Unifying Input and Output Smoothing in Neural Machine Translation

Yingbo Gao* **Baohao Liao*** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
{gao|bliao|ney}@i6.informatik.rwth-aachen.de

Abstract

Soft contextualized data augmentation is a recent method that replaces one-hot representation of words with soft posterior distributions of an external language model, smoothing the input of neural machine translation systems. Label smoothing is another effective method that penalizes over-confident model outputs by discounting some probability mass from the true target word, smoothing the output of neural machine translation systems. Having the benefit of updating all word vectors in each optimization step and better regularizing the models, the two smoothing methods are shown to bring significant improvements in translation performance. In this work, we study how to best combine the methods and stack the improvements. Specifically, we vary the prior distributions to smooth with, the hyperparameters that control the smoothing strength, and the token selection procedures. We conduct extensive experiments on small datasets, evaluate the recipes on larger datasets, and examine the implications when back-translation is further used. Our results confirm cumulative improvements when input and output smoothing are used in combination, giving up to +1.9 BLEU scores on standard machine translation tasks and reveal reasons why these smoothing methods should be preferred.

1 Introduction

Nowadays, neural network models are commonly used for the task of machine translation. The Transformer (Vaswani et al., 2017) architecture is the default choice for many competitive systems (Bojar et al., 2018; Barrault et al., 2019; Ott et al., 2018). In order to make use of large amount of available data in the target language, among others, back-translation is a frequently used method (Sennrich et al., 2016a; Edunov et al., 2018; Graça et al., 2019).

Recently, a method under the name "soft contextualized data augmentation" (Gao et al., 2019) is introduced and focus on the input side of neural machine translation models. Intuitively, the method smoothes both the source input and the target input to the model. While one-hot vectors are traditionally used to feed the word / token information to the network, this method instead uses an external language model (LM) trained on the parallel data without additional monolingual data, and achieve a "smoother" input where a "linear interpolation" operation is done on the word embedding matrices, instead of the usual "table lookup".

Conceptually, the "input smoothing" method above largely resembles the "label smoothing" method used in the output side. Initially introduced in the field of image recognition (Szegedy et al., 2016), label smoothing is included in the original Transformer setup (Vaswani et al., 2017) by default. Deducting a certain probability mass from the true target and redistributing uniformly across the vocabulary, label smoothing can also be thought of as penalizing over-confident system outputs.

From here on, to highlight our motivation, we will use "input smoothing" and "output smoothing" to refer to "soft contextualized data augmentation" and "label smoothing" respectively. In this paper, our motivation is straight-forward, that we want to carefully look at both input smoothing and output

* Equal contribution.

Proceedings of the 28th International Conference on Computational Linguistics, pages 4361–4372
Barcelona, Spain (Online), December 8-13, 2020

smoothing in combination. Specifically, below is a list of important research questions that we wish to answer:

- Do improvements from input smoothing and output smoothing stack?
- While there are several hyperparameters one can tune, such as the choice for the prior distribution to smooth with and the discounted probability mass, what is a rather good out-of-the-box recipe?
- Since back-translation is often included in the setup of a competitive neural machine translation system, what implications are there in terms of using input smoothing and output smoothing in combination with back-translation?

Consequently, we try to organize the paper in a reader-friendly structure:

1. For methodoloy in Section 3, we formally define the translation models and training criterions with various smoothing methods.
2. For experimental results in Section 4, we describe our experiment progress following a clear logic: first tune on a small dataset for fast iteration of experiments, then extend to other and larger datasets for more robust conclusions. With unified input and output smoothing, we are able to obtain improvements up to +1.9 BLEU scores.
3. For anaylses in Section 5, after confirming cumulative improvements with input and output smoothing, also in the case of back-translation, we further give our interpretations why the smoothing methods work in practice.

2 Related Work

In the recent work by Gao et al. (2019), input smoothing is motivated and introduced. Conceptually, the parallel data used to train neural machine translation models is only a sampled subset of the total "true" translation data. Under the name of "soft contextualized data augmentation", the authors use an additional LM to give soft distributions across the vocabulary for randomly selected positions. Compared to previous data augmentaion methods (Iyyer et al., 2015; Xie et al., 2017; Fadaee et al., 2017; Artetxe et al., 2018; Lample et al., 2018), the soft smoothing method achieves more significant improvements. The method is quickly adopted and motivates many interesting work, e.g. selecting words to "smooth" with dependency parsing (Duan et al., 2020), automatic repairing noisy synthetic parallel data (Cheng et al., 2020), and distilling knowledge from BERT (Devlin et al., 2019) via "text smoothing" (Wu et al., 2020).

