Tokenization Matters: Training your Tokenizer Right

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

Rafael Braga Medeiros Mota Borges

Supervisor(s): Prof. Arie van Deursen, Asst. Prof. Maliheh Izadi, Aral De Moor

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Rafael Braga Medeiros Mota Borges
Final project course: CSE3000 Research Project
Thesis committee: Assoc. Prof. Thomas Abeel, Prof. Arie van Deursen, Asst. Prof. Maliheh Izadi, Aral de Moor

Abstract

Large language models (LLMs) are rapidly increasing in parameter count, but this growth is not matched by an availability of high-quality data. This discrepancy raises concerns about the sustainability of current approaches to language model improvement, especially as forecasts suggest a potential data shortage by the end of the decade. This study investigates the impact of different tokenization strategies on the performance of small transformer (around 10 million parameters) models, evaluating three prominent subword tokenization methods: Byte-Pair Encoding (BPE), WordPiece, and SentencePiece. Additionally, we examine the trade-off between vocabulary size and embedding size and measure these factors' effects on language understanding and model efficiency within the BabyLM pipeline with BLiMP and SuperGLUE scores. Our findings indicate that while the different tokenization strategies have minimal impact on model performance, the trade-off between vocabulary size and embedding size significantly affects both language understanding and efficiency. Increasing the vocabulary size beyond a certain threshold does not seem to enhance language understanding. This research improves our understanding of how tokenization influences the language modeling process, specifically within the context of small language models.

1 Introduction

The introduction of the transformer architecture in 2017 [13] revolutionized the field of language modeling. Since then, the size of these models has increased dramatically, from 1.5 billion parameters with GPT-2 in 2019 to over 1 trillion parameters with Google's Switch Transformers by 2022 [4]. However, this rapid expansion in size requires a corresponding increase in high-quality data to effectively train these models—a requirement that is not expected to be met in the near future. Forecasts suggest a potential shortfall of high-quality data by 2026 [14], prompting a need to explore alternative methods for enhancing language model performance. Although Large Language Models (LLMs) process significantly more information over their training cycles than a human does in a lifetime, they do not yet leverage this information efficiently. Enhancing sample efficiency can be an alternative direction to improve model performance.

Recent studies have shown that smaller language models ($<$ 33 million parameters) can exhibit equivalent language understanding of their larger counterparts, and similarly display the desired emergent properties (e.g., reasoning and creativity) that drive their prevalence [3; 17]. Besides that, the small scale of these models makes them inherently more interpretable [3], while also making their local deployment easier, leading to better support for privacy, personalisation, and democratisation.

Research has yet to fully understand how architectural choices impact natural language understanding and task-specific performance in small LMs. Although studies like Eldan et al. [3] suggest that small LMs can outperform larger models like GPT-2 by narrowing their training data, quantitative evaluations in applied settings are lacking. Similarly, Warstadt et al. [17] identify architectural optimizations as a promising direction, yet comprehensive surveys of these decisions across models are rare. Moreover, studies on LMs applied to downstream tasks often neglect to explore beyond default hyperparameters, much less delve into architectural choices [19].

This research is dedicated to exploring a part of that gap, focusing on tokenization strategies and their parameters, on which there has been little to no published work. While existing studies on transformer models often reuse tokenizers from other projects without any modifications, it has been demonstrated that tailoring the tokenizer to the specific task can significantly improve results. Our study seeks to deepen the understanding of how tokenization impacts the performance of small language models (LMs), which remain under-explored in this context.

We investigate the three most prominent subword tokenization methods: Byte-Pair Encoding [12], WordPiece [10], and SentencePiece [7], exploring various vocabulary sizes for each. In our experiments, we analyze the trade-off between vocabulary size and transformer embedding size, maintaining a constant total parameter count to suit the *small* LM context of this research. This tradeoff allows us to analyse the balance between the information contained in a token, and the embedding size on which the model fits this information.

To comprehensively evaluate the impacts of our tokenization strategies and parameters, we utilized the BLiMP [18] and SuperGLUE [15] benchmarks to assess language understanding and tested model efficiency in inference and generative settings. We conducted experiments using both RoBERTa [8] and GPT-Neo [1] models, given that they represent the two main transformer architectures: encoder, and decoder. We pretrained them on the TinyStories dataset, which is known for its cognitive simplicity, allowing the models to focus primarily on language comprehension.

Our contributions from the study of the impact of different tokenization strategies and the trade-off between vocabulary size and embedding size on small LMs are as follows:

- We demonstrate that beyond a certain vocabulary threshold, there is no additional benefit to language understanding in our small model settings.
- We establish that there are no significant differences in language understanding or model efficiency among Byte-Pair Encoding, WordPiece, and SentencePiece tokenization strategies.
- We introduce a new method for measuring language model generation speed that accommodates comparisons across models with varying average token sizes.
- We provide a replication package[1] for reproducing our findings, and our models[2] published on HuggingFace.

