Sequence Length is a Domain: Length-based Overfitting in Transformer Models

Dušan Variš and Ondřej Bojar
Faculty of Mathematics and Physics, Charles University,
Malostranské náměstí 25,
118 00 Prague, Czechia
{varis,bojar}@ufal.mff.cuni.cz

Abstract

Transformer-based sequence-to-sequence architectures, while achieving state-of-the-art results on a large number of NLP tasks, can still suffer from overfitting during training. In practice, this is usually countered either by applying regularization methods (e.g. dropout, L2-regularization) or by providing huge amounts of training data. Additionally, Transformer and other architectures are known to struggle when generating very long sequences. For example, in machine translation, the neural-based systems perform worse on very long sequences when compared to the preceding phrase-based translation approaches (Koehn and Knowles, 2017).

We present results which suggest that the issue might also be in the mismatch between the length distributions of the training and validation data combined with the aforementioned tendency of the neural networks to overfit to the training data. We demonstrate on a simple string editing task and a machine translation task that the Transformer model performance drops significantly when facing sequences of length diverging from the length distribution in the training data. Additionally, we show that the observed drop in performance is due to the hypothesis length corresponding to the lengths seen by the model during training rather than the length of the input sequence.

1 Introduction

Current state-of-the-art Transformer-based sequence generation models, either fine-tuned for chosen downstream tasks (Devlin et al., 2019), or trained from scratch for specific tasks such as machine translation (Vaswani et al., 2017) or speech recognition (Pham et al., 2019), more and more often achieve performance comparable to that of humans (Hassan et al., 2018; Popel et al., 2020; Nguyen et al., 2020). However, such models frequently require billions of trainable parameters together with huge amounts of data (billions of tokens) to reach such performance (Brown et al., 2020).

The good performance on held-out test sets seems to confirm the good generalization power of these models, although the inherent strong biases, sometimes leading to the use of a foul and toxic language, preserving stereotypes, etc., are well acknowledged (Gehman et al., 2020). Brown et al. (2020) claim that their Transformer model is also capable of simple arithmetics, however, it is yet to be validated whether the model truly learns the arithmetic algorithms or simply encodes a lookup table for a subset of specific examples.

In this paper, we argue that the assumed generalization power of the current state-of-the-art Transformer-based language generators does not come from the architecture itself but rather from the sheer volume of training data and the model's ability to exploit the similarities between the training and validation data. We demonstrate how the Transformer-based sequence-to-sequence models fail when the target sequence lengths of the training and validation data do not match. We show that this holds not only for very long test sequences but can be observed even with short sequences if they are omitted from the training data. Furthermore, we show that we can artificially improve the test performance on longer sequences by only using shorter training sequences and concatenating them into longer training examples.

We do not argue about Transformer's (in)ability to handle long-distance dependencies, but our results suggest that a considerably simpler reason of mismatching sequence length can also contribute to the performance drop. We think that our findings can lead to better understanding of the Transformer architecture and help to design better training schemes (e.g. curriculum learning).

2 Related Work

The problem of modeling very long sequences has been studied mainly in the context of recurrent neural networks (RNNs). Early studies showed that using LSTMs (Sutskever et al., 2014) and introducing attention (Bahdanau et al., 2014; Luong et al., 2015) can improve the model performance on long sequences. However, these models still got outperformed on long sequences by phrase-based models (Koehn and Knowles, 2017). This problem was not resolved with the introduction of Transformers (Vaswani et al., 2017). Surprisingly, even though there were previous studies explaining the weaknesses of RNNs with respect to long sequence modeling (Hochreiter and Schmidhuber, 1997; Hochreiter, 1998), similar analyses are yet to be done for Transformers which are fundamentally different from RNNs.

There is an ongoing debate about the proper way of splitting the available data to training and evaluation subsets. Gorman and Bedrick (2019) show that using only standard dataset splits can lead to a biased evaluation resulting in overestimating the generalization ability of the model. Furthermore, Søgaard et al. (2020) argue that even using randomly sampled dataset splits does not solve the overestimation problem. They instead suggest using multiple test sets possibly of an adversarial nature to properly evaluate the generalization ability of the model.

In the following experiments, we evaluate vanilla Transformer on such adversarial splits created with respect to the lengths of the modeled sequences. Although similar analyses were performed in the past (Neishi and Yoshinaga, 2019; Kondo et al., 2021), it was at a smaller scale and mainly in the context source-side length bucketing.

3 Experiments

We demonstrate the lack of ability to generalize to sequences of lengths not seen during training on two separate tasks: *string editing* and *machine translation* (MT).

