Novel Applications of Factored Neural Machine Translation

Patrick Wilken, Evgeny Matusov

AppTek
Aachen, Germany
pwilken@apptek.com, ematusov@apptek.com

Abstract

In this work, we explore the usefulness of target factors in neural machine translation (NMT) beyond their original purpose of predicting word lemmas and their inflections, as proposed by García-Martínez et al. [1]. For this, we introduce three novel applications of the factored output architecture: In the first one, we use a factor to explicitly predict the word case separately from the target word itself. This allows for information to be shared between different casing variants of a word. In a second task, we use a factor to predict when two consecutive subwords have to be joined, eliminating the need for target subword joining markers. The third task is the prediction of special tokens of the operation sequence NMT model (OSNMT) of Stahlberg et al. [2]. Automatic evaluation on English → German and English → Turkish tasks showed that integration of such auxiliary prediction tasks into NMT is at least as good as the standard NMT approach. For the OSNMT, we observed a significant improvement in BLEU over the baseline OSNMT implementation due to a reduced output sequence length that resulted from the introduction of the target factors.

1. Introduction

The state-of-the-art in machine translation are encoder-decoder neural network models [3,4]. The encoder network maps the source sentence into a vector representation, while the decoder network outputs target words based on the input from the encoder. In the standard approach, the decoder generates one output per step that directly corresponds to a single target token. In this work, we explore applications of decoder architectures with multiple outputs per decoding step. Such an architecture was introduced for factored neural machine translation (FNMT) [1] to separate the prediction of word lemmas from morphological factors such as gender and number. The main motivation of the authors was to reduce the problem of out-of-vocabulary words. While in state-of-the-art systems this problem is tackled by the use of subword units [5,6], which in theory allow for an open vocabulary, we argue that FNMT still has the following potential advantages over standard NMT:

1. Factorization can be applied on top of subword splitting. This increases the effective number of subwords for a given vocabulary size, allowing for more words to be kept in their original form.
2. Factorization can be used to explicitly share information between related (sub)words.
3. FNMT can be used to produce several tokens of the target sequence at the same time, decreasing the decoding sequence length and increasing decoding speed.

To demonstrate the first two advantages, we use FNMT to pool subwords that only differ in casing or in the presence of a joining marker. Factors are used to make the corresponding casing or joining decisions (Section 5). The third advantage is shown for the example of neural operation sequences [2]. Here, we factor out the prediction of special tokens that move the read head to reduce the number of decoding steps.

2. Related Work

The term factored translation first appeared in the context of statistical machine translation (SMT). The Moses system [7] included implementation of source and target factors. They were used to either consume or generate different sets of label sequences: full word surface forms, lemmas, part-of-speech tags, morphological tags [8]. The original motivation was to reduce the data sparseness and estimation problems for the phrase-level translation models and target language models. Factored models were most effectively used in SMT for translation into morphologically rich languages, as shown e.g. in the work of [9].

With the rapid emergence of NMT, a factored neural translation model was introduced by [1]. In that work, the out-of-vocabulary and large vocabulary size problems are mitigated by generating lemma and morphological tags. Our work is a re-implementation of that model, but we apply it for a different cause.

Word case prediction using a count-based factored translation model was done in [10], as an alternative to restoring case information in post-processing using a language model [11]. Another common alternative is truecased training, where however casing variants of a word are handled as distinct tokens. In our work, we evaluate case prediction as part of a neural MT system, which allows for an unbound context of previous target words and casing decisions.
Translation using subword units is a very popular vocabulary reduction technique, first proposed for NMT by [5]. To the best of our knowledge, in previous work the information on how to restore full words was always encoded as part of subword representation (e.g. with a joining marker @@@), and cases where two subwords were identical except for the presence or absence of a joining marker were not handled.

3. Baseline NMT Architecture

We used an open-source toolkit [12] based on TensorFlow [13] to implement a recurrent neural network (RNN) based translation model with attention similar to [3]. The model consists of 620-dimensional embedding layers for both the source and the target words, an encoder of 4 bidirectional LSTM layers with 1000 units each, a single-layer decoder LSTM with the same number of units and an additive attention mechanism. We augmented the attention computations using fertility feedback similar to [14].