[1] https://github.com/AISE-TUDelft/tiny-transformers

[2] https://huggingface.co/collections/AISE-TUDelft/brp-tiny-transformers-666c352b3b570f44d7d2a519

2 Related Works

2.1 Transformers

The introduction of the transformer architecture by Vaswani et al. in 2017 [13] has revolutionized the field of language modeling. This architecture powers well-known tools such as ChatGPT, Gemini, and Claude. Transformer architectures are typically categorized into three styles: decoder-only, encoder-only, and encoder-decoder hybrids. Decoder-only models, such as those used in GPT, employ causal self-attention to condition each prediction solely on preceding information, making them ideal for text generation tasks. Conversely, encoder-only models like BERT utilize bidirectional self-attention, allowing them to process both preceding and succeeding information, which is advantageous for tasks like text classification. Lastly, the encoder-decoder hybrids, as initially described by Vaswani et al., combine features of both architectures, using the encoder to process input sequences and the decoder to generate output sequences, making them suitable for sequence-to-sequence tasks such as machine translation, as can be seen in Figure 1.

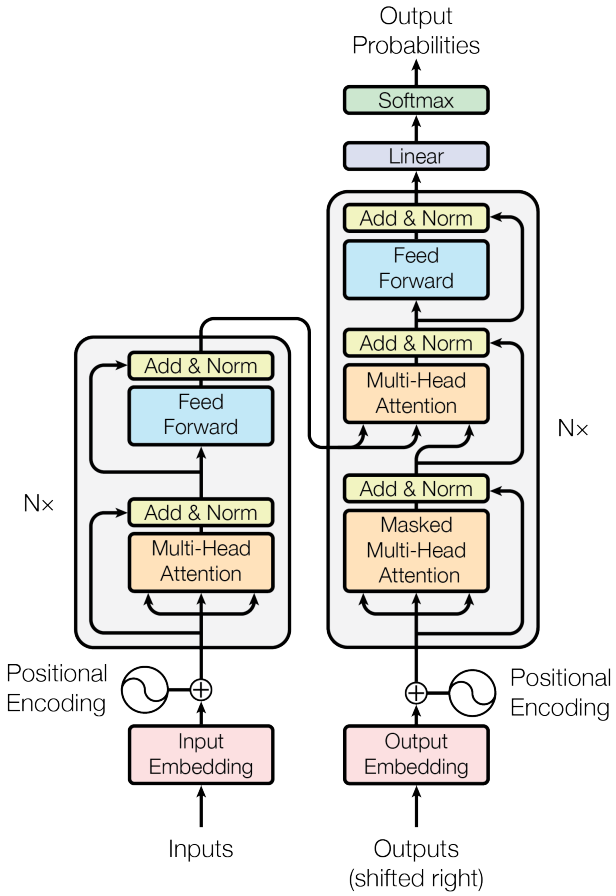

Figure 1: Encoder-Decoder hybrid transformer model architecture. [13]

2.2 Tokenization

Transformer architectures traditionally begin with a tokenization step, where natural language inputs are converted into vector-representable tokens. The basic method of splitting text into words offers the advantage of high compression, i.e., average length of a sentence measured in tokens. It does, however, result in a large vocabulary that increases the model size and creates Out-of-Vocabulary (OOV) issues [12]. OOV occurs when tokenizers, trained on a fixed corpus, encounter new words during training or inference, a common challenge with languages with agglutinative forms such as German [12]. In such cases, the model must default to an 'unknown' token, potentially ignoring close semantic relationships with word already present in the vocabulary.

To mitigate the Out-of-Vocabulary (OOV) issue, one effective approach is using a character tokenizer that splits inputs into individual characters. Since all words are composed of characters from a consistent set, such as Unicode or UTF-8, this method eliminates OOV tokens and results in a smaller vocabulary, reducing model size. However, character tokenization significantly lowers the degree of compression, meaning that the fixed-size context window of traditional transformer models will contain less information.

Word and character tokenization methods represent two extremes in managing vocabulary size. Most tokenizers opt for a middle ground approach known as subword tokenization, which balances the vocabulary size with a high degree of compression. This method involves creating tokens for frequently used character combinations, including common words like 'the' and crucial morphemes like 'aly' and 'ize'. It also retains character tokens to address unseen words effectively, thereby eliminating the OOV problem. A visual representation of this tradeoff can be seen in Figure

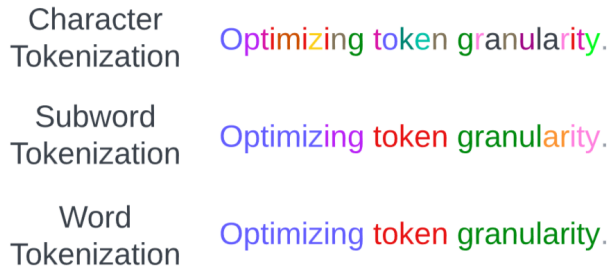

Figure 2: Representation of Character, Word, and Subword tokenization.

There have been three main subwork tokenization algorithms:

- **Byte Pair Encoding (BPE)** [12] - It starts with a token vocabulary composed out of characters. It then iteratively merges the tokens that are most frequently seen consecutively in the training data until the desired vocabulary size is reached.
- **WordPiece** [6] - Works identically to BPE except for the token merging criterion. Instead of merging the tokens that are most frequently seen consecutively, it merges the tokens that increase the likelihood of the data by the

greatest amount, e.g., $\{c \rightarrow e : 10, c \rightarrow v : 1, u \rightarrow d : 20, u \rightarrow a : 20\}$ will lead to c and e being merged.

- **SentencePiece** [7] - This algorithm isn't a subword tokenization method itself, but a wrapper on subword tokenization methods. It supports both BPE and Unigram Language Model [6] as its base subword tokenizers. It works by pruning a large vocabulary of tokens via entropy reduction while minimizing the impact on a specified loss function (usually related to data likelihood).

