Curriculum Learning and Minibatch Bucketing in Neural Machine Translation

Tom Kocmi and **Ondřej Bojar**
Charles University,
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
surname@ufal.mff.cuni.cz

Abstract

We examine the effects of particular orderings of sentence pairs on the on-line training of neural machine translation (NMT). We focus on two types of such orderings: (1) ensuring that each minibatch contains sentences similar in some aspect and (2) gradual inclusion of some sentence types as the training progresses (so called "curriculum learning"). In our English-to-Czech experiments, the internal homogeneity of minibatches has no effect on the training but some of our "curricula" achieve a small improvement over the baseline.

1 Introduction

Machine translation (MT) has recently seen another major change of paradigms. MT started with rule based approaches which worked successfully for small domains. Generic MT was first reached with statistical methods, the early word-based and the late phrase-based dominant approaches, that build upon large training data. The current change is due to the first successful application of deep-learning methods (neural networks) to the task, giving rise to neural MT (NMT; Collobert et al., 2011; Sutskever et al., 2014). The data-driven methods have always been resource-heavy (e.g. word alignment needing a day or two for large parallel corpora) and NMT pushed this to new extremes: to reach the state-of-the-art performance, the model often needs a few weeks on the highly parallel graphics processing units (GPUs), equipped with large memory (8–12 GB) on a large training corpus.

The complexity of the training is a direct consequence of the complexity of the neural MT model: we need to find optimal setting of dozens millions of real-valued NMT model parameters that, according to the hard-coded model structure, define the calculation that converts the sequence of source words to the sequence of target words. The core of NMT training is thus numerical optimization, gradient descent, towards the least error as defined by the objective function. The common practice is to evaluate cross entropy against the reference translation.

The gradient of the objective function, in which the algorithm progresses, can be established on the whole dataset (called "batch training"), on individual examples ("online training") or a small set of examples ("minibatch training"). The full batch training has a clear advantage of reliable gradient estimates, while online training can easily suffer from instability. As documented by Wilson and Martinez (2003) on 27 learning tasks, online training reaches the same level of optima as the full batch training while having much lower memory demands and faster computation in general.

Minibatches typically contain 50 to 200 examples, calculate and average the error for all of them and propagate the error back through the network to update the weights. They have the advantages of both: the gradient is more stable and we decide how much of the training data it is convenient to handle at each training step. A further benefit comes from parallelizability on GPUs: the error of all the examples in the batch can be calculated simultaneously with the exact same formulas.

The training sets in NMT are simply too large, so full batch training is out of question and everybody uses minibatches.[1] The benefit of parallelization in minibatches can be somewhat diluted if minibatches contain sentences of varying length. In common frameworks for parallel computation, all the items in the minibatch must usually have

[1] In fact, the terms "batch" or batch size in NMT refer to minibatches; the whole corpus is then called an "epoch".

Proceedings of Recent Advances in Natural Language Processing, pages 379–386,
Varna, Bulgaria, Sep 4–6 2017.
https://doi.org/10.26615/978-954-452-049-6_050

the same length, and shorter sentences are therefore padded with dummy symbols. Calculations over the padded areas are wasted.

Khomenko et al. (2016) and Doetsch et al. (2017) report improvements in training speed by organizing (bucketing) training sentences so that sentences of identical or similar length arrive in the same minibatches. A related idea is called "curriculum learning" (Bengio et al., 2009) where the network is first trained with easier examples, making the task more complex only gradually.

In this work, we attempt to improve the final translation quality and/or reduce the training time of an NMT system by organizing minibatches in two particular ways. In Section 2, minibatches are created to contain sentences similar not only in length but in other (linguistic) phenomena, hoping for a better quality. In Section 3, similar criteria are used to organize the whole corpus, increasing the complexity of examples as training progresses, aiming at a better quality in shorter time. Section 4 evaluates our ideas in thems of translation quality and discusses the results. Related work is summarized in Section 5 and we conclude in Section 6.

2 Minibatch Bucketing

Minibatches stabilize the online training from fluctuations (Murata and Amari, 1999) and help to avoid a problem with overshooting local optima.

