Post-correction of OCR results using pre-trained language model

Mateusz Piotrowski

Problem

The OCR post-correction task is very similar to common pre-training objectives for neural language models eg. Masked Language Modeling (Devlin et al. 2018). In both cases the goal is to *denoise* the input text - restore original tokens from corrupted version.

Model

The proposed solution uses mT5 transformer model (Xue et al. 2021):

- Multilangual model pretrained on massive corpus including 130 billion tokens in Polish subcorpus
- Encoder-decoder architecture can perform insertion, deletion and substitution operations
- Pre-trained models available in configuration ranging from 300 million to 13 billion parameters

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

1. Calculate maximum character offset according to model vocabulary

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

1. Calculate maximum character offset according to model vocabulary
2. Find matching regions using Python *difflib* library

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

1. Calculate maximum character offset according to model vocabulary
2. Find matching regions using Python *difflib* library
3. Select first sufficiently long matching region left of the maximum offset

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

1. Calculate maximum character offset according to model vocabulary
2. Find matching regions using Python *difflib* library
3. Select first sufficiently long matching region left of the maximum offset
4. Split input and output text on common point within this region and create training example

Example preparation

Transformer architecture imposes a limit on the number of input and output tokens. The following procedure was applied to correctly split longer examples:

1. Calculate maximum character offset according to model vocabulary
2. Find matching regions using Python *difflib* library
3. Select first sufficiently long matching region left of the maximum offset
4. Split input and output text on common point within this region and create training example
5. Repeat from pt. 1 for the remaining chunk of text

Example preparation

In some cases the OCR may fail to detect parts of text. Including such examples may cause the model to exhibit generative behaviour during inference:

Target
Ażeby się nie odrywać od tych zajęć odpisał na wezwanie z Rzymu następujące słowa; „Choć pragniecie mego powrotu i uważacie, iż winienem wracać jak najprędzej, przypuszczam, że pragnąć musicie

Predicted
Ażeby się nie odrywać od tych zajęć odpisał na wezwanie z Rzymu następujące słowa: „Choć pragniecie mego powrotu i uważacie, *iż mnie oddalić nie warto, oddam cały swój taniec i życie jego w całym świecie. Nie potrzebuję się wracać do domu, gdyż wiem, że w całym świecie*

Example preparation

To prevent such behaviour only examples matching similarity criteria were included:

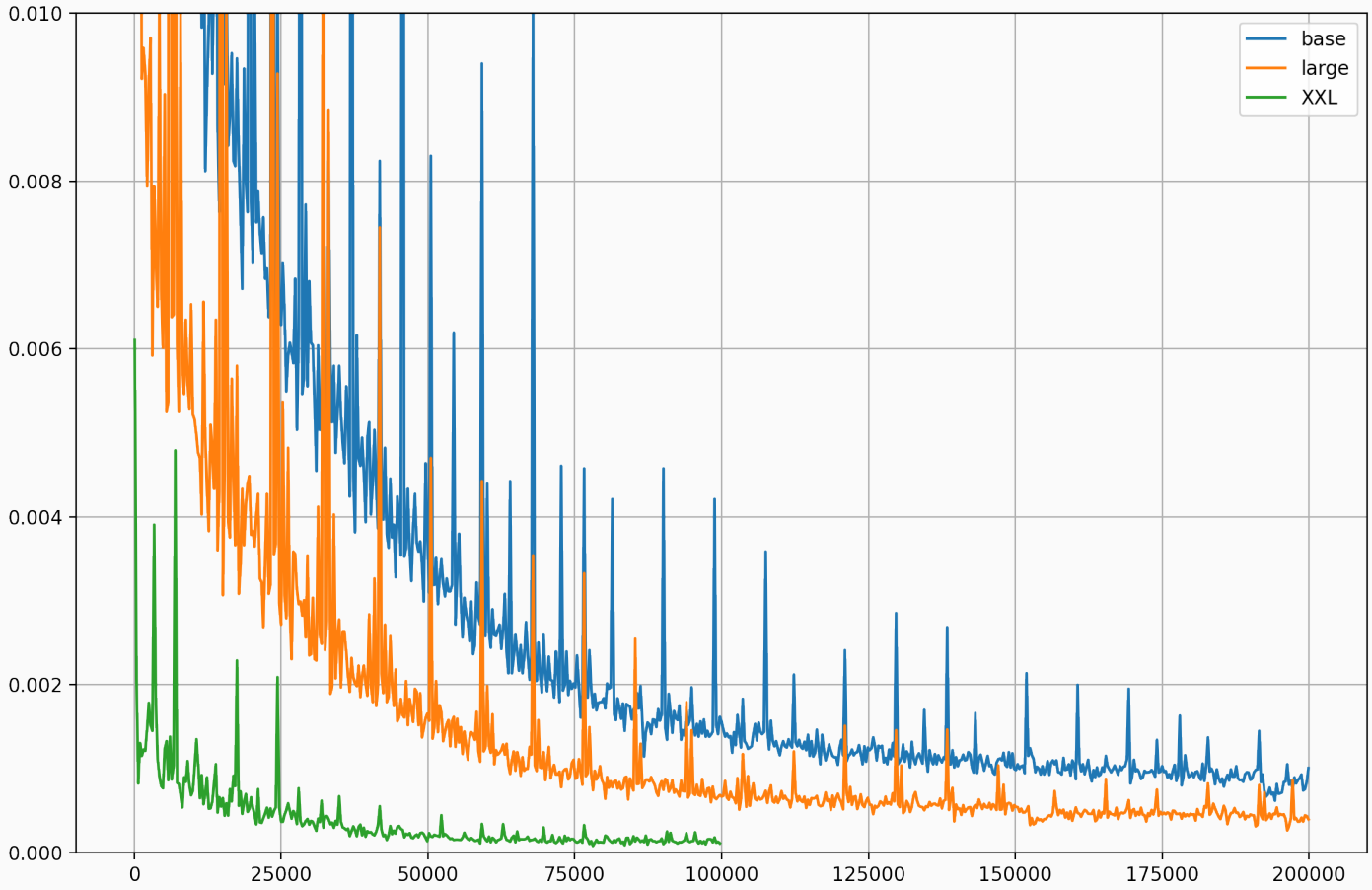
- Similarity ratio as reported by *difflib* is above 0.4
- Difference in length is less than 20%

