How Much Does Tokenization Affect Neural Machine Translation?

Miguel Domingo[1], Mercedes García-Martínez[2], Alexandre Helle[2], Francisco Casacuberta[1], and Manuel Herranz[2]

[1] Pattern Recognition and Human Language Technology Research Center
Universitat Politècnica de València
Camino de Vera s/n, 46022 Valencia, Spain
{midobal, fcn}@prhlt.upv.es
[2] Pangeanic / B.I Europa
PangeaMT Technologies Division
Valencia, Spain
{m.garcia, a.helle, m.herranz}@pangeanic.com

Abstract. Tokenization or segmentation is a wide concept that covers simple processes such as separating punctuation from words, or more sophisticated processes such as applying morphological knowledge. Neural Machine Translation (NMT) requires a limited-size vocabulary for computational cost and enough examples to estimate word embeddings. Separating punctuation and splitting tokens into words or subwords has proven to be helpful to reduce vocabulary and increase the number of examples of each word, improving the translation quality. Tokenization is more challenging when dealing with languages with no separator between words. In order to assess the impact of the tokenization in the quality of the final translation on NMT, we experimented on five tokenizers over ten language pairs. We reached the conclusion that the tokenization significantly affects the final translation quality and that the best tokenizer differs for different language pairs.

1 Introduction

Segmentation is an essential process that has been extensively studied in literature (Nießen and Ney, 2004; Goldwater and McClosky, 2005; Dyer, 2009; Nguyen et al., 2010). It covers simple processes such as separating punctuation from words (tokenization), splitting words in subparts based on their frequency or more sophisticated processes such as applying morphological knowledge. In this work, we use tokenization referring to separating punctuation and splitting tokens into words or subwords.

Tokenizing words has proven to be helpful to reduce vocabulary and increase the number of examples of each word. It is extremely important for languages in which there is no separation between words and, therefore, a single token corresponds to more than one word. The way in which tokens are split can greatly change the meaning of the sentence. For example, the Japanese word 警 means *admonish*, and 察 means *observe*. However, together they form the word *police* (警察). Therefore, a correct tokenization can help to improve translation quality.

In this study, we aim to find the impact of tokenization on the quality of the final translation. To do so, we experimented with five tokenizers over ten language pairs. To

the best of our knowledge, this is the first work in which an exhaustive comparison between tokenizers has been run for NMT. We include tokenizers based on morphology that could guide the splitting of the words (Pinnis et al., 2017).

Some previous works include studying the effect of word-level preprocessing for Arabic on Statistical Machine Translation (SMT). A comparison of several segmenters for Chinese on SMT was done by Zhao et al. (2013). Huck et al. (2017) compared morphological segmenters for German in NMT. Finally, Kudo (2018a) compared his statistical word segmenter with other well-known Japanese morphological segmenters, reaching the conclusions that statistical segmenters worked better than morphological ones.

Our main contributions are as follows:

- First study of tokenizers for neural machine translation.
- Experimentation with five different tokenizers over ten language pairs.

The rest of this document is structured as follows: Section 2 introduces the neural machine translation system used in this work. After that, in Section 3, we present the tokenizers applied for comparison purposes. Then, in Section 4, we describe the experimental framework, whose results are presented and discussed in Section 5. Section 6 shows some translation examples of the results. Finally, in Section 7, conclusions are drawn.

2 Neural Machine Translation

Given a source sentence $x_1^J = x_1, \ldots, x_J$ of length J, NMT aims to find the best translated sentence $\hat{y}_1^{\hat{I}} = \hat{y}_1, \ldots, \hat{y}_{\hat{I}}$ of length $\hat{I}$:

$$\hat{y}_1^{\hat{I}} = \arg\max_{I, y_1^I} Pr(y_1^I \mid x_1^J) \tag{1}$$

where the conditional translation probability is modelled as:

$$Pr(y_1^I \mid x_1^J) = \prod_{i=1}^{I} Pr(y_i \mid y_1^{i-1}, x_1^J) \tag{2}$$

NMT frequently relies on a Recurrent Neural Network (RNN) encoder-decoder framework. The source sentence is projected into a distributed representation at the encoding step. Then, the decoder generates, at the decoding step, its translation word by word (Sutskever et al., 2014).

The input of the system is a word sequence in the source language. Each word is projected linearly to a fixed-size real-valued vector through an embedding matrix. Then, these word embeddings are fed into a bidirectional (Schuster and Paliwal, 1997) Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) network. As a result, a sequence of annotations is produced by concatenating the hidden states from the forward and backward layers.