As mentioned, better performance of parallel processing has been achieved by bucketing training examples to contain sentences of similar length. The benefit of this approach however comes purely from the technical reason: avoiding wasted computation on paddings.

Each minibatch leads to one update of the model parameters and each example in the minibatch contributes to the average error. We assume that if all the examples in the minibatch are similar in some *linguistic sense*, they could jointly highlight the fitness of the current model in this particular aspect. Each minibatch would be thus focused on some particular language phenomenon and the gradient derived from this minibatch could improve the behavior of the model in this respect, allowing the network an easier identification of shared features of the examples.

We experiment with several features, by which we bucket the data. Those features are: sentence length, number of coordinating conjunctions, number of nouns, number of proper nouns and the number of verbs in the training data pairs. In our experiments we do not mix features together, but such mixed-focus minibatches are surely also possible.

The exact procedure of training corpus composition is the following: First, we divide all data based on their features into separate buckets (e.g. one bucket of sentences with at most one verb, another bucket of sentences with two or three verbs etc.). We then shuffle all examples in each bucket and break them down to groups of size same as the minibatch size. Finally, all these groups are shuffled and concatenated. The corpus is then read sequentially but our shuffling procedure ensured that all minibatches contain data having the same feature but among minibatches, the features are shuffled.

3 Curriculum Learning

When humans are trained, they start with easier tasks and gradually, as they gain experience and abstraction, they are able to learn to handle more and more complex situations. It has been shown by Bengio et al. (2009) that even neural networks can improve their performance when they are presented with the easier examples first.

For neural networks, it is important to keep on training also on the easy examples, because the networks are generally prone to very quick overfitting as we discuss in Section 4.5. If the network was presented only with the more difficult examples, its performance on the easy ones would drop. Some mixing strategy is thus needed.

Bengio et al. (2009) propose a relatively simple strategy. They organize all training data into bins of similar complexity. The training then starts with all the examples in the easiest bin (step-by-step in minibatches). With the easiest bin covered, the first and second easiest bins are allowed. In the final stage, examples from all the bins are used in the training.

The disadvantage of this approach is that examples in easier batches are processed several times. This boosts their importance for the training and also prevents us from directly comparing this strategy with the baseline of simply shuffled corpus.

We improve this strategy to use each example only once during an epoch. For our method to work, we require that the number of examples in the bin only decreases as we move to the bin of the higher complexity. This is usually easy to reach as there are generally more easier sentence pairs than complex sentence pairs in parallel corpora. The bin thresholds can be also adjusted to fulfill this condition.

The strategy for selecting examples from the bins is the following. First, we draw examples from the easiest bin only until there remain the same number of examples as in the second most easy bin. We then continue to draw uniformly from the first two easiest bins until in each of them, there remain the same number of examples as in the third one, etc. When taking the examples, we always accumulate one minibatch and feed it to the training. If the number of bins is smaller than the size of the minibatch, the minibatches in the late stages will contain examples from all complexity bins. If there are more bins than the minibatch size, each minibatch will be highly varied in complexity and the training will gradually proceed over examples of all complexities.

3.1 Selected Features

It is not entirely clear which examples are easy and which are hard for NMT (in various stages of the training). We experiment with several linguistically-motivated features.

The first feature is the length of the target sentence. (Source sentences usually have a corresponding length.) Our bins are for sentences of up to 8 tokens, up to 12 tokens, 16, 20, up to 40 tokens and for longer sentences. The thresholds were chosen to satisfy the requirement of more examples in easier bins.

The second binning is based on the number of coordinating conjunctions in the target sentence as one possible (rough) estimate of the number of clauses in the sentence. Conjuctions are also used in lists of items, so a higher number of them suggests that the sentence structure is cluttered with lists. Such examples may be easy to translate but do not correspond well to the generally hierarchical structure of sentences that we want to expose to the network. We use the same thresholds as for sentence length.

Learners of foreign languages often read books written with a simplified vocabulary. To replicate this learning strategy, we sort words by their decreasing frequency and define ranks on this list. For example, the first rank contains the 5000 most frequent words. Sentences are then organized into bins based on the least frequent word in them: the first bin contains sentences with all the words appearing the first rank.

