Subword regularization with segmentation candidates

Subword segmentation is potentially ambigious and multiple segmentations are possible even with the same vocabulary. That's why the authors propose to present the model training with multiple segmentation candidates to make the model more robust to noise and segmentation errors. Another improvement the authors come up with is a new segmentation algorithm that is based on a language model and produces segmentation candidates with probabilities. Having probabilities is important for this kind of data augmentation as it helps to narrow down the exponential number of possible segmentation combinations for longer sentences. The authors show that the subword regularization technique with multiple segmentation candidates is especially helpful in low ressource settings as well as open-domain settings. In their comparison they showed that models with subword regularization enabled had on average a BLEU score that was 1/2 points higher.

Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates

Taku Kudo
Google Inc.
taku@google.com

Abstract

Subword units are an effective way to alleviate the open vocabulary problems in neural machine translation (NMT). While sentences are usually converted into unique subword sequences, subword segmentation is potentially ambiguous and multiple segmentations are possible even with the same vocabulary. The question addressed in this paper is whether it is possible to harness the segmentation ambiguity as a noise to improve the robustness of NMT. We present a simple regularization method, subword regularization, which trains the model with multiple subword segmentations probabilistically sampled during training. In addition, for better subword sampling, we propose a new subword segmentation algorithm based on a unigram language model. We experiment with multiple corpora and report consistent improvements especially on low resource and out-of-domain settings.

1 Introduction

Neural Machine Translation (NMT) models (Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016; Vaswani et al., 2017) often operate with fixed word vocabularies, as their training and inference depend heavily on the vocabulary size. However, limiting vocabulary size increases the amount of unknown words, which makes the translation inaccurate especially in an open vocabulary setting.

A common approach for dealing with the open vocabulary issue is to break up rare words into subword units (Schuster and Nakajima, 2012; Chitnis and DeNero, 2015; Sennrich et al., 2016; Wu et al., 2016). Byte-Pair-Encoding (BPE) (Sennrich et al., 2016) is a de facto standard subword segmentation algorithm applied to many NMT systems and achieving top translation quality in several shared tasks (Denkowski and Neubig, 2017; Nakazawa et al., 2017). BPE segmentation gives a good balance between the vocabulary size and the decoding efficiency, and also sidesteps the need for a special treatment of unknown words.

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence "Hello World"

BPE encodes a sentence into a unique subword sequence. However, a sentence can be represented in multiple subword sequences even with the same vocabulary. Table 1 illustrates an example. While these sequences encode the same input "Hello World", NMT handles them as completely different inputs. This observation becomes more apparent when converting subword sequences into id sequences (right column in Table 1). These variants can be viewed as a spurious ambiguity, which might not always be resolved in decoding process. At training time of NMT, multiple segmentation candidates will make the model robust to noise and segmentation errors, as they can indirectly help the model to learn the compositionality of words, e.g., "books" can be decomposed into "book" + "s".

In this study, we propose a new regularization method for open-vocabulary NMT, called **subword regularization**, which employs multiple subword segmentations to make the NMT model accurate and robust. Subword regularization consists of the following two sub-contributions:

- We propose a simple NMT training algo-

rithm to integrate multiple segmentation candidates. Our approach is implemented as an on-the-fly data sampling, which is not specific to NMT architecture. Subword regularization can be applied to any NMT system without changing the model structure.

- We also propose a new subword segmentation algorithm based on a language model, which provides multiple segmentations with probabilities. The language model allows to emulate the noise generated during the segmentation of actual data.

Empirical experiments using multiple corpora with different sizes and languages show that subword regularization achieves significant improvements over the method using a single subword sequence. In addition, through experiments with out-of-domain corpora, we show that subword regularization improves the robustness of the NMT model.

2 Neural Machine Translation with multiple subword segmentations

2.1 NMT training with on-the-fly subword sampling

Given a source sentence X and a target sentence Y, let $\mathbf{x} = (x_1, \ldots, x_M)$ and $\mathbf{y} = (y_1, \ldots, y_N)$ be the corresponding subword sequences segmented with an underlying subword segmenter, e.g., BPE. NMT models the translation probability $P(Y|X) = P(\mathbf{y}|\mathbf{x})$ as a target language sequence model that generates target subword y_n conditioning on the target history $y_{<n}$ and source input sequence $\mathbf{x}$:

$$P(\mathbf{y}|\mathbf{x};\theta) = \prod_{n=1}^{N} P(y_n|\mathbf{x}, y_{<n};\theta), \quad (1)$$

where θ is a set of model parameters. A common choice to predict the subword y_n is to use a recurrent neural network (RNN) architecture. However, note that subword regularization is not specific to this architecture and can be applicable to other NMT architectures without RNN, e.g., (Vaswani et al., 2017; Gehring et al., 2017).