On the other hand, output smoothing enjoys a slightly longer history. In this era where neural network are extensively used for modeling, Szegedy et al. (2016) initially introduce the method of "label smoothing" and Vaswani et al. (2017) include it by default into the Transformer setup. Discounting a certain probability mass from to one-hot true target distribution, and redistributing uniformly across the vocabulary, label smoothing wishes to solve the problem of over-fitting and boost adaptability. Relatedly, Pereyra et al. (2017) introduces a confidence penalty term, which is also mentioned in Szegedy et al. (2016), to encourage more even model outputs. Investigating the implications of label smoothing in terms of properties of learned features and in the case of knowledge distillation, Müller et al. (2019) further extends the knowledge of the smoothing method when used at the output side of the network.

To build a competitive machine translation system, the large amount of target-side monolingual data can not be neglected. Traditionally for statistical machine translation, back-translating target-side monolingual data already sees some use (Huck, 2018). For neural machine translation, Sennrich et al. (2016a) formally introduces the method of back-translation, which marks an important improvement in the training pipelines. Edunov et al. (2018) extends the method to the data scenario where huge amounts of monolingual data is available. While a separate target-to-source back-translation model is commonly used, how exactly the artificial source-side texts should be generated largely remains a yet-to-resolve research question. The sampling vs. search comparisons are already made in several works (Edunov et al., 2018; Imamura et al., 2018; Graça et al., 2019), while Wang et al. (2019) being a most recent work introducing uncertainty-based confidence estimation as an alternative.

The term "smoothing" is also traditionally used in the context of count-based language modeling (Jelinek and Mercer, 1980; Katz, 1987; Church and Gale, 1991; Kneser and Ney, 1995; Chen and Goodman, 1996). In a broader sense, one-hot vectors commonly used at the input-side and output-side of neural networks serve as empirical counts to the model. In this interpretation, both input smoothing and output smoothing are similar in that they both discount certain probability masses from the what is observed in data, to those rare events (rare combinations of input and output words / tokens). In other words, the smoothing methods discussed in this paper should have an overall effect of combating over-fitting and boosting generalization. Closely related, the standing investigations into knowledge distillation (Hinton et al., 2015; Kim and Rush, 2016; Freitag et al., 2017; Tan et al., 2019) are also connected works and may be further referred to.

3 Methodology

In this section, we define the translation models and training criterions using different combinations of smoothing at the input side and output side. The distinction among these models is necessarily made in order to draw a fair conclusion if the improvements from input and output smoothing can stack.

Consider parallel training data $f_1^J = f_1...f_j...f_J, f \in F$ and $e_1^I = e_1...e_i...e_I, e \in E$ used to train machine translation models, where f and e denote tokens in the source and target vocabularies F and E, j, J, i and I mark token positions in sentences. A typical neural model parametrized by θ can be written as $p_\theta(e_i|f_1^J, e_1^{i-1})$. Note that for all equations in this section, we drop the dependencies on the running index n in the total number of training sentence pairs N. Of course the criterions need to be normalized over the sentences accordingly.

3.1 No Smoothing

In the case of no smoothing, the model dependencies on the source input tokens f_1^J and the target input tokens e_1^{i-1} do not need to be modified. At the output side, the one-hot representations of the target token e_i can be denoted with the Kronecker-Delta δ. Therefore, the model and the criterion can be defined as in Equation (1).

$$
\begin{aligned}
p &= p_\theta\left(e_i|f_1^J, e_1^{i-1}\right) \\
L &= -\frac{1}{I}\sum_{i=1}^{I}\sum_{e\in E}\delta(e_i, e)\log p
\end{aligned}
\tag{1}
$$

Here, L is the training loss to be minimized, I is the target sentencee length in question and $\sum_{e\in E}$ is a summation over the vocabulary.