In the baseline system, the output layer of the decoder consists of a single softmax that computes one target word \( y \) for each decoding step. Such a factorization is illustrated in Figure 2. The model makes predictions of its lower-cased form (\( y_1 \)) and cases where two subwords were identical except for the presence or absence of a joining marker were not handled.

4. Target-side Factors

Factored neural machine translation (FNMT) alters the decoder network such that in each decoding step two outputs \( y_1 \) and \( y_2 \) are generated instead of one. We choose an architecture similar to [1], see Figure 1.

The probability distribution for \( y_1 \) is computed directly from the readout vector \( t \) of the decoder:\footnote{following the notation of [1]}

\[
p(y_1|t) = \text{softmax}(W_{o,1}t), \tag{1}
\]

where \( W_{o,1} \) is the output weight matrix for the first output. We let \( p(y_2) \) depend on the embedding \( E_1y_1 \) of the first output, which is called lemma dependency in [1]:

\[
p(y_2|t, y_1) = \text{softmax}(W_{o,2}[t; E_1y_1]). \tag{2}
\]

Here, \([\cdot]\) denotes concatenation. The hidden state \( s \) of the decoder LSTM is the concatenation of the embeddings of both previous outputs \( y_1 \) and \( y_2 \):

\[
s = f(s', [E_1y_1', E_2y_2'], c), \tag{3}
\]

where \( s' \) is the previous state and \( c \) the context vector computed by the attention mechanism. \( E_1 \) and \( E_2 \) have the same embedding size as the baseline.

In training, we optimize the sum of the cross-entropies of both factors. For beam search, we first calculate the softmax of the first factor \( y_1 \), which results in an intermediate beam of the \( n \)-best hypotheses after adding scores for \( y_1 \). After that, for all hypotheses in the beam, the embedding of \( y_2 \) and the readout \( t \) from which \( y_1 \) originated are fed into the softmax for the second factor \( y_2 \). Here, scores for \( y_2 \) according to Equation 2 are added (in log-space). The final beam therefore contains hypotheses that were expanded by both \( y_1 \) and \( y_2 \) in the current decoding step.

Compared to the standard output layer, the factored output layer requires the computation of an additional softmax and logic to map the indices of the readout vectors \( t \) of the incoming beam into the intermediate beam. In practice however, this does not lead to an increase in decoding time, as explained in Section 7.2.

Note, that this factored output implementation can be generalized to more than two factors for more complex applications. Also, it is independent of the underlying network and can therefore be used on top of other architectures such as the Transformer [4].

5. Applications of the Factored Model

5.1. Word case prediction

As a first application, we use the factored output architecture to split the prediction of a target (sub)word into the prediction of its lower-cased form (\( y_1 \)) and a casing class (\( y_2 \)). Such a factorization is illustrated in Figure 2. The model makes no distinction between different casing variants of the same word w.r.t. the first factor \( y_1 \). This is helpful e.g. in the case of German verb nominalization, which entails capitalization. For example, there is a direct correspondence between the words treffen (meet) and Treffen (meeting). In a true-cased representation this connection would be lost, as these words would be represented as two distinct tokens. Another advantage of this factorization is that the vocabulary for the factor \( y_1 \) is lower-case only, and thus its size is reduced significantly.

We use four classes for the casing factor: lower-case, capitalized, all-caps, and undefined. Each original target word \( y \) of the target sentence is assigned to one of the classes. To fall into lower-case or all-caps, all alphabetical characters
in a word have to be lower- or upper-case, respectively. For capitalized, the first character has to be upper-case and the remaining ones lower-case. Tokens without any letters, as well as mixed-cased words other than capitalized, are assigned to undefined. In decoding, the predicted casing is applied to each corresponding word as a postprocessing step. For simplicity, in case of the undefined class we keep the word as is. This leads to errors for mixed cased subwords, which however make up less than 0.04% of the subwords in the test sets.