An attention mechanism (Bahdanau et al., 2015) allows the decoder to focus on parts of the input sequence, computing a weighted mean of annotated sequences. A soft

alignment model computes these weights, weighting each annotation with the previous decoding state.

Another LSTM network is used for the decoder. This network is conditioned by the representation computed by the attention model and the last generated word. Finally, a distribution over the target language vocabulary is computed by the deep output layer (Pascanu et al., 2013).

The model is trained by applying stochastic gradient descent jointly to maximize the log-likelihood over a bilingual parallel corpus. At decoding time, the model approximates the most likely target sentence with beam-search (Sutskever et al., 2014).

3 Tokenizers

In this section, we present the tokenizers we employed in order to assess their impact on the quality of the final translation.

SentencePiece[3]: an unsupervised text tokenizer and detokenizer mainly for Neural Network-based text generation systems where the vocabulary size is predetermined prior to the neural model training. It can be used for any language, but its models need to be trained for each of them. To do so, we used the unigram (Kudo, 2018b) mode and a vocabulary size of 32000 over each corpora's training partition. Fig. 1a shows an example of tokenizing a sentence using *SentencePiece*.

Mecab[4]: an open source morphological analysis engine for Japanese, based on conditional random fields. It extracts morphological and syntactical information from sentences and splits tokens into words. Fig. 1b shows an example of tokenizing a sentence using *Mecab*.

Stanford Word Segmenter (Tseng et al., 2005): a Chinese word segmenter based on conditional random fields. Using a set of morphological and character reduplication features, it is able to split Chinese tokens into words. In this work, we use the toolkit's CTB scheme. Fig. 1c shows an example of tokenizing a sentence using *Stanford Word Segmenter*.

OpenNMT tokenizer (Klein et al., 2017): the tokenizer included with the *OpenNMT* toolkit. It normalizes characters (e.g., quotes Unicode variants) and separates punctuation from words. It can be used with any language. Fig. 1d shows an example of tokenizing a sentence using *OpenNMT tokenizer*.

Moses tokenizer (Koehn et al., 2007): the tokenizer included with the *Moses* toolkit. It separates punctuation from word —preserving special tokens such as URL or dates— and normalizes characters (e.g., quotes Unicode variants). It can be used with any language. Fig. 1e shows an example of tokenizing a sentence using *Moses tokenizer*.

4 Experimental Framework

In this section, we describe the corpora, systems and metrics used in order to asses our proposal.

[3] `https://github.com/google/sentencepiece`

[4] `http://taku910.github.io/mecab/`

Original: *In a browser window (Internet Explorer or Firefox) browse to www.dellconnect.com.*
Segmented: *_In _a _browser _window _(Internet _Explorer _or _Firefox) _browse _to _www . dell connect . com .*

(a) Example of a sentence tokenized using *SentencePiece*. _ indicates the start of a word in the original sentence. The tokenization has split punctuation and transformed the url into several words.

Original: ブラウザウィンドウ*(Internet Explorer*または*Firefox)*で、*www.dellconnect.com*にアクセスします。
Segmented: ブラウザウィンドウ *(Internet Explorer* または *Firefox)* で 、 *www . dellconnect . com* に アクセス し ます 。

(b) Example of a sentence tokenized using *Mecab*. In the original sentence, the only spaces were written to separate foreign words (*Internet Explorer*). The tokenization has added spaces between Japanese words, split the punctuation and transformed the url into several words.

Original: 到 *http://www.kace.com/trial*，然后“下 *K1000* 用版”，将的 *OVF*（放虚化格式）文件下到 *vSphere* 系。
Segmented: 到 *http : //www.kace.com/trial* ， 然后 “ 下 *K1000* 用版 ” ， 将 的 *OVF* （ 放 虚 化 格 式 ） 文件 下 到 *vSphere* 系 。

(c) Example of a sentence tokenized using *Stanford Word Segmenter*. The original sentence only contained spaces to separate foreign words (e.g., *vSphere*). The tokenization has added spaces between the Chinese words, split the punctuation, and separated the *http:* from the url.

Original: *In a browser window (Internet Explorer or Firefox) browse to www.dellconnect.com.*
Segmented: *In a browser window (Internet Explorer or Firefox) browse to www . dellconnect . com .*

(d) Example of a sentence tokenized using *OpenNMT tokenizer*. The tokenization has split punctuation and transformed the url into several words.