We use Fairseq framework for sequence-to-sequence learning (Ott et al., 2019) in our experiments.[1] Details about the model parameters and training are available in Appendix A.

Input	Output
push 1 | 1 0 1 0	1 0 1 0 1
reverse − | 1 0 0 1 1	1 1 0 0 1

Table 1: Input and output example for `push` and `reverse` tasks. Hyphen (−) indicates an empty argument for the latter task.

	0-10	11-15	16-20
copy	62.6	100.0	0.0
push	59.1	100.0	0.0
pop	0.1	100.0	0.0
shift	52.5	100.0	0.0
unshift	41.2	100.0	0.0
reverse	0.0	84.4	0.0
all	15.822	97.5	0.978

Table 2: Accuracy (in %) of models trained on various string editing tasks using only training data from the 11-15 length bucket evaluated on datasets with different sequence lengths. Each model was evaluated on its respective task domain.

3.1 String Editing Operations

In the first set of experiments, we focus on learning simple string editing algorithms. We chose this task because we think it is an interesting alternative to standard NLP tasks that often struggle with evaluation ambiguity (multiple possible outputs in MT or text generation with nuanced degree of quality) and proper training/validation separation (partial overlap between train and test sentences leading to lack of clarity how much model actually generalizes to new inputs).

We chose to study the following tasks:

- *copy*: copy the input sequence to the output,
- *unshift X*, *push X*: add a single character (X) to the beginning or the end of the sequence respectively
- *shift*, *pop*: remove a single character from the beginning or the end respectively,
- *reverse*: reverse the character order in the input sequence

As for the experiment setup, we generate a dataset of sequences consisting of two characters (e.g. 0 and 1), separated by whitespace, with no duplicate sequences. Then, we split the dataset into three separate buckets according to sequence

[1] https://github.com/pytorch/fairseq

Bucket	0-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80
# of sent. pairs (M)	30.9	18.0	7.5	3.9	2.3	1.2	0.7	0.4
# of tokens (M)	375.3	502.6	361.6	268.9	198.9	132.6	87.3	59.5

Table 3: Sizes of the respective training buckets (created based on the target sequence length) in millions of sentence pairs and millions of tokens (after tokenization and applying BPE, combined source and target size).

lengths, $0-10$, $11-15$ and $16-20$ respectively. We sample 1,000 sequences from the $0-10$ and $16-20$ buckets for test-time evaluation. We split the $11-15$ bucket into a validation (1,000 examples), test (1,000 examples) and training (28,000 examples) from a sample of 30k examples without repetition.

Given these data splits, we create datasets for each task by adding the task label, character (0, 1 for unshift and push, $-$ for others) and a separator (|) to the beginning of each sequence.[2] We create target sequences for each task according to the respective task definition. Table 1 shows examples of the networks inputs.

For each task, we train a separate network on the 11-15 training data. Model details are available in Appendix A.1. We evaluate the models by measuring accuracy $ACC = num\_correct/num\_examples$, where $num\_correct$ is the number of exact matches between the hypothesis and reference strings. Table 2 shows the accuracy of the models trained on each task and evaluated on the varying test set buckets. We can see that the models generalize very well on the unseen sequences with length similar to the training sequences, all reaching the perfect accuracy except the *reverse* task. On the other hand, when facing shorter or longer sequences, the performance drops significantly.

Table 2 also shows results of the training a network on all tasks simultaneously (*all*; by concatenating and shuffling respective training data and performing evaluation on the concatenation of the respective testsets). The resulting performance is similar to that of a single-task model.

These results suggest that the length distribution similarity between the training and validation data is important and that the vanilla Transformer decoder is prone to overfitting to the sequence lengths seen during training.

[2]The arguments for unshift and push are sampled from a Bernoulli distribution with 0 character having $p = 0.5$.

3.2 Machine Translation

To see whether our findings within the string editing tasks also hold for natural language which has more complex structure, we perform similar experiments on English-Czech translation.

We use CzEng 2.0[3] (Kocmi et al., 2020) as our training corpus, a concatenation of WMT2020 (Barrault et al., 2020) `newstest13-16` as held-out test set and a concatenation of `newstest17-20` for final evaluation.[4] We tokenize our data using Moses tokenizer.[5] We use byte-pair encoding (Sennrich et al., 2016) on our training data, to create subword segmentation of size 30k.[6] We split all tokenized and BPE-segmented datasets into buckets of sizes 1-10, 11-20, ..., 91-100 (labeled as 10, 20, ..., 100 respectively) based on the number of tokens on the target side. Table 3 shows the sizes of the respective training corpora. We train a separate model for each training bucket. Details on the model hyper-parameters are available in Appendix A.2.