We define the ranks separately for source and for target language and experiment with binning based on one of them or both at the same time.

4 Experiments

This section describes our experiments and results with minibatch bucketing and curriculum learning.

4.1 Model Details

We use Neural Monkey (Helcl and Libovický, 2017), an open-source neural machine translation and general sequence-to-sequence learning system built using the TensorFlow machine learning library.

Neural Monkey is quite flexible in model configuration but we restrict our experiments to the standard encoder-decoder architecture with attention as proposed by Bahdanau et al. (2015). We use the same model parameters as defined for the WMT 2017 NMT Training Task (Bojar et al., 2017). The task defines models of two sizes, one that fits a 4GB GPU and one that fits an 8GB GPU. We use the former one where the encoder uses embeddings of size 300 and the hidden state of 350. Dropout is turned off and maximum input sentence length is set to 50 tokens. The decoder uses attention mechanism and conditional GRU cells, with the hidden state of 350. Output embedding has the size of 300, dropout is turned off as well and the maximum output length is again 50 tokens. The Adam (Kingma and Ba, 2014) optimizer is used as the gradient descend algorithm.

To reduce vocabulary size, we use byte pair encoding (Sennrich et al., 2016) which breaks all words into subword units defined in the vocabulary. The vocabulary is initialized with all letters and larger units are added on the basis of corpus statistics. Frequent words make it to the vocabulary, less frequent words are (deterministically) broken into smaller units from the vocabulary.

As defined for the NMT Training Task, we set the vocabulary of size to 30,000 subword units. The vocabulary is constructed jointly for the source and target side of the corpus.

During the inference, we use simple greedy algorithm which generates the most frequent word depending on the previously generated words, the state of the decoder and attention. We did not employ any better decoding algorithm such as beam

Feature	Performance score
None (baseline)	14.25 ± 0.18 BLEU
Number of conjuctions	14.71 ± 0.24 BLEU
Number of proper nouns	14.58 ± 0.22 BLEU
Number of nouns	14.57 ± 0.24 BLEU
Sentence length	14.43 ± 0.23 BLEU
Number of verbs	14.43 ± 0.21 BLEU

Table 1: Minibatch bucketing after one epoch.

search (Sigtia et al., 2015; Graves, 2012) mainly due to technical difficulties. Although this decision leads to a poorer performance, it should not have any influence on the results of our work.

All experiments are based on one epoch of training over whole training dataset. The training takes roughly one week on NVIDIA GeForce GTX 1080. We should note that our model used only 4 GB of memory, instead of 8 GB available in the GPUs.

For the plots and presentation of the results, we compute test score (BLEU, Papineni et al., 2002) after every 100k training examples. To compensate for fluctuations during the training, we report the mean and standard deviation of the last 10 test errors of the training. This simple smoothing method is a substitute for proper significance testing (Clark et al., 2011), since we cannot run all experiments multiple times due to the lack of computing resources.

4.2 Training Data

We use the dataset provided for the WMT 2017 NMT Training Task. The dataset comes from the CzEng 1.6 corpus (Bojar et al., 2016) and it was cleaned by the organizers of the NMT Training Task. The resulting corpus is 48.6 million sentence pairs for English-to-Czech translation.

We use the test set from the WMT 2016 News Translation Task as our only heldout set. We do not need any separate development or validation set, because we are not doing any hyperparameter search or run experiments several times to find the best-performing setup.

4.3 Minibatch Bucketing

Table 1 shows the results of our experiments with minibatch bucketing. The bucketed runs are slightly better than the baseline but they usually fall in the standard deviation range so we cannot claim any significant improvement.

Feature	Performance score
None (baseline)	14.25 ± 0.18 BLEU
Source sentence length	15.41 ± 0.18 BLEU
Target sentence length	15.24 ± 0.27 BLEU
English word ranks	15.07 ± 0.28 BLEU
Czech word ranks	15.06 ± 0.29 BLEU
Number of conjuctions	15.04 ± 0.24 BLEU
Combined word ranks	14.77 ± 0.16 BLEU
Max word ranks	14.73 ± 0.22 BLEU

Table 2: Curriculum learning after one epoch.