3.2 Input Smoothing

In the case of input smoothing only, the model needs an additional helper model q_{src} for source input and an additional helper model q_{tgt} for target input, and additional hyperparameters Φ that control the smoothing process. The cross-entropy training criterion stays the same here. Therefore, the model and the criterion can be defined as in Equation (2).

$$
\begin{aligned}
p &= p_\theta\left(e_i|f_1^J, e_1^{i-1}; q_{\text{src}}, q_{\text{tgt}}, \Phi\right) \\
L &= -\frac{1}{I}\sum_{i=1}^{I}\sum_{e\in E}\delta(e_i, e)\log p
\end{aligned}
\tag{2}
$$

In Gao et al. (2019), q_{src} and q_{tgt} are autoregressive LMs trained using only the parallel data. They are linearly combined with the source and target word vectors to provide smoothed inputs to the network for those selected tokens specified by Φ, replacing the usual embedding matrix lookup.

$$
\tilde{f}_{j,\text{smoothed}} = \sum_{f\in F} q_{\text{src}}(f|f_1^{j-1})\tilde{f} \tag{3}
$$

As an example, Equation (3) shows that for a selected position j, all of the word vectors $\tilde{f}$ in the source vocabulary are combined via the posterior weights from $q_{\text{src}}(f|f_1^{j-1})$ to obtain the smoothed source input vector $\tilde{f}_{j,\text{smoothed}}$. For input tokens at the target side e_1^{i-1}, similar linear combinations can be done as well.

The selection procedure for tokens to smooth is heuristical in Gao et al. (2019). In their case, $\Phi = \{\gamma\}$, where γ is the probability to randomly choose a position in J or I. Recently, Duan et al. (2020) propose to smartly select words to "smooth" with dependency parsing. Alternatively, one can also select those positions where the helper models q_{src} and q_{tgt} are most uncertain, e.g. determined by the information entropy (Shannon, 1948). Furthermore, we introduce a smoothing strength hyperparameter λ, shown in Equation (4).

$$\tilde{f}_{j,\text{smoothed}} = \lambda \sum_{f \in F} q_{\text{src}}(f|f_1^{j-1})\tilde{f} + (1-\lambda)\tilde{f}_j \tag{4}$$

3.3 Output Smoothing

In the case of output smoothing only, the model dependencies do not change from Section 3.1. However, the one-hot output targets need to be modified. Specifically, a certain probability mass m is deducted from the one-hot target distribution, and redistributed across the output vocabulary with some helper model q_{out}. Therefore, the model and the criterion can be defined as in Equation (5).

$$\begin{aligned} p &= p_\theta\left(e_i|f_1^J, e_1^{i-1}\right) \\ L &= -\frac{1}{I}\sum_{i=1}^{I}\sum_{e \in E}\left((1-m)\delta(e_i,e) + m q_{\text{out}}\right)\log p \end{aligned} \tag{5}$$

In Vaswani et al. (2017), q_{out} is a simple uniform distribution (zero-gram LM). In our case, following our previous experience, we examine both zero-gram LM $q_{\text{out}} = \frac{1}{E}$ and uni-gram LM $q_{\text{out}}(e)$.

3.4 Input Smoothing + Output Smoothing

As illustrated in Figure 1, where $\tilde{f}$, $\tilde{e}$ and $\tilde{e}'$ are word vector parameters, the smoothing methods treats the underlying architecture of the encoder, decoder and the encoder-decoder attention as black boxes. Although we only experiment with the Transformer architecture (Vaswani et al., 2017), the smoothing methods can also be applied to LSTM attention models (Bahdanau et al., 2015).

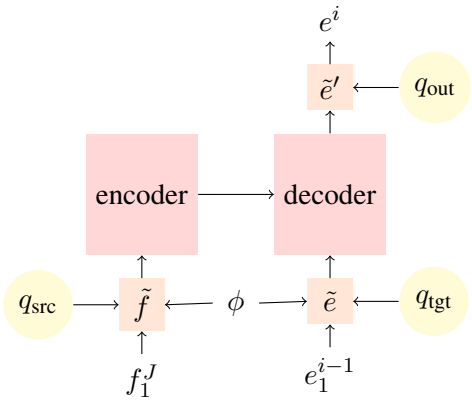


Figure 1: An illustration of input smoothing and output smoothing, where encoder and decoder architectures can be thought of as black boxes.

Summarizing discussions in Section 3.1, 3.2 and 3.3, to apply both input smoothing and output smoothing, the model and criterion can be formulated as in Equation (6).

$$\begin{aligned} p &= p_\theta\left(e_i|f_1^J, e_1^{i-1}; q_{\text{src}}, q_{\text{tgt}}, \Phi\right) \\ L &= -\frac{1}{I}\sum_{i=1}^{I}\sum_{e \in E}\left((1-m)\delta(e_i,e) + m q_{\text{out}}\right)\log p \end{aligned} \tag{6}$$

Compared to using one-hot vectors everywhere, using smoothed representations have the benefit of combining potentially all word vectors in each update step. It is important to mention that, all smoothing methods mentioned in this paper are only used during training, while during testing, the one-hot vectors are again used, following the idea that a model trained with smoothed representation of words should better generalize during testing. This can be further related to the topic of exposure bias (Schmidt, 2019), where the training-testing mismatch may lead to unseen or corrupted context during search. Although we do not conduct experiments to further validate the point, qualitatively we think input smoothing effectively lets the model see more unique contexts during training, mitigating the exposure bias problem.