NMT is trained using the standard maximum likelihood estimation, i.e., maximizing the log-likelihood $\mathcal{L}(\theta)$ of a given parallel corpus $D = \{\langle X^{(s)}, Y^{(s)}\rangle\}_{s=1}^{|D|} = \{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)}\rangle\}_{s=1}^{|D|}$,

$$\theta_{MLE} = \arg\max_{\theta} \mathcal{L}(\theta)$$
$$where,\ \mathcal{L}(\theta) = \sum_{s=1}^{|D|} \log P(\mathbf{y}^{(s)}|\mathbf{x}^{(s)};\theta). \quad (2)$$

We here assume that the source and target sentences X and Y can be segmented into multiple subword sequences with the segmentation probabilities $P(\mathbf{x}|X)$ and $P(\mathbf{y}|Y)$ respectively. In subword regularization, we optimize the parameter set θ with the marginalized likelihood as (3).

$$\mathcal{L}_{marginal}(\theta) = \sum_{s=1}^{|D|} \mathbb{E}_{\substack{\mathbf{x}\sim P(\mathbf{x}|X^{(s)}) \\ \mathbf{y}\sim P(\mathbf{y}|Y^{(s)})}} [\log P(\mathbf{y}|\mathbf{x};\theta)] \quad (3)$$

Exact optimization of (3) is not feasible as the number of possible segmentations increases exponentially with respect to the sentence length. We approximate (3) with finite k sequences sampled from $P(\mathbf{x}|X)$ and $P(\mathbf{y}|Y)$ respectively.

$$\mathcal{L}_{marginal}(\theta) \cong \frac{1}{k^2} \sum_{s=1}^{|D|} \sum_{i=1}^{k} \sum_{j=1}^{k} \log P(\mathbf{y}_j|\mathbf{x}_i;\theta) \quad (4)$$
$$\mathbf{x}_i \sim P(\mathbf{x}|X^{(s)}), \quad \mathbf{y}_j \sim P(\mathbf{y}|Y^{(s)}).$$

For the sake of simplicity, we use $k = 1$. Training of NMT usually uses an online training for efficiency, in which the parameter θ is iteratively optimized with respect to the smaller subset of D (mini-batch). When we have a sufficient number of iterations, subword sampling is executed via the data sampling of online training, which yields a good approximation of (3) even if $k = 1$. It should be noted, however, that the subword sequence is sampled on-the-fly for each parameter update.

2.2 Decoding

In the decoding of NMT, we only have a raw source sentence X. A straightforward approach for decoding is to translate from the best segmentation $\mathbf{x}^*$ that maximizes the probability $P(\mathbf{x}|X)$, i.e., $\mathbf{x}^* = argmax_{\mathbf{x}} P(\mathbf{x}|X)$. Additionally, we can use the n-best segmentations of $P(\mathbf{x}|X)$ to incorporate multiple segmentation candidates. More specifically, given n-best segmentations $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, we choose the best translation $\mathbf{y}^*$ that maximizes the following score.

$$score(\mathbf{x}, \mathbf{y}) = \log P(\mathbf{y}|\mathbf{x})/|\mathbf{y}|^{\lambda}, \quad (5)$$

where $|\mathbf{y}|$ is the number of subwords in $\mathbf{y}$ and $\lambda \in \mathbb{R}^+$ is the parameter to penalize shorter sentences. λ is optimized with the development data.

In this paper, we call these two algorithms **one-best decoding** and **n-best decoding** respectively.

3 Subword segmentations with language model

3.1 Byte-Pair-Encoding (BPE)

Byte-Pair-Encoding (BPE) (Sennrich et al., 2016; Schuster and Nakajima, 2012) is a subword segmentation algorithm widely used in many NMT systems[1]. BPE first splits the whole sentence into individual characters. The most frequent[2] adjacent pairs of characters are then consecutively merged until reaching a desired vocabulary size. Subword segmentation is performed by applying the same merge operations to the test sentence.