4.4 Curriculum Learning

This sections describes our experiments with curriculum learning. We organized the training data based on the following features: the length of the sentences, the number of coordinating conjunctions, the highest rank of a word in the Czech or the English part and two combinations of the word ranks: “max word rank” which puts sentences into bins based on the maximum rank of their English and Czech words and “combined rank” is based on word ranks derived from concatenated source and target corpora.

As documented in Table 2, several of the curriculum setups improve over the baseline. The most beneficial is to organize the bins by the (source-side) sentence length, reaching a gain of 1.16 BLEU point.

Figure 1 plots learning curves for the baseline, one minibatch bucketing run (Section 2) and some curricula setups. Bucketing closely follows the baseline while curricula start much worse and make up later, as the complexity of training examples matches the fixed complexity of the test set.

The difference between source- and target-length curriculum is particularly interesting. Binning by target length ensures strict target-sentence limits and the decoder indeed follows the restriction never producing longer sentences regardless the source length. This results in serious penalization, see the sharp jumps in “Curriculum by target length”. Source-side binning makes target lengths slightly more varied. Assuming some model of sentence length in the decoder (Shi et al., 2016), training it on strictly capped sentences seems to damage its learning while the more varied data better allow to learn to predict output length based on the input length.

4.5 Quick Adaptation or Overfitting

Neural networks are known to quickly adapt to new types of data as they arrive in the training.

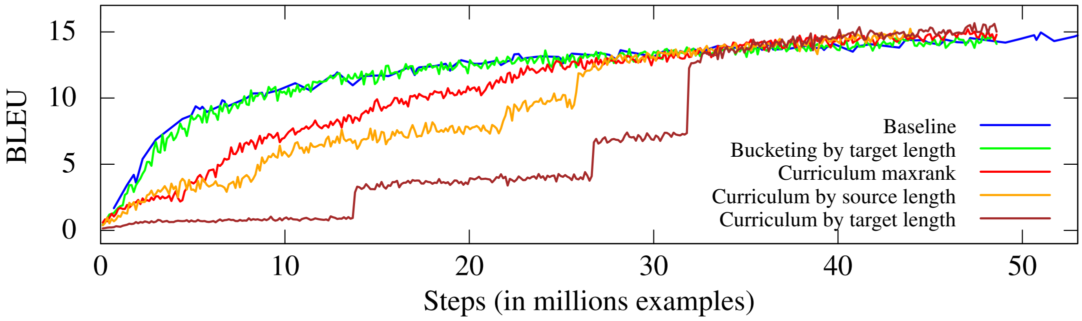

Figure 1: Selected learning curves for minibatch bucketing and curriculum.

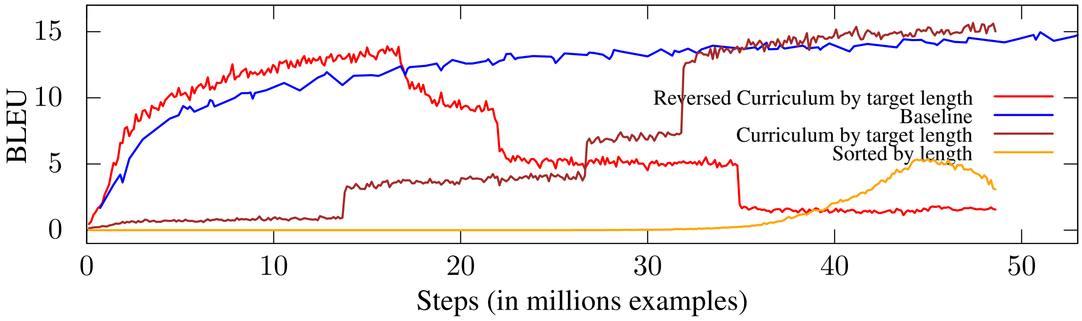

Figure 2: Learning curves of selected curriculum learning runs and other contrastive runs.