Although subword tokenization is the most prominent in current language models, a new wave of token free approaches have been proposed in the last few years [21] [20]. Token free approaches remove the cumbersome and sometimes language dependent process that is tokenization, making the training of the transformer a truly end-to-end process. Token free transformers also have the benefit of being modality agnostic, since they can learn directly from the input byte data. Two of the most notable proposed transormer based token free approaches have been MegaByte [21] and ByT5 [20].

3 Approach

Tokenization as a component of the language modelling process is is frequently overlooked and under-researched. Most work on transformer models reuse, without adaptation, already trained tokenizers from other projects. The GPT-2 tokenizer, for example, is still one of the most used tokenizers despite it being almost 5 years since its release. And this persistent neglect of tokenization does not stem from a lack of influence on model performance. On the contrary, it has been shown that fine-tuning the tokenizer for the specific task at hand can yield significant results [2].

The main purpose of this work is to enhance our understanding of the impact of tokenization on language modeling, particularly in small transformers with fewer than 33 million parameters. We aim to systematically compare various tokenization strategies, focusing also on their most relevant parameter: vocabulary size. The literature reveals a notable lack of studies examining different tokenization methods and vocabulary sizes across both small and larger language models (LLMs).

3.1 Tokenization Strategies

Over the past decade, various algorithms have been proposed to generate token vocabularies for language models, primarily focusing on subword tokenization. These methods typically operate by iteratively analyzing the frequency of subword units in a specific text corpus, leading to vocabularies that capture fundamental morphological elements, such as morphemes. This approach allows models to efficiently decompose words into meaningful, reusable components, which aids in learning efficiency.

Although these various algorithms all aim to identify the most common subwords within a text corpus, they differ in the criteria they employ for this task. These differences naturally result in distinct token vocabularies, even when applied to the same training data. It is then relevant to assess how these variances in tokenization strategies affect language understanding and the speed of inference and generation. For

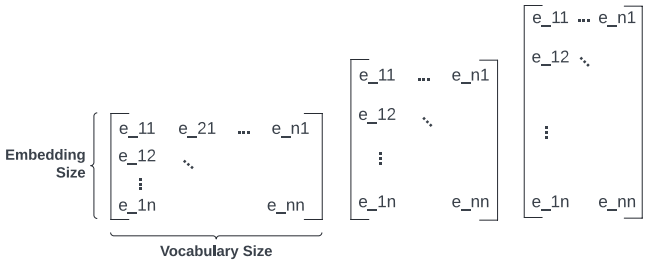

Figure 3: Visual representation of the trade-off between vocabulary size and embedding dimession in a transformer embedding matrix. As the embedding size increases, the vocabulary size decreases.

3.2 Vocabulary Size - Embedding Size Tradeoff

All of the subword tokenization strategies we examine revolve around an important parameter: vocabulary size. This parameter has an important impact on model performance, as a larger vocabulary usually has larger tokens, containing higher information on average and thus increasing model contextual reach. The vocabulary size is responsible for the number of columns on the embedding matrix, with a larger vocabulary increasing the width of the matrix as can be seen in Figure 3.

The other dimension of the embedding matrix is known as the embedding size. This dimension determines the number of features each token embedding will have. The choice of embedding size also influences model performance; a smaller embedding size compacts the token embeddings into a more constrained vector space, which can lead to a loss of information and diminish the model's ability to discern certain tokens. On the other hand, while a larger embedding size might enhance the model's ability to distinguish token embeddings, it also increases the computational load and potential for overfitting.

Both vocabulary size and embedding size influence model performance, as well as the speed of inference and generation. Our research aims to determine how, within the constraints of a fixed total parameter count, different allocations of vocabulary size and embedding size affect the model's language understanding capabilities and its operational speed. An intuition for how these parameters interact with each other can be observed in Figure 3.

4 Experimental Setup

4.1 Research Questions

The concrete research questions we aim to answer are the following:

1. How does the use of different tokenization strategies (Byte-Pair Encoding, WordPiece, SentencePiece) impact the performance metrics of small scale (< 33 million parameters) transformers in terms of language understanding, inference speed, and generation speed?
2. In small-scale transformers, how does balancing vocabulary size with embedding size influence model performance in terms of language understanding, inference speed, and generation speed?

4.2 Datasets

To train the three sub-word tokenization algorithms—Byte-Pair Encoding [12], WordPiece [6], and SentencePiece [7]—we use the BabyLM dataset [16]. This dataset comprises 100 million words of both written and spoken English, featuring children's content and general adult dialogue. It was specifically selected for its suitability for small-scale language models, focusing on fundamental language understanding while minimizing domain-specific knowledge. This allows the tokenizer to better generalize across various forms of English. Additionally, the diverse linguistic content of BabyLM makes it befitting of the model performance evaluation described in 4.4, which has a very large focus on linguistic coherence, while putting less emphasis on fact remembering and complex reasoning.

For the pre-training of the language models, we utilize the TinyStories dataset [3], which contains 2.1 million short children's stories generated by GPT 3.5 and GPT 4, with each story averaging about 175 words. This dataset was specifically chosen for its emphasis on grammatical structure and cognitive simplicity, characteristics that are beneficial for small language models. Importantly, it is also devoid of specialized domain knowledge, thereby allowing the models to concentrate on modelling fundamental sentence structure and grammar. Previous research has demonstrated that small language models (with fewer than 33 million parameters) pretrained on this dataset have surpassed GPT-2 (which has 125 million parameters) in tasks measuring natural language understanding [3].