An advantage of BPE segmentation is that it can effectively balance the vocabulary size and the step size (the number of tokens required to encode the sentence). BPE trains the merged operations only with a frequency of characters. Frequent substrings will be joined early, resulting in common words remaining as one unique symbol. Words consisting of rare character combinations will be split into smaller units, e.g., substrings or characters. Therefore, only with a small fixed size of vocabulary (usually 16k to 32k), the number of required symbols to encode a sentence will not significantly increase, which is an important feature for an efficient decoding.

One downside is, however, that BPE is based on a greedy and deterministic symbol replacement, which can not provide multiple segmentations with probabilities. It is not trivial to apply BPE to the subword regularization that depends on segmentation probabilities $P(\mathbf{x}|X)$.

3.2 Unigram language model

In this paper, we propose a new subword segmentation algorithm based on a unigram language model, which is capable of outputing multiple subword segmentations with probabilities. The unigram language model makes an assumption that each subword occurs independently, and consequently, the probability of a subword sequence $\mathbf{x} = (x_1, \ldots, x_M)$ is formulated as the product of the subword occurrence probabilities $p(x_i)$[3]:

$$P(\mathbf{x}) = \prod_{i=1}^{M} p(x_i), \quad (6)$$
$$\forall i \; x_i \in \mathcal{V}, \; \sum_{x \in \mathcal{V}} p(x) = 1,$$

where $\mathcal{V}$ is a pre-determined vocabulary. The most probable segmentation $\mathbf{x}^*$ for the input sentence X is then given by

$$\mathbf{x}^* = \arg\max_{\mathbf{x} \in \mathcal{S}(X)} P(\mathbf{x}), \quad (7)$$

where $\mathcal{S}(X)$ is a set of segmentation candidates built from the input sentence X. $\mathbf{x}^*$ is obtained with the Viterbi algorithm (Viterbi, 1967).

If the vocabulary $\mathcal{V}$ is given, subword occurrence probabilities $p(x_i)$ are estimated via the EM algorithm that maximizes the following marginal likelihood $\mathcal{L}$ assuming that $p(x_i)$ are hidden variables.

$$\mathcal{L} = \sum_{s=1}^{|D|} \log(P(X^{(s)})) = \sum_{s=1}^{|D|} \log\Big(\sum_{\mathbf{x} \in \mathcal{S}(X^{(s)})} P(\mathbf{x}) \Big)$$

In the real setting, however, the vocabulary set $\mathcal{V}$ is also unknown. Because the joint optimization of vocabulary set and their occurrence probabilities is intractable, we here seek to find them with the following iterative algorithm.

1. Heuristically make a reasonably big seed vocabulary from the training corpus.
2. Repeat the following steps until $|\mathcal{V}|$ reaches a desired vocabulary size.
   (a) Fixing the set of vocabulary, optimize $p(x)$ with the EM algorithm.
   (b) Compute the $loss_i$ for each subword x_i, where $loss_i$ represents how likely the likelihood $\mathcal{L}$ is reduced when the subword x_i is removed from the current vocabulary.
   (c) Sort the symbols by $loss_i$ and keep top η % of subwords (η is 80, for example). Note that we always keep the subwords consisting of a single character to avoid out-of-vocabulary.

[1] Strictly speaking, wordpiece model (Schuster and Nakajima, 2012) is different from BPE. We consider wordpiece as a variant of BPE, as it also uses an incremental vocabulary generation with a different loss function.

[2] Wordpiece model uses a likelihood instead of frequency.

[3] Target sequence $\mathbf{y} = (y_1, \ldots, y_N)$ can also be modeled similarly.

There are several ways to prepare the seed vocabulary. The natural choice is to use the union of all characters and the most frequent substrings in the corpus[4]. Frequent substrings can be enumerated in $O(T)$ time and $O(20T)$ space with the Enhanced Suffix Array algorithm (Nong et al., 2009), where T is the size of the corpus. Similar to (Sennrich et al., 2016), we do not consider subwords that cross word boundaries.

As the final vocabulary $\mathcal{V}$ contains all individual characters in the corpus, character-based segmentation is also included in the set of segmentation candidates $\mathcal{S}(X)$. In other words, subword segmentation with the unigram language model can be seen as a probabilsitic mixture of characters, subwords and word segmentations.

