Training your Tokenizer Right

Testing the Impact of Tokenization on Language Modelling with (Small) Transfomers

Rafael Braga Medeiros Mota Borges

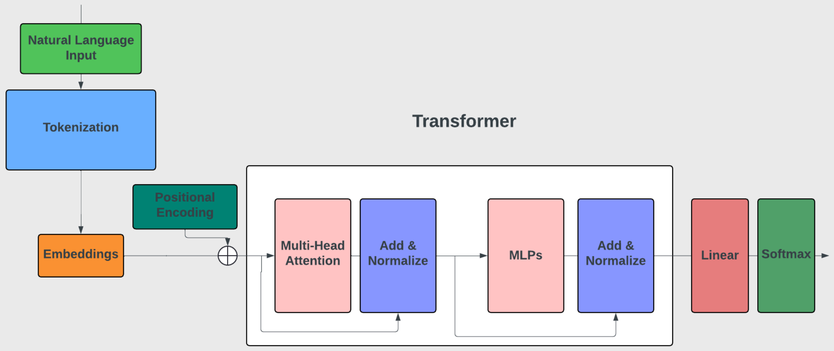
R.BragaMedeirosMotaBorges@student.tudelft.nl

Prof. Arie van Deursen, Asst. Prof. Malileh Izadi, Aral de Moor

Supervisors

01. INTRODUCTION

Transformers have been the architecture of choice for Large Language models since their introduction in 2017. The last few years have seen incredible development in the capabilities showcased by these language models, with parameter count being the driving factor. This direction will soon hit a data limit, forcing us to look elsewhere for improvements.



Tokenization, which is a component of almost all modern language models, is one such direction. It is often understudied and neglected, as its usually trained separately from the main transformer. However, as we found, that does not make it less important. The parameters with which you train your tokenizer proved to have a significant impact on the model's language understanding and efficiency.

REFERENCES

[1]. Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In Katrin Erk and Noah A. Smith, editors, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics

[2]. Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates, April 2018. arXiv:1804.10959 [cs].

[3]. Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing, August 2018. arXiv:1808.06226 [cs].

02. TOKENIZATION

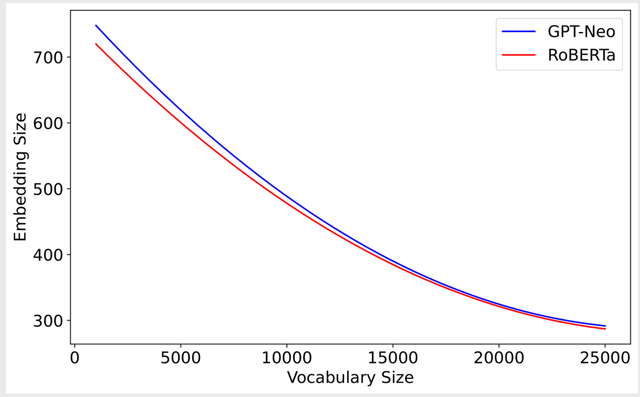
Splitting on words or characters are two of the simplest ways to tokenize natural language inputs. These strategies are on two opposing sides of the tokenization spectrum, which balances robustness to OOV (Out-of-vocabulary) words, language compression and efficiency.

Character Tokenization	Optimizing token granularity.
Subword Tokenization	Optimizing token granularity.
Word Tokenization	Optimizing token granularity.

The tokenizers of most modern language models sit between those two, using the subword paradigm. We test different strategies of subword vocabulary creation:

- **Byte-Pair Encoding** [1] - merges on frequency
- **WordPiece** [2] - merges on data likelihod
- **SentencePiece** [3] - prunes on data likelihood

We also test tokenization's main parameter: **vocabulary size**, as a trade-off with embedding size while fixing model size.

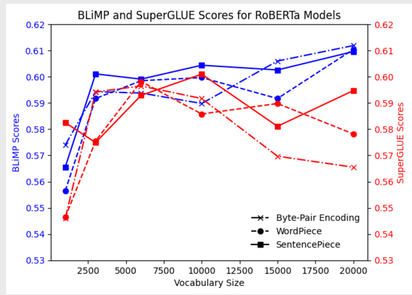


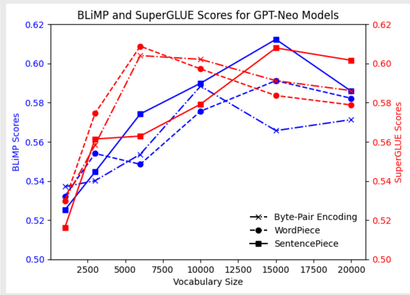
03. EXPERIMENTAL SETUP

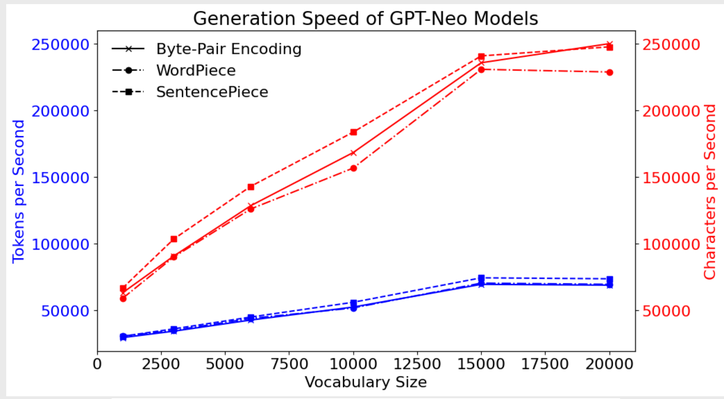
We sample 6 points from the trade-off (1k, 3k, 6k, 10k, 15k, 20k) for each of the tokenization strategies, and measure language understanding through BLiMP and SuperGLUE, while assessing efficiency with both generative and inference speed. We train the tokenizers on the BabyLM dataset and pretrain GPT-Neo and RoBERTa models on the TinyStories dataset.

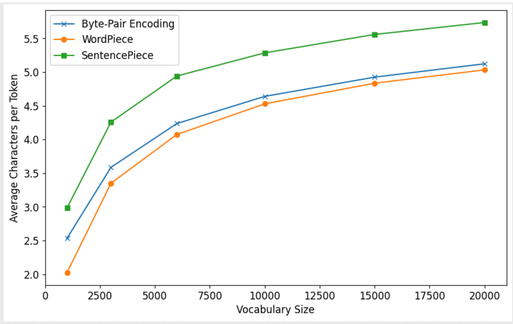
04. RESULTS

Results show that increasing vocabulary size beyond 6,000 tokens for RoBERTa and 10,000 for GPT-Neo in 9M parameter models does not enhance language understanding. However, generation speeds demonstrate a steady rise in tokens per second and a sharp increase in characters per second.









05. DISCUSSION & FUTURE WORK

Language models like those we tested do not benefit from increased vocabulary size beyond a certain threshold, likely because additional tokens are rarer rather than larger aggregates.

We also find that a larger embedding size negatively affects generation speed more than the corresponding reduction in vocabulary size benefits it.

Future work in addressing some of the validity threats present in this study, as well as an isolated test of both vocabulary size and embedding size would improve our understanding of the impact of tokenization on LM.