We evaluate how the length of the training data affects the performance with respect to the length of the test data using BLEU (Papineni et al., 2002), namely the SacreBLEU implementation (Post, 2018).[7] Figure 1 (Top) shows that regardless of the training bucket, the model performs best when presented with data of target-side length similar to the length of the training data. This confirms that the model overfits to the length of the training data, affecting its performance even on shorter sentences. The performance further decreases with increasing train-test length difference, although it needs to be noted that the BLEU scores between different testset buckets are not directly comparable due to the nature of the scoring metric and the fact that each testset bucket contains different test

[3]https://ufal.mff.cuni.cz/czeng

[4]We download the `newstest` corpora using SacreBLEU (Post, 2018).

[5]https://github.com/moses-smt/mosesdecoder.git

[6]https://github.com/rsennrich/subword-nmt.git

[7]https://github.com/mjpost/sacrebleu

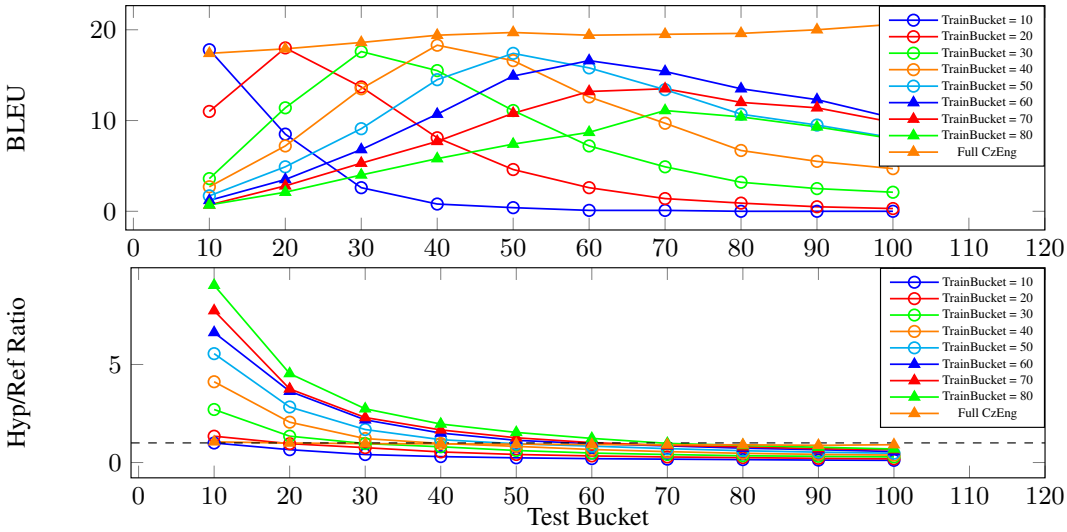

Figure 1: **Top**: Varying performance of Transformers on test data trained only on the data from a specific target-side length bucket (various lines) when evaluated on a specific test bucket (x-axis). When the train-test sentence length difference increases, the performance drops. Note that BLEU scores are not directly comparable across different test sets (i.e. horizontally). Within each test set, we see that the Full CzEng and the training bucket of the matching length are the two best results. **Bottom**: Average ratio between a hypothesis and reference. Dashed line indicates a ratio of 1.0. Systems trained on short sentences produce short outputs, systems trained on long sentences produce up to 10x longer outputs (Train Bucket 80 evaluated on Test Bucket 10).

examples. Figure 1 (Bottom) explains the main reason behind the BLEU decrease: the increased hypothesis/reference length ratio, further supporting the length overfitting argument. Note that the lower performance of the models trained on the 70 and 80 buckets migth be due to significantly smaller size of training data (< 1M sentence pairs). In Appendix B, we also provide a case study of the models trained on various length buckets.

The length-controlled experiment results presented by Neishi and Yoshinaga (2019), while not directly focused on exploring the target-side length overfitting phenomenon, point to a similar behavior of vanilla Transformers with regards to both longer and shorter test sentences. Based on their results, the replacement of the absolute positional embeddings with a variation of relative-position embeddings (Shaw et al., 2018; Neishi and Yoshinaga, 2019) seems like a promising approach towards alleviating the length overfitting problem.

To see whether we can exploit the target-side length overfitting behaviour, we also set up a separate experiment, similar to Kondo et al. (2021). We take the 10, 20 and 30 training buckets and concatenate the sentences in each of them to create synthetic datasets with target-side lengths 51-60 (containing on average 6, 3 and 2 sentences per training example, respectively). We apply the same training strategy using the synthetic data to see how strongly can the length of the training examples (although artificial) affect the model performance on the test examples of similar length.

