Data Augmentation for Inline Tag-Aware Neural Machine Translation

Yonghyun Ryu, Yoonjung Choi, Sangha Kim
Samsung Research
Seoul, Republic of Korea
{yonghyun.ryu, yj0807.choi, sangha01.kim}@samsung.com

Abstract

Despite the wide use of inline formatting, not much has been studied on translating sentences with inline formatted tags. The detag-and-project approach using word alignments is one solution to translating a tagged sentence. However, the method has a limitation: tag reinsertion is not considered in the translation process. Another solution is to use an end-to-end model which takes text with inline tags as inputs and translates them into a tagged sentence. This approach can alleviate the problems of the aforementioned method, but there is no sufficient parallel corpus dedicated to such a task. To solve this problem, an automatic data augmentation method by tag injection is suggested, but it is computationally expensive and augmentation is limited since the model is based on isolated translation for all fragments. In this paper, we propose an efficient and effective tag augmentation method based on word alignment. Our experiments show that our approach outperforms the detag-and-project methods. We also introduce a metric to evaluate the placement of tags and show that the suggested metric is reasonable for our task. We further analyze the effectiveness of each implementation detail.

1 Introduction

While most machine translation studies are focused on plain text, the textual information that we encounter every day on the internet contains words with different styles and links within the sentence. Various styling of any part of the text is called inline formatting and is represented by markup and markdown tags. The inline formatting not only improves the readability of documents but also provides additional information with tags; so it is important to correctly translate sentences including tag information. In addition, the widespread use of formatting tags in the computer-based document system makes it inevitable to increase the demand for translating web text or structured documents containing inline tags.

There are two main approaches to translating segments with inline tags. One solution is the detag-and project method (Hanneman and Dinu, 2020). It first strips tags from the source sentence and translates only the plain text. Then, the removed tags are reinserted into the translation results using word alignments, which can be induced from attention weight in the model or an external aligner such as SimAlign (Jalili Sabet et al., 2020). This method does not take into account the re-insertion of tags in the translation process, making it difficult to restore tags at the proper positions.

Another way is to use an end-to-end model which takes sentences including tags as inputs and generates translation results with tags. Since tag information is considered, the translation can be performed with more context, thus this method potentially improves the quality of translation and the placement of tags. To train end-to-end models, a parallel corpus, where both source target sentences contain aligned tags, is required. Even though a parallel corpus with markup tags was released by Hashimoto et al. (2019), their data is limited to the domain of online help and there is still not many of such data available to train a high-quality model.

To address this lack of tagged parallel corpus, Hanneman and Dinu (2020) introduces a data augmentation approach using tag injection. Their method is to insert tags into corresponding fragments in the source and the target. In their approach, the aligned phrases are identified by an exhaustive search by matching all translated source fragments with all target fragments. This method has two drawbacks by its nature. The first is that their approach requires a high computational cost because it requires computing translation for all possible phrases for at least millions of parallel sentences to train a model. Secondly, only constrained tags can be augmented because they find corresponding pairs with out-of-context translation.

In this paper, we propose an efficient and effec-

Proceedings of the Seventh Conference on Machine Translation (WMT), pages 886 - 894

tive tag augmentation method using word alignments (Brown et al., 1993) to overcome the above shortcomings. Our method uses an external word aligner to compute correspondence between the source and target words, and find aligned fragments by phrase extraction algorithm (Och et al., 1999). Then tags are inserted according to the phrasal alignments. The tag-augmented parallel corpus by this method can train a model that translates sentence containing tags in an end-to-end way.

For comparisons, we implement competitive baselines and propose a metric to automatically evaluate the placement of tags. Through experiments, we show that our approach is superior to the detag-and-project methods and demonstrate the effectiveness of each implementation detail.