4.3 Models and Tokenizers

For all of the subwork tokenizers trained, we pretrain a RoBERTa and a GPT-Neo model. We make use of their respective Hugging Face implementations[3] [4]. We test with these two model architectures as they represent the two main transformer styles available: decoder and encoder respectively. There have been newer decoder and encoder style models proposed, with more complex optimizations such as flash-attention, but these possess the simplicity that is suitable for the scope of this project. All of the models trained had 2 transformer blocks, 4 attention heads within each block, a context window of 512 tokens and an intermediate size of 1024, totalling a fixed ballpark of 9 million parameters, and we trained the models for one epoch.

For each of the subword tokenization strategies, we trained six tokenizers with the following vocabulary sizes: 1,000; 3,000; 6,000; 10,000; 15,000; and 20,000. To maintain a fixed parameter count across models, corresponding embedding sizes were adjusted to 780, 700, 604, 516, 412, and 348, respectively. This range of values was chosen to cover a broad spectrum within the vocabulary size - embedding size spectrum, facilitating the understanding of the significance of this trade-off, particularly in terms of model efficiency and language understanding. It is worth noting that we have trained the SentencePiece tokenizer with the Unigram model [6] as its base.

4.4 Evaluation Metrics

Language Understanding

To evaluate all of pretrained models, we utilise of the BabyLM evaluation pipeline [5] [16], which includes three main components: BLiMP, GLUE, and SuperGLUE. The BLiMP component specifically measures the language understanding of models by presenting pairs of sentences—one grammatically correct and the other incorrect—and scoring the model based on its ability to assign a higher likelihood to the correct sentence [18; **?**]. GLUE and SuperGLUE, on the other hand, extend the evaluation to a broader range of language understanding and reasoning tasks, following a fine-tuning process. The BabyLM pipeline integrates both GLUE and SuperGLUE tasks.

Generation Speed

To accurately evaluate the efficiency of different subword tokenization strategies and the trade-off between vocabulary size and embedding size, we focused on measuring the text generation speed of GPT-Neo models. RoBERTa models, which are not autoregressive without finetuning, were excluded from this test. Traditionally, generation speed is measured in tokens per second. However, this metric proved inadequate due to variations in vocabulary sizes, as models with smaller vocabularies generate fewer, larger tokens compared to those with larger vocabularies. Since we wanted to measure the speed at which information is generated, and not tokens specifically, include another measure. Initially we considered using bits per second, but this approach tended to overestimate the speed of models generating more characters that require larger encoding sizes, such as those from the extended ASCII set. To ensure a more accurate measurement, we opted to assess generation speed in characters per second, explicitly excluding spaces and any characters specific to the tokenization algorithms.

All speed tests were conducted using an NVIDIA GeForce RTX 3080 GPU with 10 GB of VRAM. For each model, inputs were generated with a batch size of 256, each sequence consisting of 32 tokens and producing outputs of 250 tokens. Prior to testing, we execute a warm-up stage consisting of three runs using a batch size of 150 to ensure GPU optimization. The performance metrics were then calculated by averaging the results over 25 consecutive runs.

Inference Time

To assess the efficiency of the RoBERTa and GPT-Neo models, we implemented a test that measures the time taken

[3] https://huggingface.co/docs/transformers/model_doc/gpt_neo

[4] https://huggingface.co/docs/transformers/model_doc/roberta

[5] https://github.com/babylm/evaluation-pipeline-2023

for a forward pass through the transformer models using a specific dataset. For this purpose, we selected sentences from the *argument structure subtask* of the BLiMP dataset, which consists of 16,496 short sentences, due to their structural simplicity. We recorded the total time required to process all these sentences in sequence, providing a comprehensive measure of inference speed. All tests were conducted on an NVIDIA GeForce RTX 3080 GPU with 10 GB of VRAM.

5 Results

5.1 Language Understanding Evaluation

The relationship between vocabulary size and embedding size is illustrated in Figure 4, revealing a non-linear relationship that resembles exponential decay. This indicates that as vocabulary size increases, only minor reductions in embedding size are necessary to achieve the same proportional increase in vocabulary, whilst maintaining total parameter count. This curves were obtained under the specific model parameters outlined in Subsection 4.3.

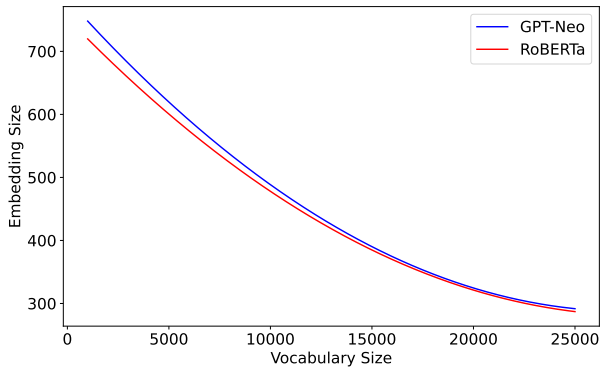

Figure 4: Vocabulary size vs. the Embedding size per token in the vocabulary for a model with 9M parameters.