Original: *In a browser window (Internet Explorer or Firefox) browse to www.dellconnect.com.*
Segmented: *In a browser window (Internet Explorer or Firefox) browse to www.dellconnect.com.*

(e) Example of a sentence tokenized using *Moses tokenizer*. The tokenization has split the punctuation, without modifying the url.

Fig. 1: Examples of segmenting sentences with each word segmenter.

4.1 Corpora

The corpora selected for our experimental session was extracted from translation memories from the translation industry. The files are the result of professional translation tasks demanded by real clients. The general domain is technical (see Table 1 for the specific content of each language pair), which is harder for NMT than other general domains such as news. Unlike in other domains, in technical domains certain words correspond to specific terms and have a different translation to their most frequent one: e.g., *rear arm* translates into German as *hinterer Arm*. However, in this domain, it should be translated as *hinterer Querlenker*. In order to increase language diversity, we selected the following language-pairs: Japanese–English, Russian–English, Chinese–English, German–English, and Arabic–English. Table 2 shows the corpora statistics.

Specific Domain	**Language**				
	Ja–En	Ru–En	Zh–En	De–En	Ar–En
Computer Software - Instructions for use		X			X
Medical Equipment and Supplies	X	X	X	X	X
Consumer Electronics	X		X	X	X
Industrial Electronics		X		X	
Stores and Retail Distribution	X	X	X		
Healthcare		X			

Table 1: Specific domains for each language pair. *Ja* stands for Japanese, *En* for English, *Ru* for Russian, *Zh* for Chinese, *De* for German and *Ar* for Arabic.

The training dataset is composed of around three million sentences in the German–English language pair and around half a million sentences in the rest of the language pairs. Development and test datasets are composed of two thousand sentences for all the language pairs.

4.2 Systems

NMT systems were trained with *OpenNMT* (Klein et al., 2017). We used LSTM units taking into account the findings in (Britz et al., 2017). The size of the LSTM units and word embeddings were set to 1024. We used Adam (Kingma and Ba, 2014) with a learning rate of 0.0002 (Wu et al., 2016), a beam size of 6 and a batch size of 20. We reduced the vocabulary using Byte Pair Encoding (BPE) (Sennrich et al., 2016), training the models with a joint vocabulary of 32000 BPE units. Finally, the corpora were lowercased and, later, recased using *OpenNMT*'s tools.

4.3 Evaluation metrics

We made use of the following well-known metrics to assess our proposal:

Partition	Type	Language				
		Ja–En	Ru–En	Zh–En	De–En	Ar–En
Train	Sentences	532.0K	496.0K	460.8K	2.9M	557.0K
	Tokens	10.0/7.3M	7.6/7.4M	6.7/6.4M	35.9/39.4M	7.3/7.8M
	Vocabulary	41.5/111.6K	180.9/133.3K	82.8/102.6K	1.1M/615.7K	115.5/61.8K
	$\text{Tokens}_{\text{BPE}}$	10.5/8.3M	9.8/9.5M	7.5/7.4M	49.8/49.0M	8.4/8.7M
	$\text{Vocabulary}_{\text{BPE}}$	16.0/17.1K	24.8/11.6K	22.0/16.6K	25.6/22.3K	21.6/10.7K
Development	Sentences	2000	2000	2000	2000	2000
	Tokens	39.0/27.6K	34.0/32.2K	27.8/27.8K	42.4/45.4K	21.1/21.7K
	Vocabulary	2.3/3.4K	7.6/5.4K	2.7/3.8K	6.2/4.4K	3.6/2.9K
	$\text{Tokens}_{\text{BPE}}$	42.1/31.3K	41.2/38.5K	29.5/31.2K	53.7/51.0K	23.3/24.2K
	$\text{Vocabulary}_{\text{BPE}}$	1.9/2.5K	6.5/3.7K	2.5/2.9K	4.9/3.6K	3.4/2.1K
Test	Sentences	2000	2000	2000	2000	2740
	Tokens	18.4/26.8K	28.6/28.3K	48.7/30.5K	41.7/44.6K	22.1/23.3K
	Vocabulary	3.5/3.9K	7.3/5.1K	9.2/3.8K	6.0/4.3K	3.2/2.6K
	$\text{Tokens}_{\text{BPE}}$	39.5/30.2K	98.7/94.4K	32.9/35.6K	83.9/82.8K	34.4/32.9K
	$\text{Vocabulary}_{\text{BPE}}$	1.8/2.7K	8.0/5.4K	2.7/3.0K	8.3/6.8K	4.1/2.3K

Table 2: Corpora statistics. *Ja* stands for Japanese, *En* for English, *Ru* for Russian, *Zh* for Chinese, *De* for German and *Ar* for Arabic. $Tokens_{BPE}$ and $Vocabulary_{BPE}$ are the number of tokens and size of the vocabulary after applying BPE to the corpora. K stands for thousand and M for millions.

BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002): corresponds to the geometric average of the modified n-gram precision. It is multiplied by a brevity factor to penalize short sentences.

Translation Error Rate (TER) (Snover et al., 2006): number of word edit operations (insertion, substitution, deletion, and swapping), normalized by the number of words in the final translation.

Confidence intervals ($p = 0.05$) are computed for all metrics by means of bootstrap resampling (Koehn, 2004).

5 Results

In this section, we present the results of the experiments conducted in order to assess the impact of the tokenizer on the translation quality. Table 3 shows the experimental results.

For the Ja–En experiment, the best results were yielded by *Moses tokenizer* and *Mecab*. It must be taken into account that in both experiments, the English side of the corpus was segmented with *Moses tokenizer*, this means that the segmentation of the target side has a greater impact on the translation quality. Overall, there is a quality improvement of around 4 points in terms of BLEU and 3 points in terms of TER with respect to the tokenizer which yielded the second best results.

For En–Ja, the best results were yielded by *Mecab*, representing a significant improvement (around 12 points in terms of BLEU and 15 points in terms of TER) with

Language	SentencePiece		OpenNMT tokenizer		Moses tokenizer		Mecab		Stanford	
	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER	BLEU	TER
Ja–En	32.0 ± 1.3	51.1 ± 1.5	29.1 ± 1.4	54.7 ± 1.4	$\mathbf{36.3 \pm 1.4}$	$\mathbf{47.5 \pm 1.3}$	$\mathbf{36.0 \pm 1.5}$	$\mathbf{48.6 \pm 1.4}$	-	-
En–Ja	26.5 ± 1.4	62.5 ± 1.9	25.0 ± 4.4	89.9 ± 4.1	33.6 ± 2.3	61.0 ± 2.5	$\mathbf{45.8 \pm 1.3}$	$\mathbf{43.7 \pm 1.3}$	-	-
Ru–En	12.9 ± 0.9	72.7 ± 1.1	11.9 ± 0.9	74.9 ± 1.3	$\mathbf{15.3 \pm 1.0}$	$\mathbf{68.6 \pm 1.2}$	-	-	-	-
En–Ru	12.2 ± 0.8	75.0 ± 1.0	11.3 ± 0.9	77.3 ± 1.1	$\mathbf{16.3 \pm 1.2}$	$\mathbf{70.4 \pm 1.6}$	-	-	-	-
Zh–En	20.5 ± 1.1	64.8 ± 1.2	23.1 ± 1.3	64.8 ± 1.3	$\mathbf{27.5 \pm 1.3}$	$\mathbf{59.8 \pm 1.2}$	-	-	$\mathbf{26.0 \pm 1.3}$	$\mathbf{59.3 \pm 1.2}$
En–Zh	17.1 ± 1.2	71.2 ± 1.2	10.4 ± 3.9	101.1 ± 3.1	21.4 ± 2.0	65.8 ± 1.7	-	-	$\mathbf{29.9 \pm 1.2}$	$\mathbf{55.6 \pm 1.2}$
De–En	21.4 ± 0.8	67.8 ± 2.1	$\mathbf{29.6 \pm 0.9}$	$\mathbf{54.2 \pm 0.9}$	$\mathbf{30.3 \pm 0.9}$	$\mathbf{52.8 \pm 0.9}$	-	-	-	-
En–De	16.1 ± 0.7	76.4 ± 2.3	$\mathbf{22.5 \pm 0.9}$	$\mathbf{65.0 \pm 1.5}$	$\mathbf{23.6 \pm 0.9}$	$\mathbf{62.9 \pm 1.0}$	-	-	-	-
Ar–En	$\mathbf{17.9 \pm 0.8}$	$\mathbf{66.9 \pm 1.3}$	14.8 ± 0.8	71.3 ± 1.1	$\mathbf{19.1 \pm 0.9}$	$\mathbf{65.4 \pm 1.9}$	-	-	-	-
En–Ar	10.1 ± 0.6	75.3 ± 1.3	9.2 ± 0.6	77.2 ± 0.9	$\mathbf{12.4 \pm 0.7}$	$\mathbf{69.8 \pm 0.9}$	-	-	-	-