2 Method

In this section, we propose an efficient and effective method to insert inline tags into an existing parallel corpus. In augmented data, the position of tags in the source segment must be preserved in the target segment. The word "*preserved*" means that tags in the target sentence must surround spans with the same role and meaning as the corresponding source spans. In other words, the source and target fragment in the same tag has to correspond with each other. Moreover, inline tags can contain not only a word but also a phrase or even any consecutive words. Therefore, how to find corresponding phrase[1] pairs for each parallel sentence is the key to synthesizing tag-aligned parallel data. This makes our method focus on finding aligned phrase pairs.

Our proposed augmentation method consists of three steps. We first generate word alignments for the parallel corpus using external word aligners (Section 2.1). Then we extract aligned phrase pairs for each sentence pair with the word alignments (Section 2.2). Lastly, for each parallel sentence and the aligned phrase pairs, since each sentence usually has a lot more aligned pairs than the number of words in the sentence, we randomly select some of the pairs and insert tags to surround the phrases (Section 2.3). Figure 1 presents the whole process of our methods. The example is from Philipp Koehn's lecture[2].

[1]In this paper, the word "*phrase*" indicates consecutive words of any length. The length can be 1 and more.

[2]https://wiki.eecs.yorku.ca/course_archive/2014-15/W/6339/_media/esslli-slides-day3.pdf

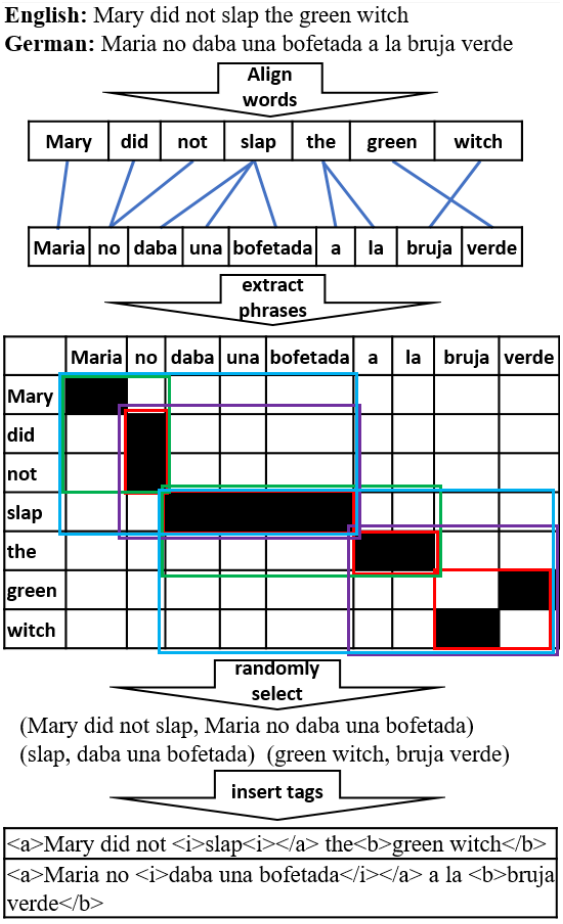

Figure 1: The process of our methods.

2.1 Word Alignment

Word alignment represents word-level correspondence in a parallel sentence. In statistical machine translation, implementation of IBM models (Brown et al., 1993) such as FastAlign (Dyer et al., 2013) and GIZA++ (Och and Ney, 2003) are famous to compute word alignment from parallel corpus. As deep neural network-based aligners, there are SimAlign (Jalili Sabet et al., 2020) and AwesomeAlign (Dou and Neubig, 2021). These neural methods use the similarity of contextual embeddings based on pretrained multilingual models to compute the correspondence between source and target words.

Our approach starts by using one of the above external word aligners to compute forward (source-to-target) and backward (target-to-source) word alignment, and then apply symmetrizing heuristics (Koehn et al., 2005) such as `grow-diag` and `grow-diag-final-and` for better alignment.

2.2 Phrase Extraction

The phrase level alignment has been proposed in Och et al. (1999) to improve statistical machine translation.