5.2. Subword segmentation

Another application of FNMT is the prediction of subword merges. Instead of encoding whether a subword is to be merged with adjacent subwords by using a special marker as part of the subword itself, we predict merges explicitly via the second output \( y_2 \) of the network, as shown in Figure 3. The motivation behind this is a better handling of compound words. Intuitively, the representation of a compound word should be a function of its compounds. However, with traditional subword methods, distinct tokens are used for the compound, depending on whether it is part of a word or stands alone. In the example in Figure 3 we have the German compound word Pilotversuch (pilot trial). In traditional subword vocabularies, the two tokens (Pilot, Versuch) would be separate subword tokens, and the fact that they are related would be only learned by the network if they are both frequent enough in the training data. In the proposed FNMT approach, the same token \( y_1 \) is used instead.

To create the two training targets \( y_1 \) and \( y_2 \), we first create a standard subword representation of the target text. From each subword \( y \) of the target sentence we then extract a binary flag \( y_2 \), telling whether the subword is to be separated from its left neighbour or not, based on the joining markers. The first output factor \( y_1 \) is created by removing the joining marker from \( y \) if present.

In decoding, we insert a space left of only those tokens for which \( y_2 \) is predicted as True.

6. Factored Operation Sequence NMT

Operation sequence neural machine translation, as proposed by [2], uses a special representation of the target sentence from which it is possible to extract an alignment link for each target word to one of the source words. This alignment information is potentially useful for a number of interesting applications, such as integrating external lexical rules into NMT or copying annotations, such as formatting, from source to target [13][16].

In OSNMT, the source words are translated monotonically. To encode reordering, the target vocabulary is extended by tokens to set markers in the sequence (SET_MARKER) and to jump between these markers (JMP_FWD and JMP_BWD). The alignment link is given by a virtual read head, which traverses the source sentence from left to right and is moved by an additional SRC_POP token. It is inserted into the sequence when all target words aligned to the source word under the current read head have been produced.

In a full operation sequence the SRC_POP token appears once for every source word. Therefore its length is at least the sum of the number of source and target words, which means high computational costs and, possibly, worse translation quality because of long range dependencies. We therefore propose to factor out the generation of SRC_POP tokens. For this, we predict all tokens except SRC_POP via the first output \( y_1 \) and let the second output \( y_2 \) decide how many SRC_POP tokens are to be inserted before \( y_1 \), i.e. how many source positions to step forward. Consequently, the target vocabulary for \( y_2 \) consists of integers ranging from 0 up to a maximum of \( V_{\text{SRC\_POP}} \).

For training, we first create the operation sequence using the original algorithm of [2] and then determine \( y_1 \) and \( y_2 \) as described above. In decoding, the SRC_POP tokens are inserted according to \( y_2 \) as a postprocessing step before compiling the operation sequence to get the plain target sentence and the alignment.
7. Experimental Results

We performed experiments on the WMT 2019 English→German and the WMT 2018 English→Turkish news translation tasks [17, 18]. For En→De, we used all parallel corpora provided by WMT. For En→Tr, we used the provided SETIMES2 corpus [19], as well as additional in-house data. We applied a number of heuristical data filtering rules, including: Only sentence pairs where both source and target side have more than 2 characters but less than 80 words are kept. Also, we require the number of source and target words to differ by a factor of 4 at most. On top of that, we used the FastText based language identification [20] to keep only those sentence pairs where both source and target language are assigned a confidence of at least 40%. Finally, we filtered out sentence pairs that have an 8-gram overlap with any sentence in the test data. For En→De, this resulted in a corpus of 23M lines and 456M running English words. For En→Tr we have 36M lines and 313M words.

In preprocessing, we converted the English source side to lower case and applied frequency-based truecasing to the target side. Both sides were encoded separately into subwords using the unigram model of the Sentencepiece toolkit [6]. A vocabulary size of 50K was used. For the case prediction experiments, we trained and applied the target subword model on lower-cased text (while still providing the original casing information to train the second factor).