Table 3: Experimental results comparing the translation quality produced by using the different tokenizers. In the columns *Mecab* and *Stanford*, *Moses tokenizer* was used for segmenting the English part of the corpora since both *Mecab* and *Stanford Word Segmenter* only work for Japanese and Chinese respectively. Best results are denoted in bold.

respect to the tokenizer which yielded the second best results. Most likely, this is due to *Mecab* being developed specifically to segment Japanese.

For Ru–En and En–Ru, *Moses tokenizer* yielded the best results (with improvements of around 2 to 4 points in terms of BLEU and 5 points in terms of TER). It is worth noting that, in both cases, *SentencePiece* and *OpenNMT tokenizer* yielded similar results.

The Chinese experiments behaved similarly to the Japanese experiments: *Moses tokenizer* and *Stanford Word Segmenter* (the specific Chinese word tokenizer, which included using *Moses tokenizer* for segmenting the English part of the corpus) achieved the best results when translating to English (yielding an improvement of around 7 points in terms of BLEU and 5 points in terms of TER), and *Stanford Word Segmenter* achieved the best results when translating to Chinese (yielding an improvement of around 8 points in terms of BLEU and 20 points in terms of TER).

For the German experiments, the best results were yielded by both *OpenNMT tokenizer* and *Moses tokenizer*, representing an improvement of around 7 to 9 points in terms of BLEU and 14 to 17 points in terms of TER. It is worth noting how, despite being the largest corpora, *SentencePiece*—which learns how to segment from the corpora's training data—yielded the worst results. As a future study, we should evaluate the relation between the size of the corpora and the quality yielded by *SentencePiece*.

Finally, Arabic behaved similarly to Russian, with *Moses tokenizer* yielding the best results for both Ar–En and En–Ar (representing improvements of around 2 to 4 points in terms of BLEU and 4 to 6 points in terms of TER). However, *SentencePiece* performed similar to *Moses tokenizer* when translating to English. When translating to Arabic, both *SentencePiece* and *OpenNMT tokenizer* yielded similar results.

Overall, *Moses tokenizer* yielded the best results for German, Russian and Arabic experiments. When using specialized morphologically oriented tokenizers, the system using *Mecab* obtained the best results for Japanese experiments; and *Stanford Word Segmenter* for Chinese experiments. Additionally, *OpenNMT tokenizer* and *SentencePiece* yielded the worst translation quality in all experiments. An explanation for these

poor results is that *OpenNMT tokenizer* is fairly simple: it only separates punctuation symbols from words. However, this is not the case for *SentencePiece*. We think that using *SentencePiece* in a bigger training dataset in order to better learn the segmentation could help to improve their results. Nonetheless, as mentioned before, we have to corroborate this in a future work.

6 Qualitative Analysis

	Example 1
Source	Revalidation of single-pilot single-engine class ratings
Reference	verlängerung von klassenberechtigungen für einmotorige flugzeuge mit einem piloten
SentencePiece	**verlängerung** *der einzelantriebsklasse einmotorischer motorklasse*
OpenNMT tokenizer	*zur validierung der einmotorik-einzelmaschine* **mit** *einzelantrieb*
Moses tokenizer	**verlängerung von klassenberechtigungen für einmotorige flugzeuge mit einem piloten**
	Example 2
Source	Cold drawing of wire
Reference	herstellung von kaltgezogenem draht
SentencePiece	*kalt zeichnung des drahtes*
OpenNMT tokenizer	*kaltbildzeichnung*
Moses tokenizer	**herstellung von kaltgezogenem draht**

Table 4: English to German translation examples comparing *SentencePiece*, *OpenNMT tokenizer* and *Moses tokenizer*. First line corresponds to the source sentence in English, second line to the German reference and third, forth and fifth lines to the translations generated using *SentencePiece*, *OpenNMT tokenizer* and *Moses tokenizer* respectively to segment the corpora. Correct translations hypothesis are denoted in bold, and incorrect translations are denoted in italic.

We obtained a better performance using *Moses tokenizer* than *OpenNMT tokenizer* and *SentencePiece*. In order to qualitatively analyze this performance, Table 4 shows a couple of examples of translation outputs generated using *SentencePiece*, *OpenNMT tokenizer* and *Moses tokenizer* for segmenting the corpora.