We find aligned phrase pairs depending on word alignments with phrase extraction algorithms. The phrase extraction algorithm finds phrasal alignments by exhaustively searching all phrase pairs that are consistent with word alignment. For the detailed algorithm description, please see the NLTK implementation (Loper and Bird, 2002)[3].

We do not use phrase probability tables (Koehn et al., 2003) to refine aligned pairs since it prevents the collection of diverse kinds of phrases. Instead, we do not allow phrases with unaligned words for more accurate phrase extraction[4].

2.3 Tag Insertion

In this step, we insert tags that surround aligned phrase pairs. The number of tags is randomly selected to less than 30% of the number of words. Then the aligned phrase pairs are randomly chosen as many as the number of tags, and tags are inserted according to the pairs. In this process, we can insert tags following the HTML syntax, and the tags can be nested.

2.4 Augmentation Cost Analysis

The cost of augmentation is crucial since machine translation models typically are trained on more than millions of parallel sentences. For this reason, we roughly analyze the amount of computation to show that our method is cost-efficient over the previous approach.

The previous tag augmentation method suggested in Hanneman and Dinu (2020) uses a machine translation model to find corresponding fragments with exhaustive search. Their method needs at least $O(n * m)$ translation model inferences for each parallel sentence, where n is the size of the maximum corresponding phrase length and m is the number of tokens in the source.

Assume that our method use SimAlign (Jalili Sabet et al., 2020) for a word aligner. Our approach requires one XLM-R (Conneau et al., 2020) inference to compute contextual word embeddings and single matrix multiplication to get cosine similarities between tokens. Since single XLM-R inference cost much to single matrix multiplication, we can count single XLM-R inference as the amount of computation. Alignment symmetrizing heuristic algorithms and phrase extraction are also required for our approach, but we do not count it to time comparison since these algorithms also take much less time than neural model inference.

Because the translation model uses beam search, both XLM-R and a translation model have almost the same computation cost, but the translation has more latency because it is autoregressive. In short, the previous method needs $O(n * m)$ translation model inference but our approach only requires a single XLM-R inference for each sentence pair. Therefore, our model is more efficient than translation-based augmentation.

3 Experimental Setup

3.1 Data

3.1.1 Training Data

Our data augmentation goal is to train an end-to-end model to translate inline tagged text with competitive translation quality. For a fair comparison of translation performance, we use the same training and test sets as Edunov et al. (2018) and Garg et al. (2019). The tagged parallel corpus released by Hashimoto et al. (2019) is also used to evaluate the placement of tags.

WMT'18 This dataset is a set of parallel corpora for the WMT'18 English-German news translation task (Bojar et al., 2018) and consists of the Europarl v7, common crawl, news commentary v13, and rapid corpus of EU press releases. Parallel sentences with either a sentence longer than 250 tokens or a source/target token length ratio exceeding 1.5 are removed[5].

LXM In this paper, we call the dataset released by Hashimoto et al. (2019) **LXM** which is their GitHub repository name's initials[6]. The data have parallel sentences with aligned inline tags. For German-to-English, there are about 100,000 train pairs and 2,000 development pairs. Only about a quarter of them contain tags.

[3] https://www.nltk.org/_modules/nltk/translate/phrase_based.html

[4] The original phrase extraction implemented in the NLTK includes unaligned words in the aligned phrase since it is still considered to be consistent with word alignment.

[5] We use the XLM-R tokenizer to filter the parallel corpus.

[6] https://github.com/salesforce/localization-xml-mt

LXM-plain This data is the **LXM** training data without tagged pairs. Since the domain of LXM is online help and WMT'18 corpora do not cover them, thus we add this data to training data. In this paper, we prove the effectiveness of the tag augmentation approach, thus we only use plain sentences from the training set.

3.1.2 Test Data

For comparison of our approach to the previous works, we use newstest2014 (**WMT'14**) to evaluate translation quality and LXM development set (**LXM-dev**) for the accuracy of tag placement.