Figure 2 shows that the simple concatenation of shorter training sentence pairs can lead to a performance similar to the model trained on the genuinely longer sentences. Only the performance of the model trained on the concatenation of very short sentences (the line "TrainBucket.Concat=10" in Figure 2) drops significantly, possibly because the model does not learn to handle any dependencies beyond the length of 10 and such dependencies seem to emerge in test sentences with length over 40, where the model starts to underperform.

Kondo et al. (2021) show that augmenting the existing training data with additional training examples that were created by concatenation of shorter sentences can help to improve model performance on very long sentences. Our results show that the synthetic concatenated data on their own can be sufficient to train a model that is competitive when applied to sentences from the similar target-length domain as the training examples. We also argue that due to a different bucket preparation strategy (based on the source-length in the previous work), the target-side length overfitting phenomenon is not as clear in Kondo et al. (2021) as in our work. In

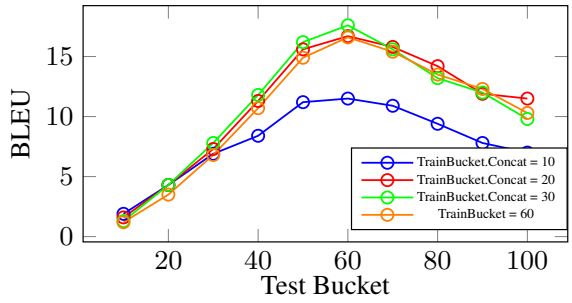

Figure 2: Comparison of the performance of a model trained on genuine data from the 60-length bucket with models trained on synthetic 60-length datasets created by concatenation of 10, 20 and 30-bucket sentences respectively.

Appendix C, we provide additional results from the experiments where the dataset bucketing is based on the source-side length instead of the target-side length for comparison.

4 Conclusion

We showed in our targeted experiment that vanilla Transformer sequence-to-sequence models have a strong tendency to overfit with regard to the target-side length of the training sequences. On a simple algorithmic task, we documented that Transformer can generalize very well to unseen examples *within the same length bucket* but falls short if the same task is required for input of a different length, shorter or longer. The algorithm of the task, even if very simple, is not learned.

We further confirmed the overfitting problem on the machine translation task. This suggests that long-distance dependencies are not the only reason behind the decreased performance when translating very long sentences. We think that our findings can shed a new light on specific areas of deep learning research, namely domain adaptation and curriculum learning.

We also showed that data augmentation can tackle the data sparsity in the domain of very long sentences.

Acknowledgements

This work was supported by the GA ČR NEUREM3 grant (Neural Representations in Multimodal and Multi-lingual Modelling, 19-26934X (RIV: GX19-26934X)) and by SVV 260 453 grant.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Loïc Barrault, Ondřej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Yvette Graham, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Makoto Morishita, Christof Monz, Masaaki Nagata, Toshiaki Nakazawa, and Matteo Negri, editors. 2020. *Proceedings of the Fifth Conference on Machine Translation*. Association for Computational Linguistics, Online.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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*, pages 176–181, Portland, Oregon, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *EMNLP (Findings)*, pages 3356–3369. Association for Computational Linguistics.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791, Florence, Italy. Association for Computational Linguistics.

Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, Shujie Liu, Tie-Yan Liu, Renqian Luo, Arul Menezes, Tao Qin, Frank Seide, Xu Tan, Fei Tian, Lijun Wu, Shuangzhi Wu, Yingce Xia, Dongdong Zhang, Zhirui Zhang, and Ming Zhou. 2018. Achieving human parity on automatic chinese to english news translation. *CoRR*, abs/1803.05567.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114 13:3521–3526.

Tom Kocmi, Martin Popel, and Ondrej Bojar. 2020. Announcing czeng 2.0 parallel corpus with over 2 gigawords. *arXiv preprint arXiv:2007.03006*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39, Vancouver. Association for Computational Linguistics.

Seiichiro Kondo, Kengo Hotate, Tosho Hirasawa, Masahiro Kaneko, and Mamoru Komachi. 2021. Sentence concatenation approach to data augmentation for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 143–149, Online. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Masato Neishi and Naoki Yoshinaga. 2019. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 328–338, Hong Kong, China. Association for Computational Linguistics.

Thai-Son Nguyen, Sebastian Stüker, and Alex Waibel. 2020. Super-human performance in online low-latency recognition of conversational speech. *CoRR*, abs/2010.03449.