This effect is used e.g. in domain adaptation for NMT (Freitag and Al-Onaizan, 2016; Luong and Manning, 2015) but there is a big risk of overfitting to some specialized data.

As shown in Figure 2, our curriculum runs are heavily affected by this quick adaptation. "Baseline" shows the standard behaviour: starting quickly and then more or less flattening towards the end of the epoch.

Our best performing curriculum setup starts with short sentences and the model thus first learns to produce only short sentences. The curve "Curriculum by target length" shows very bad scores for more than a half of the training data, and the particularly striking are the quick transitions whenever a new bin of longer sentences is added. The model adapts and starts producing longer sentences, getting a huge boost in BLEU on the fixed test set. Towards the end of the epoch, "Curriculum by target length" demonstrates its improved generalization power and surpasses the baseline.

If we did not use our strategy of revisiting shorter sentences and simply sorted the corpus by sentence length, the training would fail spectacularly, see the curve "Sorted by length". The model never reaches any reasonable performance.

The curve "Reversed Curriculum by target length" is very interesting. We simply took the best corpus organization ("Curriculum by target length") and reversed it. The training performs better in the early stages (i.e. minibatches evenly covering all length bins) but very quickly drops as the long-sentence bins get prohibited. Put differently, the model quickly adapts (overfits) to the new "domain" of short sentences and fails to produces normal-length translations of the test set.

4.6 Continuing the Curriculum

It should be noted that all results presented so far are observed after one epoch of curriculum training. It is questionable what would be the best way of subsequent training.

We considered two options, see Figure 3. Starting over from the easiest examples harms the performance terribly early in the epoch but succeeds in improving the performance of the first epoch all the time, see the "Second epoch of curriculum by target length" in Figure 3.

Another option is to continue the training after the first epoch with the training dataset shuffled.

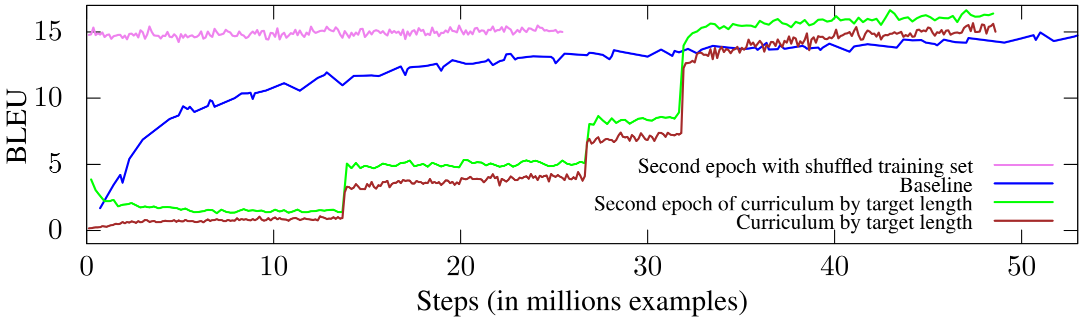

Figure 3: The best run ("Curriculum by target length") and its continuations.

As the corresponding curve in Figure 3 however shows, the model is probably already quite fixed in the current optimum and we do not see any further improvement on the test set.

5 Related Work

Khomenko et al. (2016) used a bucketing technique to accelerate the speed of the training. They prepared minibatches of training data with similar length and got a speedup in the training time of factor up to 4. The buckets are drawn randomly from the training set. A similar approach is also used in Nematus (Sennrich et al., 2017), one of the state-of-the-art open-source toolkits for NMT.

Doetsch et al. (2017) used bucketing and experimented with ordering of the bucketed batches. Their proposed method orders buckets in an alternating way: first in increasing order by length, then decreasing order, then again increasing order etc. This way the buckets of different length are periodically revisited. With this approach, the authors got a speedup in the training time and also obtained better performance results.

Bengio et al. (2009) use curriculum learning for a neural language model, not a full NMT system. They trained the network by iteratively increasing the vocabulary size, starting with the vocabulary of 5000 and increasing by 5000 each epoch. Each epoch used only sentences with words available in the current restricted vocabulary. The last epoch thus used all examples. This curriculum lead to a statistically significant improvement in the performance of the model.