In Figure 5, we analyze the performance of different tokenization strategies and vocabulary sizes on the BLiMP and SuperGLUE language understanding metrics for RoBERTa models. We can observe that there is a sharp decrease in both BLiMP and SuperGLUE scores for the vocabulary sizes of 1000 tokens, when compared to the remaining vocabulary sizes. From vocabulary sizes of 3000 tokens onwards, the BLiMP scores stabilize, showing little variation across increases in vocabulary size, which indicates a plateau in performance improvement. The SuperGLUE scores, on the other hand, show a slight negative trend as vocabulary sizes increase, suggesting that the decrease in embedding size might be affecting model performance. The variation in scores between the different tokenization strategies (Byte-Pair Encoding, WordPiece, and SentencePiece) do not exhibit a consistent or significant pattern.

The same analysis as described above can be found for GPT-Neo models in Figure 6. This analysis reveals a notably lower performance at a vocabulary size of 1000 tokens for all tokenization strategies. For the Byte-Pair Encoding and WordPiece strategies, there is a consistent increase in BLiMP scores as the vocabulary size expands up to the 10,000 token mark, after which the scores plateau, suggesting diminishing returns beyond this point. The SentencePiece strategy shows improvements up to the 15,000 token mark, beyond which its performance declines. A similar pattern is observed with SuperGLUE scores, where SentencePiece deteriorates post-15,000 tokens, whereas Byte-Pair Encoding and WordPiece peak at the 6,000 token mark before stabilizing.

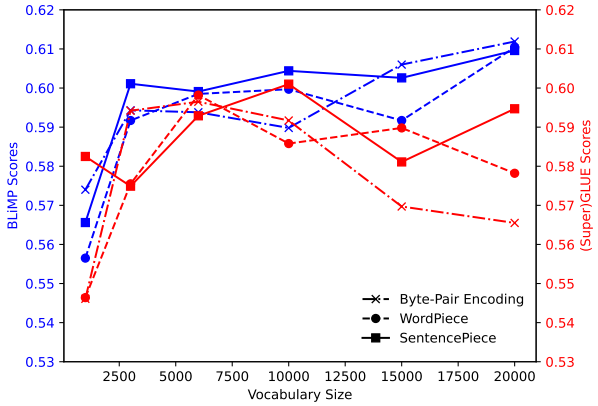

Figure 5: RoBERTa model scores with Byte-Pair Encoding, WordPiece, and SentencePiece tokenizers on both SuperGLUE and BLiMP tasks for the different vocabulary sizes.

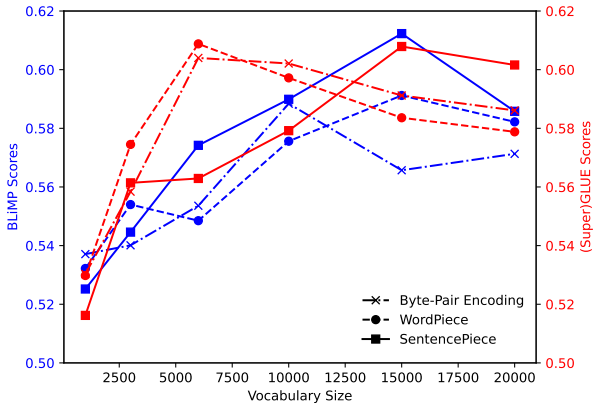

Figure 6: GPT-Neo model scores with Byte-Pair Encoding, WordPiece, and SentencePiece tokenizers on both SuperGLUE and BLiMP tasks for the different vocabulary sizes.

5.2 Efficiency Assessment

In Figure 7 we can see how an increase in vocabulary size affects the average token length for each of the tokenization strategies. We can observe a clear logarithmic pattern, which indicates that new tokens aren't significantly larger. SentencePiece also showcases a distinctly larger average token size when compared to the other two approaches.

In Figure 8, the generation speed, measured in tokens per second and characters per second for the GPT-Neo models is depicted. The results, for all of the tokenization strategies, show a consistent growth, in tokens per second, for vocabulary sizes until the 15,000 mark, after which it stabilises. The

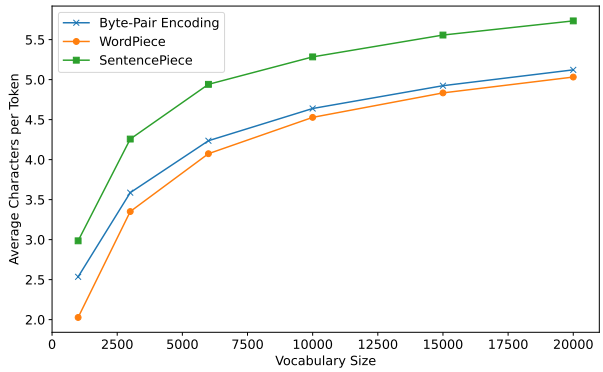

Figure 7: Average characters per token for Byte-Pair Encoding, WordPiece, and SentencePiece tokenizers for the different vocabulary sizes.

character per second measurement, shows the same pattern, with a higher slope, due to the increase in average token size.