For the OSNMT experiments, we word-aligned the training data with the Effomal toolkit [21]. Unlike [2], we did not convert the alignment to subword level. Instead we created the operation sequence on the word level[2] and only after that applied the subword encoding (without allowing splits of the special OSNMT tokens). This way, the number of SRC_POP tokens is reduced to the number of source words instead of subwords, while still allowing to infer the alignments to the original source words. Despite this, we still observe very long sequences of SRC_POP tokens in the training data corresponding to unaligned source segments. We therefore exclude all sentences with more than 10 subsequent SRC_POP operations from training (ca. 1%). Accordingly, we set $V_{SRC_{POP}} = 10$ (see Section 6).

To train the network, we used the Adam optimizer [22] with an initial learning rate of 0.001 and decayed the learning rate by a factor of 0.9 whenever validation set perplexity increases. For En→De we used news2015 as validation set, for En→Tr newstest2010. We used a layer-wise pre-training scheme [12], label smoothing of 0.1 and a softmax layer dropout of 0.3.

7.1. Automatic Evaluation

We report translation quality results on standard WMT test sets in Table 1 using case-sensitive BLEU [23] and TER [24].

| English→German | BLEU | TER | BLEU | TER |
|---------------|-----|-----|-------|-----|
| RNN baseline  | 27.6| 54.8| 37.4  | 46.7|
| + casing factor | 27.7| 54.8| 37.4  | 47.0|
| + segmentation factor | 27.8| 54.7| 37.4  | 46.9|
| OSNMT baseline | 23.2| 59.3| 27.3  | 55.8|
| + SRC_POP factor | 25.0| 57.4| 32.4  | 50.7|

Table 1: Automatic evaluation results (in %).

| English→German | BLEU | TER | BLEU | TER |
|---------------|-----|-----|-------|-----|
| RNN baseline  | 16.6| 66.1| 16.5  | 66.8|
| + casing factor | 16.3| 66.5| 16.2  | 67.3|
| + segmentation factor | 16.7| 66.2| 16.5  | 66.4|
| OSNMT baseline | 9.7 | 72.9| 9.9   | 73.4|
| + SRC_POP factor | 14.3| 69.2| 14.4  | 69.8|

Table 2: Simulated low-resource condition results (in %). All systems were trained on a corpus of 500K randomly sampled En→De sentences.

For En→De newstest2019, both a second factor predicting word casing as well as a second factor predicting subword merges resulted in identical BLEU scores to the baseline and only in a minor degradation in terms of TER. On En→De newstest2017, a tendency towards improvement can be observed for these factored systems.

The En→Tr results in Table 1 confirm the findings of the En→De experiments. The slightly worse performance of casing prediction may indicate that it is most effective for German, where capitalized words are much more frequent.

For both language pairs, the neural operation sequence model (OSNMT) cannot achieve comparable translation quality to standard NMT in our experiments. As discussed earlier, we believe this is in part caused by low target sequence lengths. Reducing them by predicting SRC_POP tokens via a factor improves the translation quality dramatically on all test sets, e.g. 5.1% BLEU absolute for En→De newstest2019. We hope to be able to further improve OSNMT in future work to close the gap to standard NMT.

Count-based factored translation models tend to be more effective under low-resource conditions [8]. Also, in the original paper [2] OSNMT is evaluated on smaller sized tasks of up to 1M sentence pairs only. For comparison, we simulate low-resource conditions on the En→De task by randomly sampling a subset of 500K sentence pairs from the
training corpus. To prevent overfitting, we used only 2 encoder layers and label smoothing of 0.2 for these experiments. In addition, we optimized the size of the subword vocabulary to 10K. As can be seen in Table 2, we observe mixed results in this setting. Casing prediction via a factor improves over the baseline by 0.8% BLEU on newstest2019. Factorized subword merging harms translation quality on newstest2017. For OSNMT, we again see a weak performance of the baseline, but a big improvement due to factoring out SRC_POP tokens.

7.2. Decoding Speed

In Table 3 we compare the decoding speed of the different systems. The decodings were run on a GeForce GTX 1080 Ti with a beam size of 12 and a batch size of 3000 tokens. The runtime was observed to be constant over three runs for each of the systems.