3.2 Naive End-to-end Baseline

This baseline takes text with markup tags as inputs and handles them like plain text but uses a model which has been trained without tagged parallel corpus.

3.3 Detag-and-project Baselines

The detag-and-project approach (Hanneman and Dinu, 2020) first strips tags from the source sentence, translates the plain one, and then places the removed tags in the corresponding positions according to the word alignments. During the projection stage, one tag can be projected into separated parts, in this case, we insert one minimum-sized tag that surrounds all of the parts, which is also called the Min-Max Tag Pair Projection in Zenkel et al. (2021).

We establish three detag-and-project baselines according to the way to get word alignment.

Layer Average Baseline The layer average baseline is to extract word alignments from the attention. There are two methods to induce word alignments from the attention (Chen et al., 2020): NAIVE-ATT and SHIFT-ATT. Word alignments are induced from attention scores between the encoder and decoder. NAIVE-ATT (Garg et al., 2019) relates the maximum attention scores with the decoder's output token and uses attention weight of the penultimate layer of the decoder. SHIFT-ATT (Chen et al., 2020) associates the maximum attention scores with the decoder's input token and uses attention weight of the third layer of the decoder.

Garg et al. (2019) This baseline can be simply called attention enhanced approach. Like the layer average baseline, this method also extracts word alignments from attention scores, but it uses the trained attention head by multi-task learning. Specifically, one attention head of the fifth layer of the decode is jointly trained with translation by word alignments from GIZA++ (Och and Ney, 2003). Furthermore, full target context is used when the attention weight learns word alignments and predicts alignments from cross-attention. We re-implement the model by the author's Fairseq (Ott et al., 2019) implementation[7] to reproduce their results. In this paper, we do not apply any pre-tokenizer, and only use an unigram language model tokenizer (Kudo, 2018) with a vocabulary size of 35,000. All other hyperparameters are the same as Garg et al. (2019).

SimAlign This method uses SimAlign (Jalili Sabet et al., 2020) as an external aligner to restore tags on the translation results. For this model, the layer average baseline model is used to generate translated sentences. We take **argmax**[8] function to extract each direction of word alignment and apply `grow-diag-final-and` heuristics (Koehn et al., 2005) to symmetrize the bidirectional alignments for better word alignments.

3.4 Implementation Details

Reversible Tokenization The previous works use Moses tokenizer (Koehn et al., 2007) as a pre-tokenizer before applying Byte-Pair-Encoding (Sennrich et al., 2016). However, we don't use any pre-tokenizers like Moses because it is impossible to detokenize tokenized results to the original sentence completely even if a well-designed rule-based detokenizer is applied. We only apply SentencePiece (Kudo and Richardson, 2018) for tokenization, because it is a reversible tokenizer and makes a purely end-to-end system possible. A unigram language model tokenizer (Kudo, 2018) is trained from the WMT'18 corpus only without applying subword regularization.

Whitespace Shift As SentencePiece (Kudo and Richardson, 2018) tokenizer adds dummy whitespace at the beginning of a sentence, we move the whitespace before the tag to the back of the tag since the whitespace at the beginning of a word plays an important role in tokenization because the whitespace is also considered a target to tokenize by the subword tokenizer. A word without whitespace at the beginning is often tokenized differently from a word with a whitespace. For example, "World" is

[7] https://github.com/facebookresearch/fairseq/tree/main/examples/joint_alignment_translation

[8] Argmax aligns words to the most similar word.