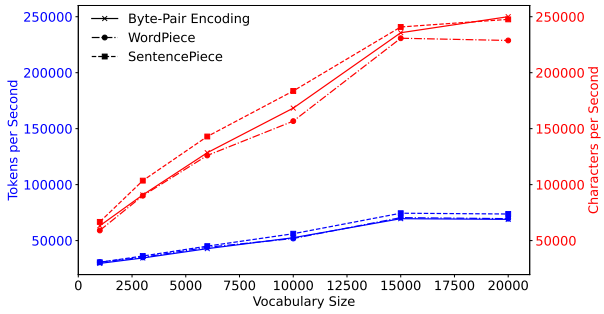

Figure 8: GPT-NEO model geneneration speed with Byte-Pair Encoding, WordPiece, and SentencePiece tokenizers measured in both tokens/s and chars/s for the different vocabulary sizes.

In Figure 9 the inference time for both GPT-NEO and ROBERTA models is depicted. There seems to be no significant impact of both tokenization strategies and the vocabulary sizes on the inference time for the ROBERTA models. On the other hand, for GPT-NEO models, there seems to be an increase in inference time until the 10,000 token mark, and a plateauing afterwards.

The detailed numerical results can be observed in Appendix A, as well as on our Wandb[6] page

6 Discussion

6.1 Implications

The findings from Subsection 5.2 reveal that there is no significant difference in language understanding or model efficiency among the three tokenization strategies studied. However, it is notable that GPT-NEO models using the SentencePiece tokenizer demonstrate a marginally higher characters-per-second rate compared to those using Byte-Pair Encoding and WordPiece. This increased rate is likely attributable to SentencePiece generating slightly larger tokens on average for the same vocabulary size, which suggests a potential efficiency advantage in certain contexts.

[6]https://wandb.ai/tiny-transformers/tokenizers

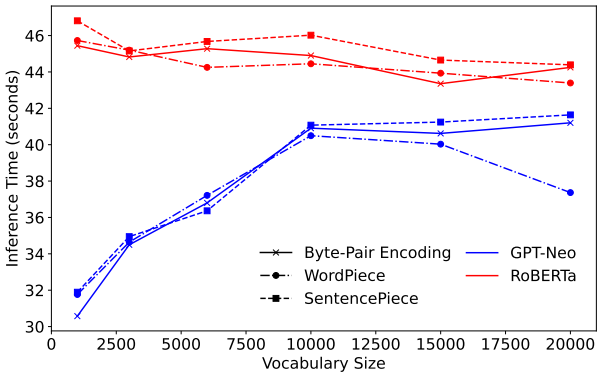

Figure 9: Inference time for Byte-Pair Encoding, WordPiece, and SentencePiece tokenizers with both ROBERTA and GPT-NEO models for the different vocabulary sizes.

The tradeoff between vocabulary size and embedding size significantly impacts both model performance and efficiency. For ROBERTA models, increasing the vocabulary beyond 6,000 tokens does not seem to improve language understanding, and larger vocabularies actually seem to degrade SuperGLUE scores sligthly. This decline could be attributed to the reduced embedding size, which limits the model's ability to handle the cognitive complexity of SuperGLUE tasks compared to simpler BLiMP tasks (which don't observe this same degradation), as a smaller embedding space restricts the model's capacity to encode nuanced information. Moreover, it could also be the case that these extra tokens may remain unused, offering no benefits in terms of practical data compression.

With regards to GPT-NEO models, the pattern is slightly different. Unlike ROBERTA models, which show no benefits from increasing vocabulary size beyond 6,000 tokens, GPT-NEO models continue to improve up to the 10,000 token threshold. The reason for this disparity is not fully understood, but it may be related to the unidirectional self-attention mechanism in GPT models, which perhaps does not utilize the additional embedding size (which is 604 vs 516) as effectively as the bidirectional attention mechanism in ROBERTA models.

With regards to model efficiency, the trade-off seems to also have a significant impact. From 10,000 tokens onwards, the inference time for both GPT-NEO and ROBERTA models seem to stabilize, with GPT-NEO being faster than ROBERTA, likely due to unidirectional self attention. The decrease in inference time for GPT-NEO models with vocabularies smaller than 10,000 is intruiguing, specially when the ROBERTA models show a consist inference time across all vocabulary sizes. This is especially intruiguing considering that GPT-NEO generation speeds show a monotonically increasing generation speed for larger vocabularies. Answering this questions requires further research.

With regards to model efficiency, the trade-off also appears to have a significant impact. Beyond 10,000 tokens, inference times for both GPT-NEO and ROBERTA models stabilize, with GPT-NEO consistently outperforming ROBERTA in speed, likely due to its unidirectional self-attention mechanism. Interestingly, while GPT-NEO models

show a decrease in inference time for vocabularies smaller than 10,000, RoBERTa models maintain consistent inference times across all vocabulary sizes. This discrepancy is particularly noteworthy given that GPT-Neo demonstrates monotonically increasing generation speeds for larger vocabularies. GPT-Neo models show faster generation speed but slower inference time, with larger vocabulary sizes. Further research is required to answer why that is the case.

6.2 Limitations

Internal validity evaluates whether the observed results are truly caused by the independent variables being tested, rather than by external influences. External validity concerns the applicability of the study's findings to different contexts outside the study itself. Construct validity assesses whether the measurement tools accurately reflect the intended concepts of the study.

Internal Validity

One threat to the internal validity of our findings is the computational limitations encountered during this project. We were only able to train the models for one epoch, likely resulting in underfitting, which introduces additional variance to the results. Strengthening the validity of our findings could be achieved by also training multiple models with different seeds for each tokenization algorithm and vocabulary size. Additionally, testing a broader range of vocabulary sizes would help provide a more robust interpretation of the trade-offs involved.