Introducing target factors causes no noticeable slowdown in our settings. This has two reasons: First, the softmax for the second factor has a very small number of output classes, therefore it is neglectable compared to first softmax in terms of computational costs. Second, factorization reduces the vocabulary size of the first factor. For example, using the segmentation factor reduced the German vocabulary from 50K to 45K subwords.

The OSNMT system is slower than the RNN baseline due to long sequence lengths. However, factorization here increases the decoding speed, as multiple SRC_POP tokens are predicted at once and simultaneously with the following token.

7.3. Analysis

We found casing prediction via a factor to work indistinguishably well as compared to using truecased subwords. Case-insensitive BLEU and TER scores for the baseline system and the one with the casing factor were found to be almost identical, e.g. 37.9% BLEU on En→De newstest2019 for both systems. Also, when calculating BLEU only on the casing classes of the translated words (see Section 5.1), the systems reach similar results (60.9% vs. 60.7% BLEU, respectively). Together with the similar case-sensitive scores (Table 1) this indicates that the systems neither differ significantly in the choice of words nor in their casing.

The lower-cased vocabulary of size 50K for the casing prediction experiments was able to represent a total of 134K distinct word casing variants from the training data. In low-resource settings, the 10K vocabulary represented 23K casing variants. We suspect that with enough training examples the baseline model is able to learn connections between different casing variants of a word, therefore a separate casing prediction does not improve translation quality. In low-resource settings, however, explicitly sharing information between casing variants can mitigate the data sparseness problem. This potentially explains the improvements seen in Table 2 comparing the casing factor to the baseline.

In Table 4 we show that the segmentation factor indeed improves translation of compound words in many cases, especially of those not seen in training. In the first two examples, the baseline is not capable of translating the unseen compound nouns show-jumping stadium and technology legend. Instead, it resorts to copying the English words to the target side. Using the segmentation factor, the model is able to produce the reference words (in the second case using a dash for concatenation, which is valid). In the last two examples, although the baseline is able to produce reasonable translations, only the factored model is able to produce the complex unseen German compounds found in the reference. The positive effect however is not significant to automatic metric scores (Section 7.1) as it affects only a small fraction of the words in the test sets.

8. Conclusion

We presented novel applications of factored NMT. We showed that word case information and joining of subword units can be predicted effectively by a target factor; this allows for a single representation of similar or even identical lexical items, so that more full forms can be kept for a given subword vocabulary limit. Experiments on two language pairs confirm that this can be done without loss in MT quality and without an increase in decoding time. We also showed that our factored OSNMT implementation significantly improves the original non-factored one by reducing the number of decoding steps. At the same time, we showed that OSNMT on two WMT tasks exhibits significantly lower MT quality as compared to state-of-the-art NMT. This result contradicts the original claims of [2], who, however, ran experiments for translation into English.

In the future, we plan to utilize target factors in NMT for other auxiliary prediction problems. In particular, we are interested in a better factored representation of word and phrase alignments, that is able to provide high quality translations and alignments at the same time.

9. References

[1] M. García-Martínez, L. Barrault, and F. Bougares, “Factored neural machine translation architectures,” in International Workshop on Spoken Language Translation (IWSLT’16), 2016.
| source | After the official part, the illustrious gathering wandered to the show-jumping stadium [...]
| reference | Nach dem offiziellen Teil wanderte die illustre Gesellschaft ins Springstadion [...]
| RNN baseline | Nach dem offiziellen Teil wanderte das illustre Treffen in das Show-Jumping-Stadion [...]
| + segm. factor | Nach dem offiziellen Teil wanderte die illustre Versammlung zum Springstadion [...]

| source | The technology legend’s latest project [...]
| reference | Das aktuelle Projekt der Technologielegende [...]
| RNN baseline | Das neueste Projekt der Technology Legend [...]
| + segm. factor | Das neueste Projekt der Technologie-Legende [...]

| source | But denuclearization negotiations have stalled.
| reference | Aber die Entnuklearisierungsverhandlungen sind blockiert.
| + segm. factor | Doch die Entnuklearisierungsverhandlungen sind ins Stocken geraten.

| source | The four Baden-Wuerttemberg-class vessels the Navy ordered back in 2007 [...]
| reference | Die vier im Jahr 2007 von der Marine bestellten Fregatten der Baden-Württemberg-Klasse [...]
| RNN baseline | Die vier baden-württembergischen Schiffe, die die Marine im Jahr 2007 bestellt hat [...]
| + segm. factor | Die vier Schiffe der Baden-Württemberg-Klasse, die die Marine im Jahr 2007 bestellt hat [...]