	WMT'14	LXM-dev				
Model	**BLEU**	**BLEU**	**XML BLEU**	**XML Acc.**	**XML Match**	**F1**
Edunov et al. (2018)	29.0					
Hashimoto et al. (2019)		52.91	51.16	99.75	99.3	
Naive End-to-end						
- WMT'18 only	28.7	25.12	22.14	98.05	95.15	44.83
- WMT'18 + LXM-plain	28.8	51.05	49.9	98.6	98.2	58.93
Layer Average Baseline						
- NAIVE-ATT	28.8	52.22	50.45	100	98.5	60.53
- SHIFT-ATT	28.8	52.22	50.71	100	98.75	61.59
Garg et al. (2019)	28.7	52.46	50.64	100	98.0	68.04
SimAlign	28.8	52.22	48.43	100	97.55	60.96
Tag Augmentation (ours)	29.1	53.37	52.8	100	99.35	74.31
Tag Shift	29.1	53.37	52.07	100	99.35	53.75

Table 1: Evaluation results on the WMT'14 and LXM-dev. Models are trained with WMT'18 and LXM-plain by default. In *Tag Shift*, all tags are moved to one word to the left in the translation hypothesis.

tokenized into "Wo", "r", "ld", however, "_World" is tokenized into "_Wor", "ld". This inconsistency affects adversely translation quality. For this reason, we move the space and put it back in the pre- and post-processing step.

Tag Replacement Markup tags often have attributes and the attributes generally don't need to be translated and just copied to the translation. Like other approaches (Müller, 2017) and (Hanneman and Dinu, 2020), we replace the real tags with indexed special tags. We insert at most 9 tags per each parallel pair in tag augmentation. In our implementation, we use "<a_0>,<a_1>,...<a_9>,</a_0>,</a_1>,...</a_9>" as special tokens. In the training step, the index of special tokens is shuffled for training efficiency. In the inference, we convert real tags to the special tokens and the convert table in the pre-processing step. After translation, we revert them to the original tags in post-processing.

Tag Augmentation Hyperparmeters We use subword alignments instead of word alignments since according to recent studies (Garg et al., 2019) (Jalili Sabet et al., 2020), and (Dou and Neubig, 2021); subword-based alignments outperform word alignments on AER (Alignment Error Rate). Since our SimAlign baseline uses the XLM-R model (Conneau et al., 2020), for a fair comparsion, parallel corpus is tokenized by the XLM-R tokenizer[9] before computing word alignment with statistical models.

Like Garg et al. (2019), for our augmentation, Giza++ with 5 iterations of IBM1, HMM, IBM3 and IBM4 are used as a word aligner. However, for training an end-to-end model, we use a different tokenizer as explained in 3.4. For end-to-end training, we use the combination of tagged data and plain data in a 1:1 ratio.

Model Training Hyperparameters Basically for all experiments, we follow the same hyperparameters as the Align and Translate Task of Garg et al. (2019). We use the fairseq toolkit (Ott et al., 2019) for all of our experiments. The big transformer architecture[10] with the post layer normalization is used for all experiments. The difference is that we use learning rate of 5e-4, learning rate warmup over the first 8000 steps, and a batch size of 32768 tokens[11] on 8 A100 GPUs for 120k updates. We use the checkpoint which averages the last 10 checkpoints, and a beam size of 5 for inference.

4 Evaluation

In this section, we describe several metrics to evaluate our methods and present experimental results.

4.1 Evaluation Metrics

For comparison of translation quality to Garg et al. (2019) and Edunov et al. (2018), we use

[9]They use SentencePiece tokenizer and the model can download in https://github.com/facebookresearch/XLM.

[10]The architecture name we used in faisreq is `trasnformer_wmt_en_de_big`.

[11]Actually, we use 16384 tokens with accumulating 2 update gradients.