Graves et al. (2017) automatically select examples during multitask learning. The method evaluates training signals from the neural network and uses them to focus on specific subtasks to accelerate the training process of the main task. The authors noted that uniformly sampling from the training data is a strong baseline.

6 Conclusion

We examined the effects of two ways of orderings of training examples for neural machine translation from English to Czech.

Trying to use sentences with similar linguistic properties in each minibatch of the online training (dubbed "minibatch bucketing") did not bring any difference from the baseline of randomly composed minibatches.

Organizing minibatches to gradually include more complex sentences (in terms of length or vocabulary size) helps to reach better translation quality of up to 1 BLEU point.

The actual process of learning is however very interesting, displaying clear jumps in the performance as longer sentences are added to the training data. The strategy cannot be thus used to shorten the training time: unless the gradually-organized epoch is finished, the model performs well below the baseline.

Our experiments also confirm the quick adaptability of deep learning methods, with a high risk of overfitting to particular properties of the very recent training examples.

Acknowledgement

This study was partly supported by the grants SVV 260 453, GAUK 8502/2016, H2020-ICT-2014-1-645442 (QT21) and Charles University Research Programme "Progres" Q18 – Social Sciences: From Multidisciplinarity to Interdisciplinarity. It has been using language resources and tools from LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*. http://arxiv.org/abs/1409.0473.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, pages 41–48.

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovický, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. Czeng 1.6: enlarged czech-english parallel corpus with processing tools dockered. In *International Conference on Text, Speech, and Dialogue*. Springer, pages 231–238.

Ondej Bojar, Jindich Helcl, Tom Kocmi, Jindich Libovick, and Tom Musil. 2017. Results of the wmt17 neural mt training task. In *Proceedings of the 2nd Conference on Machine Translation (WMT)*. Copenhagen, Denmark.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 176–181. http://www.aclweb.org/anthology/P11-2031.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Patrick Doetsch, Pavel Golik, and Hermann Ney. 2017. A comprehensive study of batch construction strategies for recurrent neural networks in mxnet. *arXiv preprint arXiv:1705.02414* .

Markus Freitag and Yaser Al-Onaizan. 2016. Fast domain adaptation for neural machine translation. *arXiv preprint arXiv:1612.06897* .

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *Proceedings of the 29th International Conference on Machine Learning (ICML 2012)* .

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. *arXiv preprint arXiv:1704.03003* .

Jindřich Helcl and Jindřich Libovický. 2017. Neural monkey: An open-source tool for sequence learning. *The Prague Bulletin of Mathematical Linguistics* (107):5–17. https://doi.org/10.1515/pralin-2017-0001.

Viacheslav Khomenko, Oleg Shyshkov, Olga Radyvonenko, and Kostiantyn Bokhan. 2016. Accelerating recurrent neural network training using sequence bucketing and multi-gpu data parallelization. In *Data Stream Mining & Processing (DSMP), IEEE First International Conference on*. IEEE, pages 100–103.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980. http://arxiv.org/abs/1412.6980.

Minh-Thang Luong and Christopher D Manning. 2015. Stanford neural machine translation systems for spoken language domains. In *Proceedings of the International Workshop on Spoken Language Translation*.

Noboru Murata and Shun-ichi Amari. 1999. Statistical analysis of learning dynamics. *Signal Processing* 74(1):3–28.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL 2002, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, pages 311–318.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Valencia, Spain, pages 65–68. http://aclweb.org/anthology/E17-3017.

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)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725. http://www.aclweb.org/anthology/P16-1162.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016. Why Neural Translations are the Right Length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2278–2282. https://aclweb.org/anthology/D16-1248.

Siddharth Sigtia, Nicolas Boulanger-Lewandowski, and Simon Dixon. 2015. Audio chord recognition with a hybrid recurrent neural network. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*. pages 127–133. http://ismir2015.uma.es/articles/227_Paper.pdf.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

D Randall Wilson and Tony R Martinez. 2003. The general inefficiency of batch training for gradient descent learning. *Neural Networks* 16(10):1429–1451.