3.3 Subword sampling

Subword regularization samples one subword segmentation from the distribution $P(\mathbf{x}|X)$ for each parameter update. A straightforward approach for an approximate sampling is to use the l-best segmentations. More specifically, we first obtain l-best segmentations according to the probability $P(\mathbf{x}|X)$. l-best search is performed in linear time with the Forward-DP Backward-A* algorithm (Nagata, 1994). One segmentation $\mathbf{x}_i$ is then sampled from the multinomial distribution $P(\mathbf{x}_i|X) \cong P(\mathbf{x}_i)^\alpha / \sum_{i=1}^{l} P(\mathbf{x}_i)^\alpha$, where $\alpha \in \mathbb{R}^+$ is the hyperparameter to control the smoothness of the distribution. A smaller α leads to sample $\mathbf{x}_i$ from a more uniform distribution. A larger α tends to select the Viterbi segmentation.

Setting $l \to \infty$, in theory, allows to take all possible segmentations into account. However, it is not feasible to increase l explicitly as the number of candidates increases exponentially with respect to the sentence length. In order to exactly sample from all possible segmentations, we use the Forward-Filtering and Backward-Sampling algorithm (FFBS) (Scott, 2002), a variant of the dynamic programming originally introduced by Bayesian hidden Markov model training. In FFBS, all segmentation candidates are represented in a compact lattice structure, where each node denotes a subword. In the first pass, FFBS computes a set of forward probabilities for all subwords in the lattice, which provide the probability of ending up in any particular subword w. In the second pass, traversing the nodes in the lattice from the end of the sentence to the beginning of the sentence, subwords are recursively sampled for each branch according to the forward probabilities.

[4]It is also possible to run BPE with a sufficient number of merge operations.

3.4 BPE vs. Unigram language model

BPE was originally introduced in the data compression literature (Gage, 1994). BPE is a variant of dictionary (substitution) encoder that incrementally finds a set of symbols such that the total number of symbols for encoding the text is minimized. On the other hand, the unigram language model is reformulated as an entropy encoder that minimizes the total code length for the text. According to Shannon's coding theorem, the optimal code length for a symbol s is $-\log p_s$, where p_s is the occurrence probability of s. This is essentially the same as the segmentation strategy of the unigram language model described as (7).

BPE and the unigram language model share the same idea that they encode a text using fewer bits with a certain data compression principle (dictionary vs. entropy). Therefore, we expect to see the same benefit as BPE with the unigram language model. However, the unigram language model is more flexible as it is based on a probabilistic language model and can output multiple segmentations with their probabilities, which is an essential requirement for subword regularization.

4 Related Work

Regularization by noise is a well studied technique in deep neural networks. A well-known example is dropout (Srivastava et al., 2014), which randomly turns off a subset of hidden units during training. Dropout is analyzed as an ensemble training, where many different models are trained on different subsets of the data. Subword regularization trains the model on different data inputs randomly sampled from the original input sentences, and thus is regarded as a variant of ensemble training.

The idea of noise injection has previously been used in the context of Denoising Auto-Encoders (DAEs) (Vincent et al., 2008), where noise is added to the inputs and the model is trained to reconstruct the original inputs. There are a couple of studies that employ DAEs in natural language processing.

(Lample et al., 2017; Artetxe et al., 2017) independently propose DAEs in the context of

sequence-to-sequence learning, where they randomly alter the word order of the input sentence and the model is trained to reconstruct the original sentence. Their technique is applied to an unsupervised machine translation to make the encoder truly learn the compositionality of input sentences.

Word dropout (Iyyer et al., 2015) is a simple approach for a bag-of-words representation, in which the embedding of a certain word sequence is simply calculated by averaging the word embeddings. Word dropout randomly drops words from the bag before averaging word embeddings, and consequently can see $2^{|X|}$ different token sequences for each input X.

(Belinkov and Bisk, 2017) explore the training of character-based NMT with a synthetic noise that randomly changes the order of characters in a word. (Xie et al., 2017) also proposes a robust RNN language model that interpolates random unigram language model.

The basic idea and motivation behind subword regularization are similar to those of previous work. In order to increase the robustness, they inject noise to input sentences by randomly changing the internal representation of sentences. However, these previous approaches often depend on heuristics to generate synthetic noises, which do not always reflect the real noises on training and inference. In addition, these approaches can only be applied to source sentences (encoder), as they irreversibly rewrite the surface of sentences. Subword regularization, on the other hand, generates synthetic subword sequences with an underlying language model to better emulate the noises and segmentation errors. As subword regularization is based on an invertible conversion, we can safely apply it both to source and target sentences.