	WMT'14	LXM-dev			
Variation	**BLEU**	**BLEU**	**XML BLEU**	**XML Match (Acc.)**	**F1**
Baseline	29.1	53.37	52.8	99.35	74.31
w/o Tag Replacement					
trained on plain corpus	28.8	51.05	49.9	98.2 (98.6)	58.93
trained on tagged corpus	28.7	52.44	51.62	99.05 (99.85)	71.14
Symmetric Heuristics					
`intersection`	29.2	53.13	52.4	99.2	60.13
`grow`	28.9	53.21	52.52	99.2	74.73
`grow-diag-final-and`	29.0	53.11	52.39	99.2	75.35
Phrase Length					
8	28.8	53.33	51.9	99.05	72.07
16	28.9	53.17	52.86	99.45	73.61
32	29.1	52.85	52.25	99.25	73.12
128	28.9	52.92	52.55	99.45	72.84
Word Aligner					
Fast-Align	29.0	52.89	52.14	99.2	72.99
SimAlign	28.9	52.99	52.41	99.15	73.5
w/o whitespace shift	28.6	52.88	52.26	99.4	71.77
NLTK phrase extraction	28.9	52.85	51.96	99.3	72.48
Violating HTML syntax	29.1	53.09	52.29	99.2	72.58
Tagged data only	28.5	52.59	51.66	99.35	72.27

Table 2: Variations on tag augmentation. The baseline uses `grow-diag` heuristics, phrase length of 64, GIZA++ as a word aligner, and improved phrase extraction. The score on XML Accuracy is not mentioned because all scores are 100. *w/o whitespace shift* do not apply whitespace shift in the training and inference.

sacreBLEU (Post, 2018) with **WMT'14**. Like Hashimoto et al. (2019)'s work, we use BLEU, XML BLEU, XML Accuracy, and XML Match as metrics in the evaluation of LXM-dev. We also use the same evaluation scripts as they do[12]. For accurate evaluation of the tag placement, we introduce an F1 score-based metric to evaluate the position of tags by focusing on the words that tags surround.

BLEU and XML BLEU The BLEU score here is the same as the existing BLEU score measured in plain text. For that, all tags first are removed if exist, and then the BLEU is measured using the same tokenizer as Hashimoto et al. (2019). The XML BLEU uses the same metric, but if there are tags, the BLEU score is measured with text containing XML tags. The XML tags are also considered to compute the score.

XML Accuracy and Match The XML accuracy is the ratio of the valid XML outputs in all translation results. The XML match is the ratio of the outputs that have the same XML structure as the reference.

F1 score This metric is introduced to evaluate the placement of tags. Since the goal of tag transfer is to surround the corresponding words accurately, we introduce a metric to focus on evaluating words surrounded by tags. In this sense, we make use of the metric from SQuAD (Rajpurkar et al., 2016), since it evaluates the words in the span. SQuAD's answers consist of a span of consecutive words in a paragraph and they evaluate how accurately the span contains the correct answer. Since what we really want to evaluate is not the position of tags but the content of the span surrounded by tags, the goal of their evaluation is similar to ours in that they aim to assess a range of words.

We apply this metric to evaluate the accuracy of tag placement. The score measures the overlap between the ground truth and the prediction to calculate a score. More precisely, they treat the hypothesis and the reference as bags of words, and calculate F1. Unlike SQuAD dataset, LXM-dev can have more than one tag for each sentence, thus

[12] https://github.com/salesforce/localization-xml-mt

Model	Alignment Error Rate (AER)					
Method	**SHIFT-ATT**			**NAIVE-ATT**		
Layer	**1**	**2**	**3**	**4**	**5**	**6**
Layer Average Baseline	**29.1**	31.8	36.1	41.9	42.8	51.2
Tag Augmentation (ours)	27.9	**26.4**	49.6	37.8	35.7	44.7
Garg et al. (2019) (all heads)	32.0	26.0	22.7	35.3	29.1 (**20.5**)*	72.2

Table 3: Results on Vilar et al. (2006). * uses the first head trained by word alignments. Others use the average. While we apply SHIFT-ATT for the half bottom layers, we apply NAIVE-ATT on the top 3 layers for better performance on AER (Chen et al., 2020).

we use the average score per tag[13].