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 NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, and Alex Waibel. 2019. Very deep self-attention networks for end-to-end speech recognition. In *INTERSPEECH*, pages 66–70. ISCA.

Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtský. 2020. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nature Communications*, 11(4381):1–15.

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, Belgium, Brussels. 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.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ebert, J. Bastings, and Katja Filippova. 2020. We need to talk about random splits. *CoRR*, abs/2005.00636.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz

Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6000–6010. Curran Associates, Inc.

A Model Details

In the following section, we provide the details of the used models and their training. All the described models are implemented in Fairseq (Ott et al., 2019).[8] During training, we use word-level cross-entropy loss with teacher forcing (Bahdanau et al., 2014; Vaswani et al., 2017) which is a current, widely used approach to the sequence-to-sequence Transformer training. During decoding, we use beam search with beam size 4 and length penalty 0.6.

A.1 String Editing

In the experiments with string editing, we use the `transformer` parameter setting with the following modifications:

- embeddings size: 128,
- feedforward size: 512,
- number of attention heads: 8,
- encoder/decoder depth: 1,
- batch size: 4,096 tokens,
- learning rate 5e-4,
- warmup steps: 4,000,
- dropout: 0.3,
- train epochs: 100

A.2 Machine Translation

In the machine translation experiments, we use the `transformer` parameter setting with the following modifications:

- embeddings size: 512,
- feedforward size: 2048,
- number of attention heads: 8,
- encoder/decoder depth: 6,
- batch size: 4,096,
- learning rate: 5e-4,
- warmup steps: 4,000,
- dropout: 0.3

[8] https://github.com/pytorch/fairseq

During training, we apply early stopping: if the model performance in BLEU (Papineni et al., 2002) does not improve for 10 epochs (evaluated on the complete held-out test set without length splits), the training is terminated.

B Translation Output Examples

Figure 3 shows example outputs from models trained on various target-length training buckets (10-, 30- and 60-bucket) produced by translating a chosen 30-bucket testset inputs. The examples demonstrate that he models have tendencies to produce outputs with length similar to the training data while trying to satisfy the translation of the source sentence resulting in the longer, 60-bucket model repeating certain phrases or sentences while introducing grammatical errors (e.g. wrong agreement, preposition choice) or mistranslations. On the other hand, the shorter, 10-bucket model manages to drop parts of the input sentence while maintaining a reasonable fluency and grammatical correctness of the output.

Figure 4 shows example of outputs from models trained on the synthetic 60-bucket data created by concatenation of the shorter training buckets. At first glance, all three hypotheses are very similar and are reasonably good translations of the source sentence, however, all systems made a wrong surface form and preposition choice for "na Vinohradech" (the same grammatical mistake as with "na Žižkově" in Figure 3), producing an incorrect but literal translation of the English "in Vinohrady". Additionally, all three systems chose a literal translation of the word "approach", which is incorrect in the given context. The incorrect surface form of the translation "založeno na doporučení" in *Hyp1* suggests that training a model on a concatenation of very short sentences may lead to incorrect modeling of long-range dependencies. Surprisingly, the *Hyp3* system mistranslated the phrase "work on the reconstruction" ("k rekonstrukci" in the output) while *Hyp2* system produced a correct translation, though this error is most likely a result of different set of training sentences in the *Hyp2* and *Hyp3* training data rather than the lenght of the training sentences (before concatenation).

C Source-Side Bucketing Experiments

For comparison, we repeated the translation experiments using source-side length-based bucketing of the training and validation data. Figure 6 shows