A general note about the input smoothing helpers q_{src} and q_{tgt} is that, during training, the context for a certain position is not limited only to the left. In other words, one does not have to use a left-to-right LM to obtain the posterior distribution as in Gao et al. (2019). For example, a bi-directional model which looks at the context at both sides like BERT (Devlin et al., 2019) can also be used, e.g. $q_{\text{src}} = p_{\theta,\text{BERT}}(f_j|f_1^{j-1}, f_{j+1}^J)$. Another example is a system combination of a left-to-right and a right-to-left autoregressive LM, e.g. let $q_{\text{src}} = \exp\Big(\alpha_1 \log p_{\theta_1}(f_j|f_1^{j-1}) + \alpha_2 \log p_{\theta_2}(f_j|f_{j+1}^J)\Big)$. One can even go to the extreme and apply position-independent models to smooth the inputs, e.g. a uni-gram LM or a zero-gram LM, the latter of which is already used for output smoothing in Vaswani et al. (2017).

3.5 Smoothing in Back-translation

To build a competitive neural translation model, back-translation is a commonly used method (Bojar et al., 2018). Following recent studies, sampling from the back-translation model instead of using beam-search is a popular alternative to obtain the synthetic source side (Edunov et al., 2018; Imamura et al., 2018; Graça et al., 2019).

Since the back-translation model is already available, one straightforward extension is to use the the posterior distributions from the back-translation model as smoothing helpers to smooth the artificial source side. This also has the benefit of "letting the model see more data" without having to sample more synthetic source sentences and train for more steps for each epoch. To put it into mathematical notations, for a synthetic source sentence $f_{1,\text{BT}}^J$ generated by the back-translation model $p_{\theta,\text{BT}}$ using the target side monolingual sentence $e_{1,\text{BT}}^I$, the weights for the linear combination of source embeddings are directly from the back-translation model, i.e. $q_{\text{src}} = p_{\theta,\text{BT}}(f_{j,\text{BT}}|e_{1,\text{BT}}^I, f_{1,\text{BT}}^{j-1})$.

4 Experiments

For fast iterations of experiments, we first conduct experiments on a smaller dataset, namely IWSLT2014 German→English (`de-en`). After obtaining meaningful and conclusive results, we repeat the core experiments and extend to two more small datasets, IWSLT2014 Dutch→English (`nl-en`) and IWSLT2014 Spanish→English (`es-en`), plus a larger dataset, WMT2014 English→German (`en-de`). For the latter, we look at both the base and big model from the original Transformer paper and evaluate on `newstest2014`, `newstest2015` and `newstest2016`.

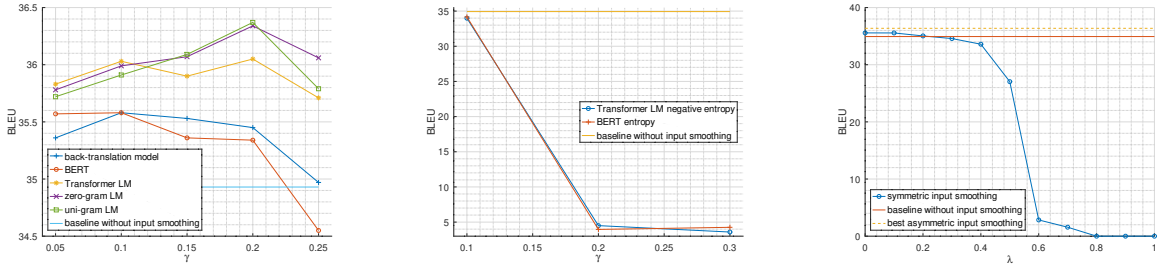
We implement the various smoothing methods in the `fairseq` toolkit (Ott et al., 2019) and conduct experiments with it. For the vocabulary, we use 10,000 and 32,000 merge operations with joint byte pair encoding from Sennrich et al. (2016b) for IWSLT and WMT respectively. For IWSLT, we largely follow Gehring et al. (2017), lowercase all sentences, remove extremely long sentence pairs, subsample sentence pairs from training data as our development set, and concatenate all available development and test sets. For WMT, we adopt the preprocessing steps in Ott et al. (2018). For all of our experiments, the three embedding matrices are tied, similar to that in Vaswani et al. (2017). For more details about preprocessing, we refer the readers to the `fairseq` source, where the preprocessing scripts are available and easy to adapt.