4.2 Results

Firstly, in order to show the relevance of the proposed F1 metric, we shift all tags to the left by one word, which must cause performance degradation in tag placement. In the results of *Tag Shift* in Table 1, compared to *Tag Augmentation*, there is only a slight drop on XML BLEU, however, the F1 score shows a significant decrease. This implies that the proposed metric is reasonable to evaluate the placement of tags.

In Table 1, there are the results of the baselines and our tag augmentation method. The experimental results show that our augmentation method achieves the best performance for all metrics. Furthermore, according to the XML Accuracy, the tag augmentation model is able to generate all XML tags grammatically correctly in the source without XML-constrained beam search.

4.3 Augmentation Variation

We conduct various experiments to figure out what greatly affects the performance. Firstly, we note that tag augmentation with intersection heuristics causes considerable degradation on the f1 score. We also note that according to (Dou and Neubig, 2021), the performance of word alignments between Fast-Align and Giza++ is considerable, but the models trained on each data show relatively similar performance compared to the AER scores.

Even though there is a little gap in performance, the result indicates that all of our proposed implementation details have a positive influence on both translation quality and tag placement. As a result, performance improvement is achieved by all factors combined.

4.4 Indirect Learning Alignment

We further investigate the effect of the aligned tagged corpus. Table 3 shows that the AER score from all layers is improved than the *Layer Average Baseline*, but does not reach the score of multi-task training model (Garg et al., 2019) where word alignments are trained directly. This result indicates that the tag-augmented data help models' attention to learn the correspondence between source and target words indirectly.

5 Conclusion

In this paper, we have presented an efficient and effective inline tag augmentation method to insert tags into existing parallel corpora using a word aligner and the phrase extraction algorithm. Our approach injects inline tags economically and accurately.

We also introduced a reasonable metric for the automatic and accurate evaluation of the placement of tags and analyzed the effectiveness of the detailed methods used in our approach. The experiment results show that the model trained on data augmented by our method outperforms the previous detag-and-project methods.

[13]Unfortunately, some sentences in LXM-dev have multiple of the same name tags in a sentence. Because there is no way to align the same name tags, we regard the multiple separate spans with the same name as one consecutive span in the evaluation.

References

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, 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. Association for Computational Linguistics.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter

estimation. *Computational Linguistics*, 19(2):263–311.

Yun Chen, Yang Liu, Guanhua Chen, Xin Jiang, and Qun Liu. 2020. Accurate word alignment induction from neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 566–576, Online. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Zi-Yi Dou and Graham Neubig. 2021. Word alignment by fine-tuning embeddings on parallel corpora. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2112–2128, Online. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. 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 4453–4462, Hong Kong, China. Association for Computational Linguistics.

Greg Hanneman and Georgiana Dinu. 2020. How should markup tags be translated? In *Proceedings of the Fifth Conference on Machine Translation*, pages 1160–1173, Online. Association for Computational Linguistics.

Kazuma Hashimoto, Raffaella Buschiazzo, James Bradbury, Teresa Marshall, Richard Socher, and Caiming Xiong. 2019. A high-quality multilingual dataset for structured documentation translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers)*, pages 116–127, Florence, Italy. Association for Computational Linguistics.

Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. 2020. SimAlign: High quality word alignments without parallel training data using static and contextualized embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1627–1643, Online. Association for Computational Linguistics.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the Second International Workshop on Spoken Language Translation*, Pittsburgh, Pennsylvania, USA.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Mathias Müller. 2017. Treatment of markup in statistical machine translation. In *Proceedings of the Third Workshop on Discourse in Machine Translation*, pages 36–46, Copenhagen, Denmark. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

David Vilar, Maja Popovic, and Hermann Ney. 2006. AER: do we need to "improve" our alignments? In *Proceedings of the Third International Workshop on Spoken Language Translation: Papers*, Kyoto, Japan.

Thomas Zenkel, Joern Wuebker, and John DeNero. 2021. Automatic bilingual markup transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3524–3533, Punta Cana, Dominican Republic. Association for Computational Linguistics.