Table 4: Translation examples: baseline vs. segmentation factor

[2] F. Stahlberg, D. Saunders, and B. Byrne, “An operation sequence model for explainable neural machine translation,” in Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Brussels, Belgium: Association for Computational Linguistics, Nov. 2018, pp. 175–186.

[3] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in Proceedings of the International Conference on Learning Representations (ICLR), May 2015.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[5] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers, 2016.

[6] T. Kudo and J. Richardson, “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, 2018, pp. 66–71.

[7] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, “Moses: Open source toolkit for statistical machine translation,” in ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23–30, 2007, Prague, Czech Republic, 2007.

[8] P. Koehn and H. Hoang, “Factored translation models,” in Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL), 2007.

[9] O. Bojar, “English-to-czech factored machine translation,” in Proceedings of the second workshop on statistical machine translation, 2007, pp. 232–239.

[10] W. Shen, R. Zens, N. Bertoldi, and M. Federico, “The jhu workshop 2006 iwslt system,” in International Workshop on Spoken Language Translation (IWSLT) 2006, 2006.

[11] L. V. Lita, A. Ittycheriah, S. Roukos, and N. Kambhatla, “Truecasing,” in Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1. Association for Computational Linguistics, 2003, pp. 152–159.

[12] A. Zeyer, T. Alkhouli, and H. Ney, “RETURNN as a generic flexible neural toolkit with application to translation and speech recognition,” in Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations, 2018, pp. 128–133.

[13] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Man´e, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens,
B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org.

[14] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, “Coverage-based neural machine translation,” CoRR, vol. abs/1601.04811, 2016.

[15] P. Arthur, G. Neubig, and S. Nakamura, “Incorporating discrete translation lexicons into neural machine translation,” in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1557–1567.

[16] T. Zenkel, J. Wuebker, and J. DeNero, “Adding interpretable attention to neural translation models improves word alignment,” arXiv preprint arXiv:1901.11359, 2019.

[17] O. Bojar, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, and C. Monz, “Findings of the 2018 conference on machine translation (wmt18),” in Proceedings of the Third Conference on Machine Translation (WMT), Volume 2: Shared Task Papers. Association for Computational Linguistics, Oct. 2018, pp. 272–307.

[18] L. Barrault, O. Bojar, M. R. Costa-jussà, C. Federmann, M. Fishel, Y. Graham, B. Haddow, M. Huck, P. Koehn, S. Malmasi, et al., “Findings of the 2019 conference on machine translation (wmt19),” in Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), 2019, pp. 1–61.

[19] J. Tiedemann, “Parallel data, tools and interfaces in opus,” in Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC’12), N. C. C. Chair), K. Choukri, T. Declerck, M. U. Dogan, B. Maegaard, J. Mariani, J. Odijk, and S. Piperidis, Eds. Istanbul, Turkey: European Language Resources Association (ELRA), may 2012.

[20] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” arXiv preprint arXiv:1607.01759, 2016.

[21] R. Östling and J. Tiedemann, “Efficient word alignment with Markov Chain Monte Carlo,” Prague Bulletin of Mathematical Linguistics, vol. 106, pp. 125–146, Oct. 2016.

[22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, May 2015.

[23] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a Method for Automatic Evaluation of Machine Translation,” in Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318.

[24] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, “A Study of Translation Edit Rate with Targeted Human Annotation,” in Proceedings of the 7th Conference of the Association for Machine Translation in the Americas, Cambridge, Massachusetts, USA, 2006, pp. 223–231.