External Validity

The main threat to the external validity of this study stems from the experimental setup, particularly the small size of the models used. At 10 million parameters, these models are considered small by today's standards, even within the scope of small transformers. This raises questions about the applicability of our findings to larger models, which may exhibit different capabilities. These models also don't include many of the optimizations that have been introduced in the last years, which can put into question the applicability of the findings.

Construct Validity

The language understanding metrics employed, while comprehensive, may not capture all facets of language understanding. Similarly, the efficiency measures, though rigorously tested through numerous iterations and a warmup process, might still vary under different hardware and settings. Testing these metrics across diverse environments could potentially yield more precise and universally applicable results.

6.3 Future Work

Future work could extend our understanding of the impact of tokenization and its parameters on model performance and efficiency in several ways. Replicating the experiments conducted in this study and specifically addressing the threats outlined in Subsection 6.2 would be particularly valuable.

From the results discussed in Subsection 6.1, there appears to be a threshold in vocabulary size beyond which there is no increase in language understanding. Given our focus on the trade-off between vocabulary size and embedding size, it seems that the embedding size in these models may be unnecessarily large. Future research could explore optimizing embedding size to determine the smallest possible dimension that does not compromise language understanding. Additionally, investigating the specific requirements of embedding size relative to vocabulary size would be valuable, as our study observed these two factors moving in opposite directions.

Additionally, further investigation into the impacts of vocabulary size and embedding size on model efficiency could prove beneficial. As highlighted in Subsection 6.1, there are contradictory trends in inference and generation speeds observed in GPT-Neo models. Delving deeper into the reasons behind these divergent trends could provide valuable insights.

7 Conclusion

This study explored the impact of different tokenization strategies and the trade-off between vocabulary size and embedding size on small language models. We found that while Byte-Pair Encoding, WordPiece, and SentencePiece strategies do not significantly affect language understanding, SentencePiece does lead to higher generation speeds due to its tendency to generate larger tokens.

Furthermore, our analysis revealed that beyond a certain vocabulary threshold, there is no additional benefit to language understanding. The trade-off between vocabulary size and embedding size significantly influences model efficiency and generation speed. Specifically, an increase in vocabulary size, which correlates with an increase in average character per token, significantly impacts generation speeds.

8 Responsible Research

To ensure the robustness of our results, we ensured that the datasets used for pretraining and evaluating our models were distinct, thereby preventing test set contamination. As emphasized in recent studies on pretraining methodologies [9], pretraining on the evaluation dataset can artificially inflate performance metrics, thereby undermining the validity of the research findings.

To address the ongoing reproducibility crisis in the field of deep learning—a critical issue highlighted by Semmelrock et al. [11]—we have provided a comprehensive replication package. This package includes all the source code needed to reproduce our results. We also provided all of the relevant hyperparameters and hardware specifications in Section 4. By doing so, we aim to facilitate exact replications of our findings, thereby enhancing the reliability and transparency of our research.

In accordance with the Netherlands Code of Conduct for Research Integrity, we conduct and report all our results with honesty, transparency, and responsibility, upholding the highest standards of research integrity. We adhere to the educational and normative frameworks outlined in chapters 2 and 3 of the Code [5], emphasizing good research practices.

References

[1] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large Scale Autoregres-

sive Language Modeling with Mesh-Tensorflow, March 2021.

[2] Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. Getting the most out of your tokenizer for pre-training and domain adaptation, February 2024. arXiv:2402.01035 [cs].

[3] Ronen Eldan and Yuanzhi Li. TinyStories: How Small Can Language Models Be and Still Speak Coherent English?, May 2023. arXiv:2305.07759 [cs].

[4] William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity, June 2022. arXiv:2101.03961 [cs].

[5] KNAW, NFU, NWO, TO2-Federatie, Vereniging Hogescholen, and VSNU. Nederlandse gedragscode wetenschappelijke integriteit, 2018.

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

[7] 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].

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, July 2019. arXiv:1907.11692 [cs].

[9] Rylan Schaeffer. Pretraining on the Test Set Is All You Need, September 2023. arXiv:2309.08632 [cs].

[10] Mike Schuster and Kaisuke Nakajima. Japanese and Korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, March 2012. ISSN: 2379-190X.

[11] Harald Semmelrock, Simone Kopeinik, Dieter Theiler, Tony Ross-Hellauer, and Dominik Kowald. Reproducibility in Machine Learning-Driven Research, July 2023. arXiv:2307.10320 [cs, stat].

[12] 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.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs].

[14] Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? Limits of LLM scaling based on human-generated data, June 2024. arXiv:2211.04325 [cs].

[15] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems, February 2020.

[16] Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. Call for Papers – The BabyLM Challenge: Sample-efficient pretraining on a developmentally plausible corpus, January 2023. arXiv:2301.11796 [cs].

[17] Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. Findings of the BabyLM Challenge: Sample-Efficient Pretraining on Developmentally Plausible Corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–6, Singapore, 2023. Association for Computational Linguistics.

[18] Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The Benchmark of Linguistic Minimal Pairs for English, February 2023.

[19] Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. Should You Mask 15% in Masked Language Modeling?, February 2023. arXiv:2202.08005 [cs].

[20] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models, March 2022. arXiv:2105.13626 [cs].

[21] Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers, May 2023. arXiv:2305.07185 [cs].