Subword regularization can also be viewed as a data augmentation. In subword regularization, an input sentence is converted into multiple invariant sequences, which is similar to the data augmentation for image classification tasks, for example, random flipping, distorting, or cropping.

There are several studies focusing on segmentation ambiguities in language modeling. Latent Sequence Decompositions (LSDs) (Chan et al., 2016) learns the mapping from the input and the output by marginalizing over all possible segmentations. LSDs and subword regularization do not assume a predetermined segmentation for a sentence, and take multiple segmentations by a similar marginalization technique. The difference is that subword regularization injects the multiple segmentations with a separate language model through an on-the-fly subword sampling. This approach makes the model simple and independent from NMT architectures.

Lattice-to-sequence models (Su et al., 2017; Sperber et al., 2017) are natural extension of sequence-to-sequence models, which represent inputs uncertainty through lattices. Lattice is encoded with a variant of TreeLSTM (Tai et al., 2015), which requires changing the model architecture. In addition, while subword regularization is applied both to source and target sentences, lattice-to-sequence models do not handle target side ambiguities.

A mixed word/character model (Wu et al., 2016) addresses the out-of-vocabulary problem with a fixed vocabulary. In this model, out-of-vocabulary words are not collapsed into a single UNK symbol, but converted into the sequence of characters with special prefixes representing the positions in the word. Similar to BPE, this model also encodes a sentence into a unique fixed sequence, thus multiple segmentations are not taken into account.

5 Experiments

5.1 Setting

We conducted experiments using multiple corpora with different sizes and languages. Table 2 summarizes the evaluation data we used [5] [6] [7] [8] [9] [10]. IWSLT15/17 and KFTT are relatively small corpora, which include a wider spectrum of languages with different linguistic properties. They can evaluate the language-agnostic property of subword regularization. ASPEC and WMT14 (en↔de) are medium-sized corpora. WMT14 (en↔cs) is a rather big corpus consisting of more than 10M parallel sentences.

We used GNMT (Wu et al., 2016) as the implementation of the NMT system for all experiments. We generally followed the settings and training procedure described in (Wu et al., 2016), however,

[5] IWSLT15: http://workshop2015.iwslt.org/
[6] IWSLT17: http://workshop2017.iwslt.org/
[7] KFTT: http://www.phontron.com/kftt/
[8] ASPEC: http://lotus.kuee.kyoto-u.ac.jp/ASPEC/
[9] WMT14: http://statmt.org/wmt14/
[10] WMT14(en↔de) uses the same setting as (Wu et al., 2016).

we changed the settings according to the corpus size. Table 2 shows the hyperparameters we used in each experiment. As common settings, we set the dropout probability to be 0.2. For parameter estimation, we used a combination of Adam (Kingma and Adam, 2014) and SGD algorithms. Both length normalization and converge penalty parameters are set to 0.2 (see section 7 in (Wu et al., 2016)). We set the decoding beam size to 4.

The data was preprocessed with Moses tokenizer before training subword models. It should be noted, however, that Chinese and Japanese have no explicit word boundaries and Moses tokenizer does not segment sentences into words, and hence subword segmentations are trained almost from unsegmented raw sentences in these languages.

We used the case sensitive BLEU score (Papineni et al., 2002) as an evaluation metric. As the output sentences are not segmented in Chinese and Japanese, we segment them with characters and KyTea[11] for Chinese and Japanese respectively before calculating BLEU scores.

BPE segmentation is used as a baseline system. We evaluate three test systems with different sampling strategies: (1) Unigram language model-based subword segmentation without subword regularization ($l=1$), (2) with subword regularization ($l=64,\ \alpha=0.1$) and (3) ($l=\infty,\ \alpha=0.2/0.5$) 0.2: IWSLT, 0.5: others. These sampling parameters were determined with preliminary experiments. $l=1$ is aimed at a pure comparison between BPE and the unigram language model. In addition, we compare one-best decoding and n-best decoding (See section 2.2). Because BPE is not able to provide multiple segmentations, we only evaluate one-best decoding for BPE. Consequently, we compare 7 systems ($1 + 3 \times 2$) for each language pair.

5.2 Main Results

Table 3 shows the translation experiment results.