During training, we monitor the development set perplexity. After convergence, we further average the most recent checkpoints for a small performance boost. Note that this checkpoint averaging step is done for all of the models mentioned in this paper, so it is fair to compare the results to draw conclusions about smoothing. During beam search, we consistently use a beam size of five for IWSLT and four for

WMT for fair comparisons to literature. Finally, we report case-insensitive BLEU scores on IWSLT and case-sensitive BLEU scores on WMT (Papineni et al., 2002) on the test sets.

4.1 Tuning on a Small Dataset

Output smoothing with a zero-gram LM (Vaswani et al., 2017) is known to consistently give performance improvements. Therefore, our first step is to replicate the results from Gao et al. (2019) about input smoothing. The original authors tune the γ parameter, which controls the probability of each token to be selected for input smoothing. In our case, we follow their setup and additionally vary λ in order to determine the optimal strength of smoothing. Furthermore, we look at alternative smoothing helper models to be used as q_{src} and q_{tgt}, and examine the token selection procedure with information entropy, as mentioned in Section 3.2.



(a) Effect of different input smoothing helper model and γ.

(b) Effect of preferring the most (un)certain input tokens to smooth.

(c) Effect of smoothing all input tokens with different smoothing strength.

Figure 2: Experimental results on `de-en`.

Figure 2a shows our initial results on `de-en`. Compared to Figure 2 in Gao et al. (2019), the overall trend of BLEU scores with respect to γ is similar. The large improvements by using input smoothing can be confirmed. Interestingly, the two position-independent models, zero-gram LM and uni-gram LM, are superior to the more complex models, giving up to +1.4 BLEU improvements. Note that all the results in this Figure are obtained with zero-gram LM output smoothing, closely following the setup in Gao et al. (2019). In preparation for later experiments with back-translation, we also test if only the source input side is smoothed with a back-translation model, i.e. $q_{\text{src}} = p_{\theta,\text{BT}}(f_{j,\text{BT}}|e^I_{1,\text{BT}}, f^{j-1}_{1,\text{BT}})$. In this case, the results are not satisfactory, indicating that even for the synthetic source side, a simpler distribution is probably more helpful.

We move on to study if the simple heuristic of using information entropy (given by the helper model) to select input tokens to smooth can also give improvements. For this, we try to be creative and experiment with ranking the input tokens both by information entropy and negative information entropy. An intuitive explanation for this distinction can be made by asking the question, should one prioritize on words like "dog" and "cat", or words like "the"? In other words, does it matter if the words that the helper model is most (un)certain about are preferred? Figure 2b reveals the answer. As seen, when only a small fraction of the input tokens are selected, the performances of both heuristics are close to the baseline. But as soon as more input tokens are selected, the BLEU scores immediately drop significantly. This indicates that the default strategy to randomly select tokens to smooth should be preferred.

Since our motivation is have a unified view of input smoothing and output smoothing, aesthetically there is a major difference between the current input smoothing Gao et al. (2019) and output smoothing (Vaswani et al., 2017) methods. That is, at the output side, all target tokens are selected for smoothing with a certain m, but at the input side, only a fraction of source and target input tokens are selected for smoothing, and also smoothed strongly ($\lambda = 1$). The absolute "symmetric" case at the input side should actually be $\gamma = 1$ and $\lambda < 1$, meaning we also smooth all of the input tokens with a certain strength. Taking the best "asymmetric" results so far into account, the behavior of input smoothing with different λ is not ideal. As shown in Figure 2c, the performance with symmetric input smoothing quickly drops as λ increases, lagging behind asymmetric input smoothing consistently.

As a short summary, tuning on the small dataset de-en gives us a solid idea that it makes sense to use a simple enough helper distribution, e.g. zero-gram LM or uni-gram LM, for input smoothing. Our previous work also tells the same story at the output side. Therefore, from here on, we focus more on the use of these simple distributions for smoothing more systemically.