Source (30-bucket)	The company does not collect its mail and it has closed its official headquarters in Žižkov more than six years ago.
Hyp1 (10-bucket) *Hyp1 (gloss)*	Společnost nesbírá poštu a zavřel oficiální sídlo. *The company does not gather mail and closed official headquarters.*
Hyp2 (30-bucket) *Hyp2 (gloss)*	Společnost neshromažďuje poštu a již před více než šesti lety zavřela své oficiální sídlo v Žižkově. *The company does not collect mail and more than six years ago closed its official headquarters in Žižkov.*
Hyp3 (60-bucket) *Hyp3 (gloss)*	Společnost nevybírá poštu a uzavřela své oficiální sídlo v Žižkově více než šest let ago. v Žižkově. Společnost neshromažďuje poštu a uzavírá oficiální ústředí v Žižkově více než šest let agr. o. *The company does not pick up mail and closed up its official its official headquarters in Žižkov more than six years ago. in Žižkov. The company does not collect mail and closes up official headquarters in Žižkov more than six years agr. o.*
Reference (30-bucket) *Ref (gloss)*	Nepřebírá poštu a oficiální sídlo na Žižkově zrušila před více než šesti lety. *(The company) does not collect mail and official headquarters in Žižkov closed up more than six years ago.*
Source (30-bucket)	The perpetrators ended up in custody, said Marie Štrbáková, the spokeswoman of Olomouc police.
Hyp1 (10-bucket) *Hyp1 (gloss)*	Mluvila s ní Marie Štrkováková *Talked to her, Marie Štrkováková*
Hyp2 (30-bucket) *Hyp2 (gloss)*	Pachatelé skončili ve vazbě, řekla Marie Štrbákováová, mluvčí Olomouckého policie. *The perpetrators ended up in custody, said Marie Štrbákováová, the spokeswoman of Olomouc police.*
Hyp3 (60-bucket) *Hyp3 (gloss)*	Uchazeči skoncovali v úschově, "uvedla Marie Štrbákováová, mluvčí Olomoucké policie, která se stala mluvčí Olomouckého vojska, a to v úschově. *The candidates ended up in storage, "introduced Marie Štrbákováová, the spokeswoman of Olomouc police, which became the spokeswoman of Olomouc army, and in storage.*
Ref (30-bucket) *Ref (gloss)*	Pachatelé skončili ve vazbě, informovala olomoucká policejní mluvčí Marie Štrbáková. *The perpetrators ended up in custody, informed Olomouc police spokeswoman Marie Štrbáková.*

Figure 3: Example translations from systems trained on specific target-length-restricted datasets. Both examples demonstrate over and under-generation of systems trained on datasets containing longer (60-bucket) and shorter (10-bucket) sentences when applied to inputs with length of reference translation different from the training data (30-bucket). We provide rough, "word-for-word" translations of the produced outputs (in *italics*) with color highlighting of some of the phrases and their corresponding English translation for better comprehension. The underline highlights grammatical errors or mistranslations in the output.

the performance of the bucketed models with respect to testset of various bucket sizes. While the results are similar to the target-side bucketing experiments, the overfitting phenomenon is less clear in several cases (e.g. 20-bucket system reaching higher BLEU than 10-bucket system on 10-bucket testset or the relative system ranking on the 60-bucket testset).

We think that the possible reason is the difference between the source-side length and the length of training/validation reference leading to possible overlap of target-side lengths between the different train/validation buckets. Figure 5 shows the length distributions of target-side lengths within each training and validation bucket. Although the length-wise overlap between the target-side of the training/validation examples is manifested mostly in the 1^{st} and 4^{th} quartile, we think that it helps to support the argument that the length-based overfitting should be studied with respect to the target-side length instead of the source-side. Furthermore, the length of the target-side (Czech) in the test dataset is generally smaller than the source-side (English), resulting in additional domain mismatch between the training-test buckets. Note that very long target-side outliers in the training data are most likely a result of an imperfect sentence-pair filtering after the inclusion of the additional synthetic parallel data (forward and backward translation) to the CzEng 2.0 corpus.

Based on the reviewer's suggestion, we also measured the effect of finetuning a system trained on the whole training dataset using a source-side bucketed training data. Each system was fine-tuned for

Source (60-bucket)	We have already worked with Lenka Langerová on our flat in the mountains based on a recommendation from another client and because everything worked well we decided to approach her to work on the reconstruction of our new flat in Vinohrady.
Hyp1 (10-bucket-concat)	Už jsme pracovali s Lenkou Langerovou na našem bytě v horách založeno na doporučení od jiného klienta a protože všechno fungovalo dobře, rozhodli jsme se k ní přiblížit k práci na rekonstrukci našeho nového bytu ve Vinohrady.
Hyp2 (30-bucket-concat)	Již jsme spolupracovali s Lenkou Langerovou na našem bytě v horách na základě doporučení jiného klienta a protože vše fungovalo dobře, rozhodli jsme se, že se k ní přiblížíme, aby pracovala na rekonstrukci našeho nového bytu ve Vinohrady.
Hyp3 (60-bucket)	Již jsme s Lenkou Langerovou spolupracovali na našem bytu v horách na základě doporučení jiného klienta a protože vše fungovalo dobře, rozhodli jsme se, že se k ní přiblížíme k rekonstrukci našeho nového bytu ve Vinohrady.
Ref (60-bucket)	S architektkou Lenkou Langerovou jsme spolupracovali už na našem horském apartmánu, tehdy na bázi osobního doporučení jiného klienta, a vzhledem k tomu, že vše dobře fungovalo, byla pro nás jasná volba i při rekonstrukci našeho nového bytu na Vinohradech.