The first example clearly shows a better performance when using *Moses tokenizer* rather than *SentencePiece*. The translation output from the system trained using *Moses tokenizer* for segmenting matches the reference. However, the output translations of the systems using *OpenNMT tokenizer* and *SentencePiece* are wrong. Translation segmented with *OpenNMT tokenizer* contains many repetitions and lacks sense. Additionally, translation segmented by *SentencePiece* has problems repeating some words in the translation (e.g., *motor*) and missing some translation words (e.g., the translation of *pilot*).

The system's behavior using *Moses tokenizer* in the second example is similar: its translation matches the reference. By contrast, the systems using *SentencePiece* and *OpenNMT tokenizer* translated wrongly. The system using *SentencePiece* translated all the words from the source but its translation is not grammatically correct. A correct translation could be *kalte Zeichnung des Drahtes*. Lastly, *OpenNMT tokenizer*'s performance is the worst in this case: the translation of its system ignored the word *wire*.

Therefore, we observed that, despite sharing the same data and model architecture, the behavior of the systems' translation changed as a result of using a different tokenizer.

7 Conclusions

In this study, we tested different tokenizers to evaluate their impact on the quality of the final translation. We experimented using 10 language pairs and arrived to the conclusion that tokenization has a great impact on the translation quality, achieving gains of up to 12 points of BLEU and 15 points of TER.

Additionally, we observed that there was not a single best tokenizer. Each one produced the best results for certain language pairs. Although, in some cases, those best results overlapped with the ones yielded by other tokenizers. Moreover, we have seen different behaviors depending on the language pair direction. The system using *SentencePiece* obtained the best results for Ar–En, but not for En–Ar translation.

As a future work, we would like to evaluate the relation between the size of the corpora and the quality yielded by *SentencePiece*—which uses each language's training corpora to learn how to segment. It would also be interesting to compare more segmentation strategies such as separating by characters or fixed n-grams. Finally, we would like to confirm that repeating these experiments on some of the general domain training data used for these languages achieves similar effects.

8 Acknowledgments

The research leading to these results has received funding from the Centro para el Desarrollo Tecnológico Industrial (CDTI) and the European Union through Programa Operativo de Crecimiento Inteligente (EXPEDIENT: IDI-20170964). We gratefully acknowledge the support of NVIDIA Corporation with the donation of a GPU used for part of this research.

Bibliography

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Britz, D., Goldie, A., Luong, T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. *arXiv preprint arXiv:1703.03906*.

Dyer, C. (2009). Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414.

Goldwater, S. and McClosky, D. (2005). Improving statistical MT through morphological analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 676–683.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huck, M., Riess, S., and Fraser, A. (2017). Target-side word segmentation strategies for neural machine translation. In *Proceedings of the Conference on Machine Translation*, pages 56–67.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *arXiv preprint arXiv:1701.02810*.

Koehn, P. (2004). Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 177–180.

Kudo, T. (2018a). Sentencepiece experiments. `https://github.com/google/sentencepiece/blob/master/doc/experiments.md`.

Kudo, T. (2018b). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 66–75.

Nguyen, T., Vogel, S., and Smith, N. A. (2010). Nonparametric word segmentation for machine translation. In *Proceedings of the International Conference on Computational Linguistics*, pages 815–823.

Nießen, S. and Ney, H. (2004). Statistical machine translation with scarce resources using morpho-syntactic information. *Computational linguistics*, 30(2):181–204.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2013). How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Pinnis, M., Krišlauks, R., Deksne, D., and Miks, T. (2017). Neural machine translation for morphologically rich languages with improved sub-word units and synthetic data. In *Proceedings of the International Conference on Text, Speech, and Dialogue*, pages 237–245.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the Association for Machine Translation in the Americas*, pages 223–231.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3104–3112.

Tseng, H., Chang, P., Andrew, G., Jurafsky, D., and Manning, C. (2005). A conditional random field word segmenter. In *Proceedings of the Special Interest Group of the Association for Computational Linguistics Workshop on Chinese Language Processing*, pages 168–171.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zhao, H., Utiyama, M., Sumita, E., and Lu, B.-L. (2013). An empirical study on word segmentation for chinese machine translation. In *Proceedings of the Computational Linguistics and Intelligent Text Processing*, pages 248–263.