4.2 Evaluating on Other Datasets

First, we extend our previous results on the other two IWSLT datasets. In Table 1, the effect of applying different combinations of q_{out}, q_{src} and q_{tgt} is shown. From the results, it is clear that compared to the no smoothing baseline, input smoothing only gives consistent improvements up to around +0.6 BLEU scores. On the other hand, output smoothing only gives consistent improvements up to +1.0 BLEU scores, slightly larger than applying input smoothing only. The numbers also confirm that simple helper distributions are good enough, without the need to train additional complex models. When best setups of input smoothing and output smoothing are used in combination, cumulative improvements can be observed, giving up to around +2.0 BLEU scores. To answer the first question brought up in Section 1, we confirm that the improvements from input smoothing and output smoothing do stack.

Compared to the literature, our baseline BLEU scores have some fluctuations but we think they are acceptable. For example, in Gao et al. (2019), the baseline and LM-smoothed BLEU scores on es-en are 41.6 and 42.6, respectively. Our Transformer baseline and simple all-uni-gram-LM smoothed model are slightly worse, giving 41.3 and 42.4 respectively. However, when comparing de-en, our results are slightly better than in Gao et al. (2019): 34.9 vs. 34.8 and 36.3 vs. 35.8.

q_{out}	q_{src}	q_{tgt}	de-en	nl-en	es-en
-	-	-	34.5	37.3	40.5
	Transformer LM	Transformer LM	34.6	37.3	41.0
	BERT	BERT	34.5	37.4	40.5
	back-translation model	-	34.5	37.5	40.8
	zero-gram LM	zero-gram LM	34.8	37.9	41.0
	uni-gram LM	uni-gram LM	35.1	37.6	41.1
zero-gram LM	-	-	34.9	37.7	41.3
	Transformer LM	Transformer LM	36.1	38.2	41.6
	BERT	BERT	35.6	38.3	41.4
	back-translation model	-	35.6	38.1	41.5
	zero-gram LM	zero-gram LM	**36.3**	**38.9**	**42.4**
	uni-gram LM	uni-gram LM	36.4	39.1	42.2
uni-gram LM	-	-	35.6	38.1	41.5
	uni-gram LM	uni-gram LM	**36.3**	**39.1**	**42.4**

Table 1: Smoothing with different q_{out}, q_{src} and q_{tgt} combinations on IWSLT.

architecture	q_{out}	q_{src} & q_{tgt}	newstest		
			2014	2015	2016
Transformer base	zero-gram LM	-	27.4	29.4	33.8
		zero-gram LM	28.5	30.2	34.7
	uni-gram LM	uni-gram LM	28.5	30.6	34.8
Transformer big	zero-gram LM	-	28.0	30.3	33.6
		zero-gram LM	29.2	31.3	34.5

Table 2: Smoothing with different q_{out}, q_{src} and q_{tgt} combinations on WMT.

In Table 2, the results on WMT are shown. Compared to using output smoothing only, roughly +1.0 BLEU scores can also be obtained when combining input smoothing and output smoothing. Generally

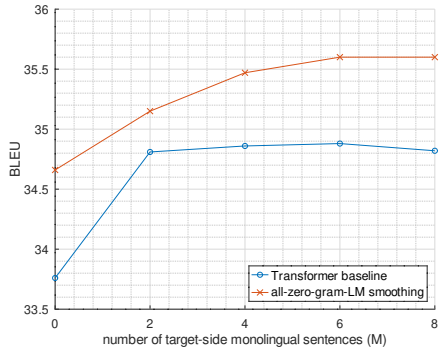
speaking, a regularization method during training should help less on larger datasets. In this case, compared to those corresponding results on IWSLT, where approximately +1.2 BLEU scores are obtained, the degradation in improvement exists, but is not drastic.

Considering Table 1 and Table 2 together, empirical evidence shows that smoothing both the input and output tokens to a Transformer model helps to improve translation performance significantly. To answer the second question brought up in Section 1, a simple and effective recipe is to use zero-gram LM or uni-gram LM helper models for q_{out}, q_{src} and q_{tgt}, without the need to train external complex models, saving both energy and time of the machine translation practitioner.

