-4):309–339.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2014. The repetition rate of text as a predictor of the effectiveness of machine translation adaptation. In *Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 166–179, Vancouver, Canada.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014a. Learning from post-editing: Online model adaptation for statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 395–404, Gothenburg, Sweden. Association for Computational Linguistics.

Michael Denkowski, Alon Lavie, Isabel Lacruz, and Chris Dyer. 2014b. Real time adaptive machine translation for post-editing with cdec and transcenter. In *Proceedings of the EACL 2014 Workshop on Humans and Computer-assisted Translation*, pages 72–77, Gothenburg, Sweden. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 375–383, Uppsala, Sweden. Association for Computational Linguistics.

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013. The efficacy of human post-editing for language translation. In *ACM CHI Conference on Human Factors in Computing Systems*, Paris, France.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Sariya Karimova, Patrick Simianer, and Stefan Riezler. 2017. A user-study on online adaptation of neural machine translation to human post-edits. *CoRR*, abs/1712.04853.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Diederik Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*, Da Nang, Vietnam.

Paul Michel and Graham Neubig. 2018. Extreme adaptation for personalized neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 312–318. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Álvaro Peris, Luis Cebrián, and Francisco Casacuberta. 2017. Online learning for neural machine translation post-editing. *CoRR*, abs/1706.03196.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. 2017. Group sparse regularization for deep neural networks. *Neurocomput.*, 241(C):81–89.

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.

Patrick Simianer and Stefan Riezler. 2013. Multi-task learning for improved discriminative training in SMT. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 292–300, Sofia, Bulgaria.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, Volume 1: Long Papers*, pages 11–21, Jeju Island, South Korea. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics.

Marco Turchi, Matteo Negri, M Amin Farajian, and Marcello Federico. 2017. Continuous learning from human post-edits for neural machine translation. *The Prague Bulletin of Mathematical Linguistics*, 108(1):233–244.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical incremental adaptation for statistical machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1065.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.