Training

Different configurations were fine-tuned to assess the impact of model size. TPUv3 was used to train the models. Each configuration used 384 as a maximum token length.

model	parameters	batch size	examples/sec	training time
base	580M	128	150	2d
large	1.2B	64	40	3d 16h
XXL	13B	16	7	2d 16h

Training

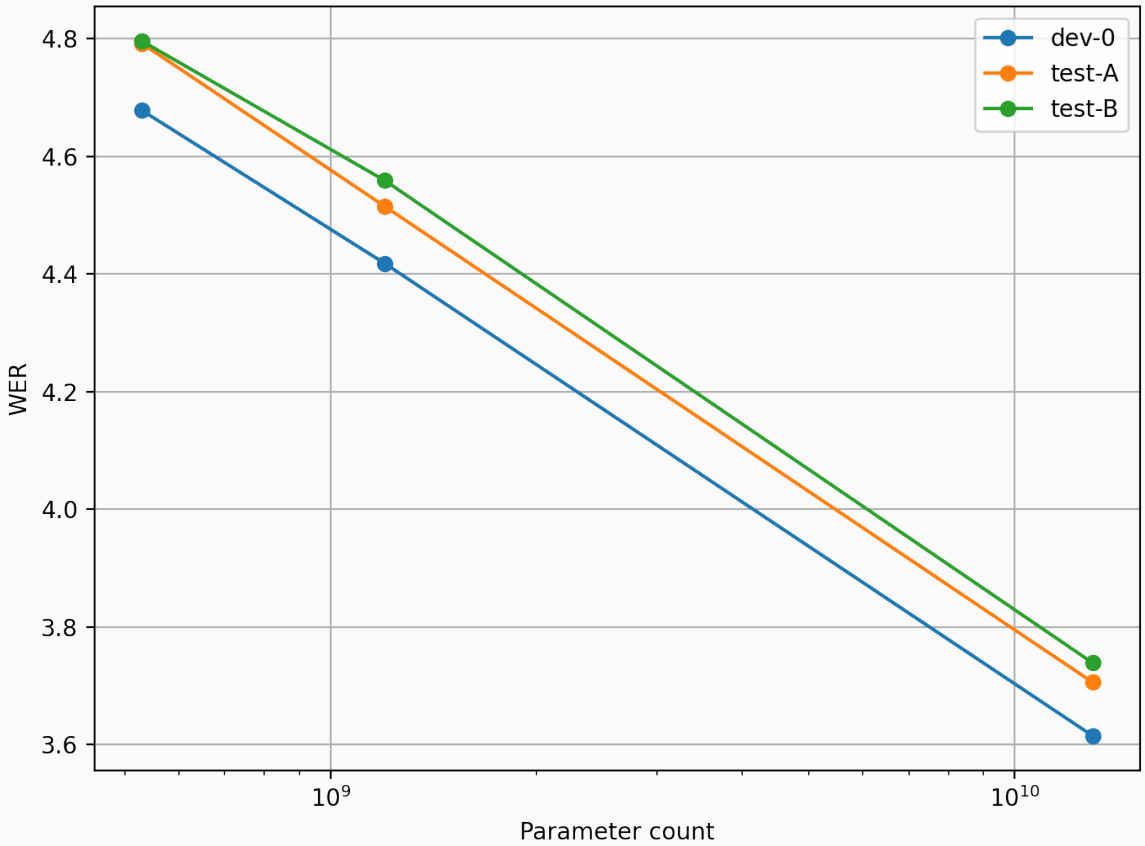

0.010
0.008
0.006
0.004
0.002
0.000
0
25000
50000
75000
100000
125000
150000
175000
200000
base
large
XXL

Results

model	dev-0	test-A	test-B
original	16.550	16.527	16.543
base	4.678	4.792	4.796
large	4.418	4.515	4.559
XXL	3.604	3.725	3.744

Results

Dependence of achieved performance on the model size in terms of a number of parameters seems to follow power-law relation (Kaplan et al. 2020)



Conclusions

- Scaling the model size is a straightforward way to obtain better results in a compute efficient matter
- Use of larger models is currently limited by the requirement of using expensive specialized hardware to meet memory requirements
- Applying neural network pruning techniques may allow for efficient deployment of larger models
- Including synthetic training data could improve the performance, but may introduce domain mismatch if not done properly

Acknowledgments

Research supported with Cloud TPUs from Google's TPU Research Cloud (TRC)

Thank you for your attention!

References

[Dev+18] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[Kap+20] Jared Kaplan et al. *Scaling Laws for Neural Language Models*. 2020. arXiv: 2001.08361 [cs.LG].

[Xue+21] Linting Xue et al. “mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 483–498. DOI: 10.18653/v1/2021.naacl-main.41. URL: https://aclanthology.org/2021.naacl-main.41.