First, as can be seen in the table, BPE and unigram language model without subword regularization ($l = 1$) show almost comparable BLEU scores. This is not surprising, given that both BPE and the unigram language model are based on data compression algorithms.

We can see that subword regularization ($l > 1$) boosted BLEU scores quite impressively (+1 to 2 points) in all language pairs except for WMT14 (en→cs) dataset. The gains are larger especially in lower resource settings (IWSLT and KFTT). It can be considered that the positive effects of data augmentation with subword regularization worked better in lower resource settings, which is a common property of other regularization techniques.

As for the sampling algorithm, ($l = \infty\ \alpha = 0.2/0.5$) slightly outperforms ($l = 64,\ \alpha = 0.1$) on IWSLT corpus, but they show almost comparable results on larger data set. Detailed analysis is described in Section 5.5.

On top of the gains with subword regularization, n-best decoding yields further improvements in many language pairs. However, we should note that the subword regularization is mandatory for n-best decoding and the BLEU score is degraded in some language pairs without subword regularization ($l = 1$). This result indicates that the decoder is more confused for multiple segmentations when they are not explored at training time.

5.3 Results with out-of-domain corpus

To see the effect of subword regularization on a more open-domain setting, we evaluate the systems with out-of-domain in-house data consisting of multiple genres: Web, patents and query logs. Note that we did not conduct the comparison with KFTT and ASPEC corpora, as we found that the domains of these corpora are too specific[12], and preliminary evaluations showed extremely poor BLEU scores (less than 5) on out-of-domain corpora.

Table 4 shows the results. Compared to the gains obtained with the standard in-domain evaluations in Table 3, subword regularization achieves significantly larger improvements (+2 points) in every domain of corpus. An interesting observation is that we have the same level of improvements even on large training data sets (WMT14), which showed marginal or small gains with the in-domain data. This result strongly supports our claim that subword regularization is more useful for open-domain settings.

5.4 Comparison with other segmentation algorithms

Table 5 shows the comparison on different segmentation algorithms: word, character, mixed

[11] http://www.phontron.com/kytea

[12] KFTT focuses on Wikipedia articles related to Kyoto, and ASPEC is a corpus of scientific paper domain. Therefore, it is hard to translate out-of-domain texts.

Corpus	Language pair	Size of sentences			Parameters		
		train	dev	test	#vocab (Enc/Dec shared)	#dim of LSTM embedding	#layers of LSTM (Enc+Dec)
IWSLT15	en ↔ vi	133k	1553	1268	16k	512	2+2
	en ↔ zh	209k	887	1261	16k	512	2+2
IWSLT17	en ↔ fr	232k	890	1210	16k	512	2+2
	en ↔ ar	231k	888	1205	16k	512	2+2
KFTT	en ↔ ja	440k	1166	1160	8k	512	6+6
ASPEC	en ↔ ja	2M	1790	1812	16k	512	6+6
WMT14	en ↔ de	4.5M	3000	3003	32k	1024	8+8
	en ↔ cs	15M	3000	3003	32k	1024	8+8

Table 2: Details of evaluation data set

Corpus	Language pair	baseline (BPE)	Proposed (one-best decoding)			Proposed (n-best decoding, $n=64$)		
			$l=1$	$l=64$ $\alpha=0.1$	$l=\infty$ $\alpha=0.2/0.5$	$l=1$	$l=64$ $\alpha=0.1$	$l=\infty$ $\alpha=0.2/0.5$
IWSLT15	en → vi	25.61	25.49	27.68*	27.71*	25.33	28.18*	28.48*
	vi → en	22.48	22.32	24.73*	26.15*	22.04	24.66*	26.31*
	en → zh	16.70	16.90	19.36*	20.33*	16.73	20.14*	21.30*
	zh → en	15.76	15.88	17.79*	16.95*	16.23	17.75*	17.29*
IWSLT17	en → fr	35.53	35.39	36.70*	36.36*	35.16	37.60*	37.01*
	fr → en	33.81	33.74	35.57*	35.54*	33.69	36.07*	36.06*
	en → ar	13.01	13.04	14.92*	15.55*	12.29	14.90*	15.36*
	ar → en	25.98	27.09*	28.47*	29.22*	27.08*	29.05*	29.29*
KFTT	en → ja	27.85	28.92*	30.37*	30.01*	28.55*	31.46*	31.43*
	ja → en	21.37	21.46	22.33*	22.04*	21.37	22.47*	22.64*
ASPEC	en → ja	40.62	40.66	41.24*	41.23*	40.86	41.55*	41.87*
	ja → en	26.51	26.76	27.08*	27.14*	27.49*	27.75*	27.89*
WMT14	en → de	24.53	24.50	25.04*	24.74	22.73	25.00*	24.57
	de → en	28.01	28.65*	28.83*	29.39*	28.24	29.13*	29.97*
	en → cs	25.25	25.54	25.41	25.26	24.88	25.49	25.38
	cs → en	28.78	28.84	29.64*	29.41*	25.77	29.23*	29.15*