Figure 4: Example of translation hypotheses generated by a system trained on a genuine 60-bucket data and systems trained only on a concatenation of shorter training examples (10-bucket-concat, 30-bucket-concat) for comparison. Color highlighting indicates the correspondence of Czech and English phrases. The underline highlights grammatical errors in the output.

30 epochs, although, it is important to note that the validation BLEU of each fine-tuned system was dropping during training (compared to the BLEU of the initial model) when evaluated against the whole non-bucketed validation dataset. In Figure 7, we can see a growing effect of catastrophic forgetting (Kirkpatrick et al., 2017): all models initially saw all lengths during pretraining but specialized for a specific length bucket during finetuning. Interestingly, the forgetting effect is stronger for test buckets that are longer than the finetuning lengths while the models show much better retention of the ability to model shorter sentences.

Lastly, we also performed a comparison between the baseline MT system and the combination of systems trained on a specific source-side length buckets training datasets. We extracted sentences from our test dataset with source-side length 0–80, translated them with the respective systems and computed the BLEU scores using MultEval (Clark et al., 2011).[9] We compared a system combination trained using only a specific length-bucket dataset (`bucketed`) applied on the respective "in-domain" parts of the test dataset. We also provide comparison with the system combination initialized by the baseline model and then fine-tuned on the respective length-bucket datasets (`bucketed.tuning`). Additionally, we also trained a system using CzEng 2.0 with additional source-side labels indicating a length-bucket in which a given training example ended up after the source-side length-based dataset splitting (`bucket.labels`), e.g "<20> Example sentence..." for a sentence from a bucket 11–20. This model was evaluated on the same test dataset with inclusion of these source-side length-bucket labels.

[9] https://github.com/jhclark/multeval

	BLEU
baseline	19.1 ± 0.2
bucketed	18.9 ± 0.2
bucketed.tuning	17.1 ± 0.2
bucket.labels	16.7 ± 0.2

Table 4: Comparison of the translation performance of the `baseline` model trained on the whole CzEng 2.0, and source-length specialized models. `bucketed` is a combination of systems trained on the source-length bucketed training data, `bucketed.tuning` is a similar combination, where systems were first initialized by the `baseline` model and then fine-tuned for 30 epochs on their respective buckets. `bucket.labels` is a system trained on the whole CzEng 2.0 with inclusion of the source-side bucket length labels on the input. The systems were evaluated using MultEval (Clark et al., 2011) using a bootstrapping over a test dataset containing sentences of source-side lengths 0–80. Only a single optimizer run was performed for each evaluated system.

The results in Table 4 suggest that the length-based specialization of the models does not outperform the baseline. One of the possible explanations is a fact that `baseline` system was trained on the

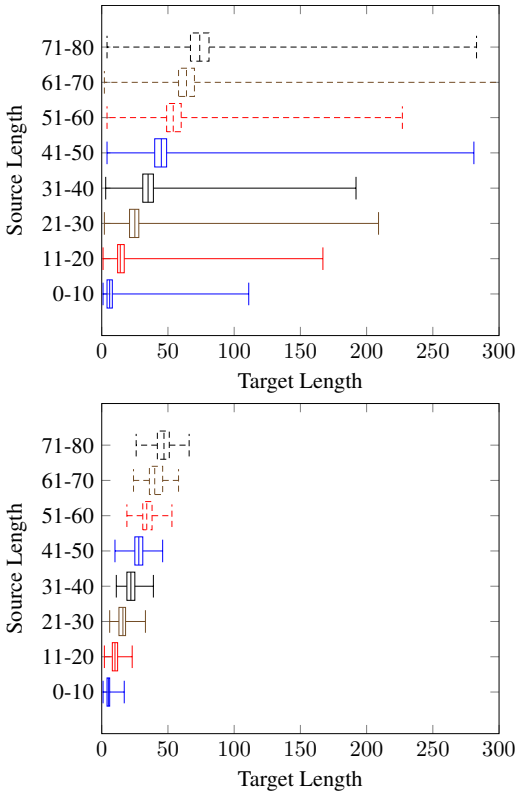

Figure 5: Distribution of lengths of target-side references within the training (**top**) and validation (**bottom**) datasets after splitting them into source-side length buckets. Both figures have identical x-axis scaling for better comparison. The long whiskers of the training bucket length distributions are a result of a noisy nature of CzEng 2.0 training corpus.

whole CzEng 2.0 containing even sentences longer than 80. Although the `bucket.labels` was also trained using the whole CzEng 2.0, the results suggest that a simple inclusion of the source-length bucket information does not contribute towards a better translation performance.