A Tables With Numerical Results

This appendix contains tables with the numerical results on which the graphs from section 5 are based. Here you can find the results for each model, which is defined with the name. BERT refers to the RoBERTa models while GPT refers to GPT-Neo models. The after that comes the vocabulary size (1k, 3k, 6k, 10k, 15k, 20k). After that the tokenizer is specified. SentencePiece with "sp", Wordpiece with "wp", and Byte-Pair Encoding with "bpe".

Table 1: BLiMP and SuperGlue Results for BERT Models

Name	BLiMP Avg	SuperGlue Avg
BERT 1k-sp	0.566	0.582
BERT 1k-bpe	0.574	0.546
BERT 1k-wp	0.556	0.546
BERT 20k-sp	0.610	0.595
BERT 15k-sp	0.603	0.581
BERT 10k-sp	0.604	0.601
BERT 6k-sp	0.599	0.593
BERT 3k-sp	0.601	0.575
BERT 20k-wp	0.610	0.578
BERT 15k-wp	0.592	0.590
BERT 10k-wp	0.600	0.586
BERT 6k-wp	0.598	0.598
BERT 3k-wp	0.592	0.576
BERT 20k-bpe	0.612	0.566
BERT 15k-bpe	0.606	0.570
BERT 10k-bpe	0.590	0.592
BERT 6k-bpe	0.594	0.597
BERT 3k-bpe	0.594	0.594

Table 2: BLiMP and SuperGlue Results for GPT Models

Name	BLiMP Avg	SuperGlue Avg
GPT 1k-sp	0.525	0.516
GPT 1k-bpe	0.537	0.530
GPT 1k-wp	0.532	0.530
GPT 20k-sp	0.586	0.602
GPT 15k-sp	0.612	0.608
GPT 10k-sp	0.590	0.579
GPT 6k-sp	0.574	0.563
GPT 3k-sp	0.545	0.561
GPT 20k-wp	0.582	0.579
GPT 15k-wp	0.591	0.584
GPT 10k-wp	0.576	0.597
GPT 6k-wp	0.549	0.609
GPT 3k-wp	0.554	0.575
GPT 20k-bpe	0.571	0.586
GPT 15k-bpe	0.566	0.591
GPT 10k-bpe	0.589	0.602
GPT 6k-bpe	0.554	0.604
GPT 3k-bpe	0.540	0.558

Table 3: GPT Models Generation Speed Metrics

Name	Tokens per Second	Chars per Second
GPT 1k-sp	30,626	66,708
GPT 1k-bpe	29,635	63,235
GPT 1k-wp	30,797	59,054
GPT 20k-sp	73,761	247,641
GPT 15k-sp	74,466	240,839
GPT 10k-sp	56,128	183,672
GPT 6k-sp	45,049	142,986
GPT 3k-sp	36,223	103,568
GPT 20k-wp	69,601	228,795
GPT 15k-wp	70,511	230,779
GPT 10k-wp	51,880	156,725
GPT 6k-wp	44,006	126,051
GPT 3k-wp	35,046	90,094
GPT 20k-bpe	68,969	250,035
GPT 15k-bpe	69,571	235,701
GPT 10k-bpe	52,555	168,361
GPT 6k-bpe	42,799	128,505
GPT 3k-bpe	34,470	90,808

Table 4: BERT Models Inference Time

Name	Inference Time (s)
BERT 1k-sp	46.81
BERT 1k-bpe	45.44
BERT 1k-wp	45.73
BERT 20k-sp	44.39
BERT 15k-sp	44.65
BERT 10k-sp	46.02
BERT 6k-sp	45.68
BERT 3k-sp	45.15
BERT 20k-wp	43.39
BERT 15k-wp	43.93
BERT 10k-wp	44.45
BERT 6k-wp	44.25
BERT 3k-wp	45.20
BERT 20k-bpe	44.25
BERT 15k-bpe	43.35
BERT 10k-bpe	44.90
BERT 6k-bpe	45.28
BERT 3k-bpe	44.82

Table 5: GPT Models Inference Time

Name	Inference Time (ms)
GPT 1k-sp	31.89
GPT 1k-bpe	30.57
GPT 1k-wp	31.76
GPT 20k-sp	41.64
GPT 15k-sp	41.24
GPT 10k-sp	41.07
GPT 6k-sp	36.36
GPT 3k-sp	34.95
GPT 20k-wp	37.37
GPT 15k-wp	40.03
GPT 10k-wp	40.49
GPT 6k-wp	37.21
GPT 3k-wp	34.64
GPT 20k-bpe	41.20
GPT 15k-bpe	40.62
GPT 10k-bpe	40.91
GPT 6k-bpe	36.80
GPT 3k-bpe	34.50

Table 6: Tokenizer Characterization

Tokenizer Name	Average Characters per Token
sp (1k)	2.9849
bpe (1k)	2.5347
wp (1k)	2.0271
sp (3k)	4.2568
bpe (3k)	3.5870
wp (3k)	3.3513
sp (6k)	4.9413
bpe (6k)	4.2359
wp (6k)	4.0741
sp (10k)	5.2835
bpe (10k)	4.6394
wp (10k)	4.5279
sp (15k)	5.5566
bpe (15k)	4.9237
wp (15k)	4.8341
sp (20k)	5.7342
bpe (20k)	5.1218
wp (20k)	5.0324