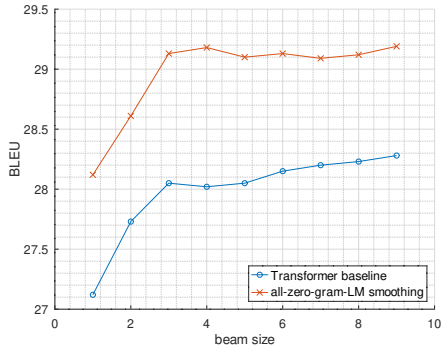
4.3 Utilizing Target-Side Monolingual Data

Here, we further look at the implications when back-translation is used. Figure 3a shows the BLEU scores of Transformer baseline system and a system where zero-gram LMs are used for q_{out}, q_{src} and q_{tgt}, also on the synthetic source sentences generated by the back-translation process. It is worth mentioning that for the Transformer baseline, the synthetic source sentences are sampled and not the hypotheses resulting from beam search. As the figure shows, the improvements seen before sustain when more target-side monolingual data is included in training. To answer the third question brought up in Section 1, the use of both input smoothing and output smoothing under the scope of back-translation should be preferred as well.

As a side experiment, we also test the robustness of the improvements against the beam size hyperparameter. As can be seen in Figure 3b, both the baseline and our model improve when a larger beam size is used in search. The improvements largely sustains and is rather robust with different hyperparameter choices in this regard.



(a) Effect of all-zero-gram-LM smoothing at both input and output sides, when back-translation is used, on `newstest16`.



(b) Effect of all-zero-gram-LM smoothing at both input and output sides, when using different beam sizes, on `newstests2014`.

Figure 3: Examining beam search and back-translation.

5 Analyses

So far, it is clear that significant empirical improvements can be obtained with both input smoothing and output smoothing. A natural question is then, why? Our interpretation of the improvements lies in how frequent the word vectors are updated during the training process.

In Chen et al. (2018), the authors stress that word vector parameters make up the lion's share in long short-term memory (Hochreiter and Schmidhuber, 1997) neural network models, where the percentage even go up to 90% for a model with a large vocabulary. In our case, the percentage is lower for Transformer models because byte pair encoding reduces vocabulary size drastically, but the word vector parameters still make up a significant portion of total parameter count. For example, for Transformer base with three-way shared vocabulary and embeddings on `en-de`, the percentage is still 27.5%.

When no smoothing is applied, each word vector in the input embedding matrices gets updated precisely the empirical count of the word times for each training epoch, because the one-hot vectors result

in simple table lookups. With input smoothing, if the helper model's posterior probability distribution covers the whole vocabulary, this number increases to the empirical count of all tokens in the training corpus. At the output side, the non-target tokens also show up in the denominator in the softmax calculation, complicating things a bit. One can nonetheless calculate the partial derivatives of the local losses without and with output smoothing, $L_{i,\text{no smooth}}$ and $L_{i,\text{smooth}}$, on a certain non-target token e_ι, e.g. with zero-gram LM output smoothing shown in Equation 7.

$$
\begin{aligned}
L_{i,\text{no smooth}} &= -\log \frac{\exp(\tilde{e}_i^{\mathrm{T}} h)}{Z} \\
\frac{\partial L_{i,\text{no smooth}}}{e_\iota} &= \frac{h}{Z} \exp(\tilde{e}_\iota^{\mathrm{T}} h) \\
L_{i,\text{smooth}} &= -(1-m) \log \frac{\exp(\tilde{e}_i^{\mathrm{T}} h)}{Z} - \sum_{e \in E} \frac{m}{|E|} \log \frac{\exp(\tilde{e}^{\mathrm{T}} h)}{Z} \\
\frac{\partial L_{i,\text{smooth}}}{e_\iota} &= \frac{h}{Z} \exp(\tilde{e}_\iota^{\mathrm{T}} h) - \frac{m}{|E|} h
\end{aligned}
\tag{7}
$$

Here, h denotes the context vector for position i and Z denotes the denominator which is a sum over all logits. It is clear that for non-target output word embeddings, the magnitude of the gradient increases with an increasing m. In other words, output smoothing tends to update the non-target words more drastically compared to when no smoothing is applied. In summary, both input embeddings and output embeddings, including those embeddings of infrequent tokens, are more strongly updated with the smoothing methods, explaining why these smoothing methods work well in practice.

Finally, we plot the negative log-probabilities with respect to the frequency of words. In Figure 4, for frequent tokens as well as the infrequent tokens, our model is able to perform better than the Transformer baseline. Notice that the negative log-probabilities become more noisy as the tokens appear less often in the data, which is expected.