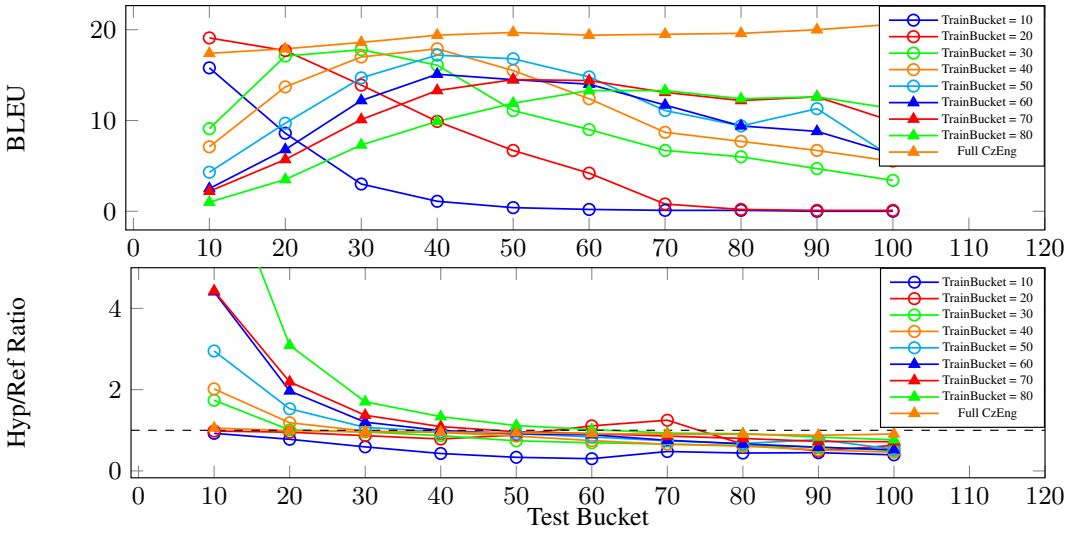

Figure 6: **Top**: Varying performance of Transformers on test data trained only on the data from a specific source-side length bucket (various lines) when evaluated on a specific test bucket (x-axis). BLEU scores are not directly comparable across different test sets (i.e. horizontally). **Bottom**: Average ratio between a hypothesis and reference. Dashed line indicates a ratio of 1.0.

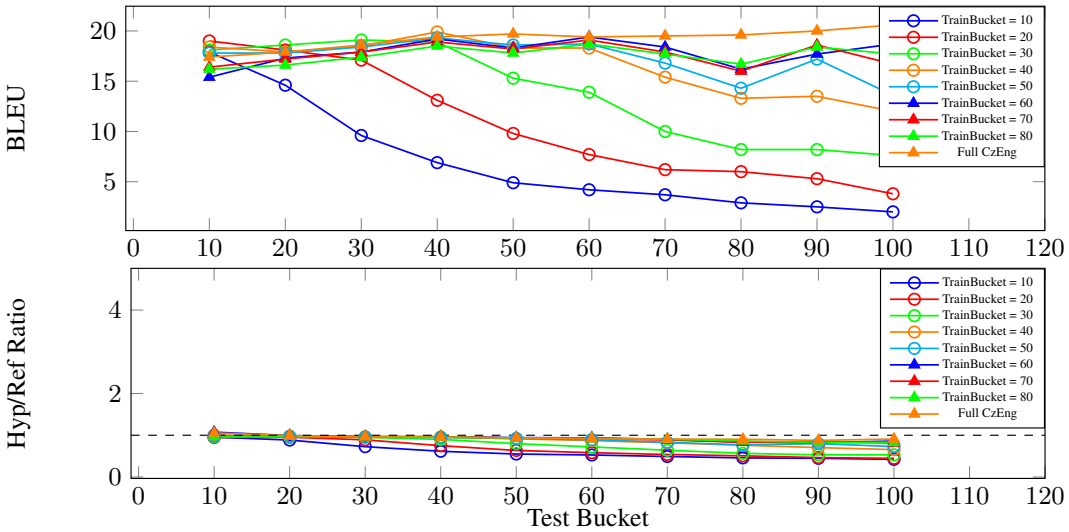

Figure 7: **Top**: Varying performance of Transformers on test data trained on all of CzEng and fine-tuned only on the data from a specific source-side length bucket (various lines) when evaluated on a specific test bucket (x-axis). BLEU scores are not directly comparable across different test sets (i.e. horizontally). **Bottom**: Average ratio between a hypothesis and reference. Dashed line indicates a ratio of 1.0. We preserve the scaling of all the plots for better comparability across the figures.