Table 3: Main Results (BLEU(%)) (l: sampling size in SR, α: smoothing parameter). * indicates statistically significant difference ($p < 0.05$) from baselines with bootstrap resampling (Koehn, 2004). The same mark is used in Table 4 and 6.

word/character (Wu et al., 2016), BPE (Sennrich et al., 2016) and our unigram model with or without subword regularization. The BLEU scores of word, character and mixed word/character models are cited from (Wu et al., 2016). As German is a morphologically rich language and needs a huge vocabulary for word models, subword-based algorithms perform a gain of more than 1 BLEU point than word model. Among subword-based algorithms, the unigram language model with subword regularization achieved the best BLEU score (25.04), which demonstrates the effectiveness of multiple subword segmentations.

5.5 Impact of sampling hyperparameters

Subword regularization has two hyperparameters: l: size of sampling candidates, α: smoothing constant. Figure 1 shows the BLEU scores of various hyperparameters on IWSLT15 (en → vi) dataset.

First, we can find that the peaks of BLEU scores against smoothing parameter α are different depending on the sampling size l. This is expected, because $l = \infty$ has larger search space than $l = 64$, and needs to set α larger to sample sequences close to the Viterbi sequence $\mathbf{x}^*$.

Another interesting observation is that $\alpha = 0.0$ leads to performance drops especially on $l = \infty$. When $\alpha = 0.0$, the segmentation probability $P(\mathbf{x}|X)$ is virtually ignored and one segmentation is uniformly sampled. This result suggests that biased sampling with a language model is helpful to emulate the real noise in the actual translation.

In general, larger l allows a more aggressive regularization and is more effective for low resource settings such as IWSLT. However, the estimation of α is more sensitive and performance becomes even worse than baseline when α is extremely small. To weaken the effect of regularization and avoid selecting invalid parameters, it might be more reasonable to use $l = 64$ for high resource languages.

Domain (size)	Corpus	Language pair	Baseline (BPE)	Proposed (SR)
Web (5k)	IWSLT15	en → vi	13.86	17.36*
		vi → en	7.83	11.69*
		en → zh	9.71	13.85*
		zh → en	5.93	8.13*
	IWSLT17	en → fr	16.09	20.04*
		fr → en	14.77	19.99*
	WMT14	en → de	22.71	26.02*
		de → en	26.42	29.63*
		en → cs	19.53	21.41*
		cs → en	25.94	27.86*
Patent (2k)	WMT14	en → de	15.63	25.76*
		de → en	22.74	32.66*
		en → cs	16.70	19.38*
		cs → en	23.20	25.30*
Query (2k)	IWSLT15	en → zh	9.30	12.47*
		zh → en	14.94	19.99*
	IWSLT17	en → fr	10.79	10.99
		fr → en	19.01	23.96*
	WMT14	en → de	25.93	29.82*
		de → en	26.24	30.90*

Table 4: Results with out-of-domain corpus ($l = \infty$, $\alpha = 0.2$: IWSLT15/17, $l = 64$, $\alpha = 0.1$: others, one-best decding)

Model	BLEU
Word	23.12
Character (512 nodes)	22.62
Mixed Word/Character	24.17
BPE	24.53
Unigram w/o SR ($l = 1$)	24.50
Unigram w/ SR ($l = 64$, $\alpha = 0.1$)	25.04

Table 5: Comparison of different segmentation algorithms (WMT14 en→de)

Although we can see in general that the optimal hyperparameters are roughly predicted with the held-out estimation, it is still an open question how to choose the optimal size l in subword sampling.