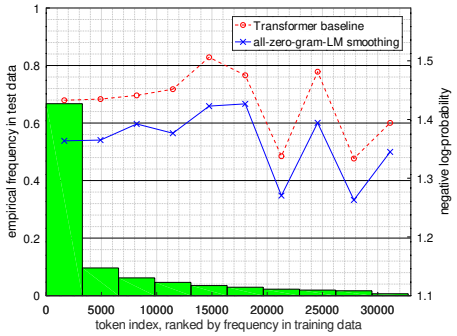


Figure 4: Scoring test tokens with Transformer baseline and all-zero-gram-LM smoothed model on `newstest2014`.

6 Conclusion

In this paper, we combine input smoothing and output smoothing for neural machine translation. We confirm strong cumulative improvements of the two methods, giving up to +1.9 BLEU scores in our experiments. Contrary to prior work, we find that simply smoothing everywhere with position-independant language models such as a zero-gram langauge model already gives significant improvements, setting us free from pre-training complex external helper models. We show that the improvements also sustain in case of back-translation. Finally, we explain that the key to the improvements lies in the increased update frequency of word vectors. Since the benefits almost come for free, we encourage more usage as well as more theoretical justifications of such smoothing methods, e.g. making symmetric smoothing work.

Acknowledgements

This work has received funding from the European Research Council (ERC) (under the European Union's Horizon 2020 research and innovation programme, grant agreement No 694537, project "SEQCLAS") and the Deutsche Forschungsgemeinschaft (DFG; grant agreement NE 572/8-1, project "CoreTec"). The GPU computing cluster was supported by DFG (Deutsche Forschungsgemeinschaft) under grant INST 222/1168-1 FUGG. Experiments were partially performed with computing resources granted by RWTH Aachen under project nova0003. The work reflects only the authors' views and none of the funding agencies is responsible for any use that may be made of the information it contains.

References

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. Findings of the 2019 conference on machine translation (WMT19). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels, October. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA, June. Association for Computational Linguistics.

Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10988–10998. Curran Associates, Inc.

Shanbo Cheng, Shaohui Kuang, Rongxiang Weng, Heng Yu, Changfeng Zhu, and Weihua Luo. 2020. Ar: Auto-repair the synthetic data for neural machine translation. *arXiv preprint arXiv:2004.02196*.

Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Sufeng Duan, Hai Zhao, Dongdong Zhang, and Rui Wang. 2020. Syntax-aware data augmentation for neural machine translation. *arXiv preprint arXiv:2004.14200*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November. Association for Computational Linguistics.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada, July. Association for Computational Linguistics.

Markus Freitag, Yaser Al-Onaizan, and Baskaran Sankaran. 2017. Ensemble distillation for neural machine translation. *arXiv preprint arXiv:1702.01802*.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy, July. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, and Yann Dauphin. 2017. A convolutional encoder model for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 123–135, Vancouver, Canada, July. Association for Computational Linguistics.

Miguel Graça, Yunsu Kim, Julian Schamper, Shahram Khadivi, and Hermann Ney. 2019. Generalizing back-translation in neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 45–52, Florence, Italy, August. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Matthias Huck. 2018. *Statistical Models for Hierarchical Phrase-based Machine Translation*. Ph.D. thesis, RWTH Aachen University, Computer Science Department, RWTH Aachen University, Aachen, Germany, August.

Kenji Imamura, Atsushi Fujita, and Eiichiro Sumita. 2018. Enhancement of encoder and attention using target monolingual corpora in neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 55–63, Melbourne, Australia, July. Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July. Association for Computational Linguistics.

Fred Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam.

Slava M. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 35:400–401.

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, November. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 181–184, Detroit, Michigan, USA, May.

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

Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. When does label smoothing help? *CoRR*, abs/1906.02629.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium, October. Association for Computational Linguistics.

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*.

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, July. Association for Computational Linguistics.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, Hong Kong, November. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany, August. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. 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, August. Association for Computational Linguistics.

Claude E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October.

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.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *International Conference on Learning Representations*.

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*, pages 5998–6008.

Shuo Wang, Yang Liu, Chao Wang, Huanbo Luan, and Maosong Sun. 2019. Improving back-translation with uncertainty-based confidence estimation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 791–802, Hong Kong, China, November. Association for Computational Linguistics.

Xing Wu, Yibing Liu, Xiangyang Zhou, and Dianhai Yu. 2020. Distilling knowledge from pre-trained language models via text smoothing. *arXiv preprint arXiv:2005.03848*.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *International Conference on Learning Representations*. 5th International Conference on Learning Representations, ICLR 2017 ; Conference date: 24-04-2017 Through 26-04-2017.