5.6 Results with single side regularization

Table 6 summarizes the BLEU scores with subword regularization either on source or target sentence to figure out which components (encoder or decoder) are more affected. As expected, we can see that the BLEU scores with single side regularization are worse than full regularization. However, it should be noted that single side regularization still has positive effects. This result implies that subword regularization is not only helpful for encoder-decoder architectures, but applicable to other NLP tasks that only use an either encoder or decoder, including text classification

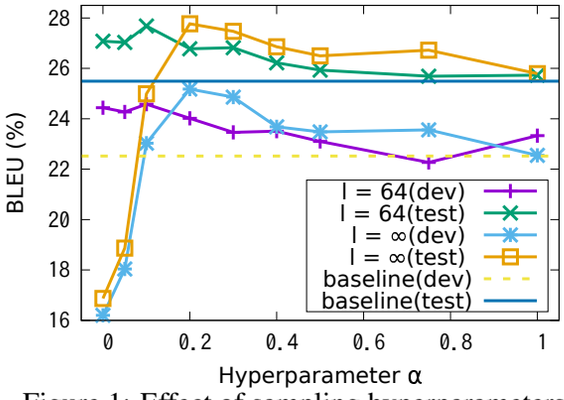

Figure 1: Effect of sampling hyperparameters

Regularization type	en→vi	vi→en	en→ar	ar→en
No reg. (baseline)	25.49	22.32	13.04	27.09
Source only	26.00	23.09*	13.46	28.16*
Target only	26.10	23.62*	14.34*	27.89*
Source and target	27.68*	24.73*	14.92*	28.47*

Table 6: Comparison on different regularization strategies (IWSLT15/17, $l = 64$, $\alpha = 0.1$)

(Iyyer et al., 2015) and image caption generation (Vinyals et al., 2015).

6 Conclusions

In this paper, we presented a simple regularization method, **subword regularization**[13], for NMT, with no change to the network architecture. The central idea is to virtually augment training data with on-the-fly subword sampling, which helps to improve the accuracy as well as robustness of NMT models. In addition, for better subword sampling, we propose a new subword segmentation algorithm based on the unigram language model. Experiments on multiple corpora with different sizes and languages show that subword regularization leads to significant improvements especially on low resource and open-domain settings.

Promising avenues for future work are to apply subword regularization to other NLP tasks based on encoder-decoder architectures, e.g., dialog generation (Vinyals and Le, 2015) and automatic summarization (Rush et al., 2015). Compared to machine translation, these tasks do not have enough training data, and thus there could be a large room for improvement with subword regularization. Additionally, we would like to explore the application of subword regularization for machine learning, including Denoising Auto Encoder (Vincent et al., 2008) and Adversarial Training (Goodfellow et al., 2015).

[13] Implementation is available at https://github.com/google/sentencepiece

References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXive preprint arXiv:1710.11041* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Yonatan Belinkov and Yonatan Bisk. 2017. Synthetic and natural noise both break neural machine translation. *arXive preprint arXiv:1711.02173* .

William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly. 2016. Latent sequence decompositions. *arXiv preprint arXiv:1610.03035* .

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proc. of EMNLP*. pages 2088–2093.

Michael Denkowski and Graham Neubig. 2017. Stronger baselines for trustable results in neural machine translation. *Proc. of Workshop on Neural Machine Translation* .

Philip Gage. 1994. A new algorithm for data compression. *C Users J.* 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* .

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proc. of ICLR*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. of ACL*.

Diederik P Kingma and Jimmy Ba Adam. 2014. A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*.

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXive preprint arXiv:1711.00043* .

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

Masaaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. In *Proc. of COLING*.

Toshiaki Nakazawa, Shohei Higashiyama, Chenchen Ding, Hideya Mino, Isao Goto, Hideto Kazawa, Yusuke Oda, Graham Neubig, and Sadao Kurohashi. 2017. Overview of the 4th workshop on asian translation. In *Proceedings of the 4th Workshop on Asian Translation (WAT2017)*. pages 1–54.

Ge Nong, Sen Zhang, and Wai Hong Chan. 2009. Linear suffix array construction by almost pure induced-sorting. In *Proc. of DCC*.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proc. of EMNLP*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *Proc. of ICASSP*.

Steven L Scott. 2002. Bayesian methods for hidden markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *Proc. of EMNLP*.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1).

Jinsong Su, Zhixing Tan, De yi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*. pages 3302–3308.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *Proc. of ACL* .

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

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory* 13(2):260–269.

Yonghui Wu, Mike Schuster, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

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 *Proc. of ICLR*.