Enhancing Large Language Models through Adaptive Tokenizers

Mengyu Zheng
The University of Sydney
Huawei Noah's Ark Lab
mzhe4259@uni.sydney.edu.au

Hanting Chen
Huawei Noah's Ark Lab
chenhanting@huawei.com

Tianyu Guo
Huawei Noah's Ark Lab
tianyu.guo@huawei.com

Chong Zhu
Huawei Noah's Ark Lab
zhuchong4@huawei.com

Binfan Zheng
Huawei GTS AI Computing LAB
zhengbinfan1@huawei.com

Chang Xu
The University of Sydney
c.xu@sydney.edu.au

Yunhe Wang*
Huawei Noah's Ark Lab
yunhe.wang@huawei.com

Abstract

Tokenizers serve as crucial interfaces between models and linguistic data, substantially influencing the efficacy and precision of large language models (LLMs). Traditional tokenization methods often rely on static frequency-based statistics and are not inherently synchronized with LLM architectures, which may limit model performance. In this study, we propose a simple but effective method to learn tokenizers specifically engineered for seamless integration with LLMs. Initiating with a broad initial vocabulary, we refine our tokenizer by monitoring changes in the model's perplexity during training, allowing for the selection of a tokenizer that is closely aligned with the model's evolving dynamics. Through iterative refinement, we develop an optimized tokenizer. Our empirical evaluations demonstrate that this adaptive approach significantly enhances accuracy compared to conventional methods, maintaining comparable vocabulary sizes and affirming its potential to improve LLM functionality.

1 Introduction

In recent years, large language models (LLMs) have emerged as foundational tools across a spectrum of applications in natural language processing [3, 7, 24]. From generating human-like text to enabling complex question-answering systems [28], LLMs have proven to be exceptionally versatile and capable. At the core of these models lies the tokenizer, a critical component that dictates how natural language is transformed into a format amenable to computational processing. The effectiveness of a tokenizer directly influences the model's ability to understand and generate language, thus playing a pivotal role in the overall performance of the LLM. Recognizing this integral relationship, it becomes essential to develop tokenizers that are not only effective but also dynamically adaptable to the evolving architectures of contemporary LLMs.

Current tokenization methods for large language models (LLMs) primarily include Byte Pair Encoding (BPE) [32], WordPiece [43], and Unigram [19], each serving to enhance text preprocessing by splitting

*Corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3608

it into manageable subwords. BPE focuses on reducing the dataset size through a greedy merging strategy based on character or subword frequency, effectively addressing the issue of infrequent words by splitting them into more common subunits. WordPiece, similar to BPE, starts with a base vocabulary and iteratively refines it by merging the most frequent pairs but incorporates a likelihood maximization step, which makes it slightly more context-aware than BPE. Unigram tokenization operates somewhat inversely, beginning with a large vocabulary and iteratively pruning it down based on token utility calculated through negative log likelihood, aiming to optimize the vocabulary against corpus loss metrics. Despite their efficiency in handling large vocabularies and improving computational feasibility, these tokenization methods are typically fixed once developed and are not designed to adapt or learn from the model's evolving understanding of language during training.

While traditional tokenization methods have been instrumental in enhancing the efficiency and effectiveness of large language models (LLMs), they are typically decoupled from the model's learning mechanisms. This means they do not adapt or evolve based on the model's performance or the specific requirements of the tasks being addressed. Instead, these methods prioritize compressing the vocabulary size, which can sometimes lead to suboptimal performance in complex language tasks where adaptability and contextual understanding are crucial. Recent advances in end-to-end learnable tokenization [17, 16, 38] aim to address these deficiencies by more closely integrating tokenization with the model's learning processes. However, these systems, while innovative, introduce significant computational overhead (*e.g.*, gradient-based tokenization and pooling modules) and lack the flexibility of traditional tokenization methods, which can be easily transferred across different models.

In this study, we address the limitations of traditional tokenization methods by creating a system where the tokenizer's development is coupled with the performance of the LLM itself. Specifically, we introduce an adaptive tokenizer that begins with a comprehensive initial vocabulary. As training progresses, we fine-tune this tokenizer by closely monitoring the model's perplexity. This ongoing adjustment allows the tokenizer to evolve in tandem with the LLM, ensuring that the tokenization process remains optimally aligned with the model's dynamic learning patterns. Our empirical results confirm that this adaptive approach markedly improves accuracy over traditional methods, demonstrating its potential to significantly enhance LLM functionality.

2 Related Works

2.1 Subword tokenizer

Subword tokenizers are widely applied in many large language models (LLMs) [32, 19, 42, 35, 20], such as GPT-3 [7], BERT [12], and T5 [30]. This is because subword tokenizers do not face the out-of-vocabulary issues that word-level tokenizers do. Unlike character-level tokenizers [36, 27], they do not require processing longer sequences at the character level, which significantly increases the complexity of the models quadratically [21]. Specifically, BPE (Byte Pair Encoding) [32] applies a compression algorithm [13] to the task of word segmentation. Unlike BPE, which builds a vocabulary from smaller to larger units, Unigram [19] starts by preparing a large seed vocabulary. The vocabulary is pruned until it reaches the specified size. Similar to the Unigram framework and its assumptions, BytePiece [35] trims the vocabulary based on token frequency to the required size. In its initial text encoding phase, BytePiece converts the corpus into bytes, enhancing the training speed of the tokenizer and achieving a higher compression rate. Obviously, these tokenizers are data-driven [5], with the generated vocabulary built based on the frequencies of word fragments [6]. However, they cannot directly interact with LLMs to enhance model capabilities [9].

2.2 Learnable tokenizer

Unlike traditional tokenizers that offer a predefined and fixed vocabulary, learnable tokenizers [17, 16, 38, 37, 34, 8] can be integrated with large language models into an end-to-end learning framework, resulting in task-specific tokenization to enhance the performance of LLMs. MANTa [16] introduces a gradient-based tokenization and pooling module that can be jointly learned with an encoder-decoder LLM [2]. RETVec [8] embeds words into a high-dimensional vector with a pre-trained model to be robust against adversarial attacks. Neural [17] adapts the tokenization behavior to the downstream task after pre-training the tokenization by distilling from a language-specific subword tokenizer. However, such tokenizers have high requirements for the quantity and quality of the training data. If

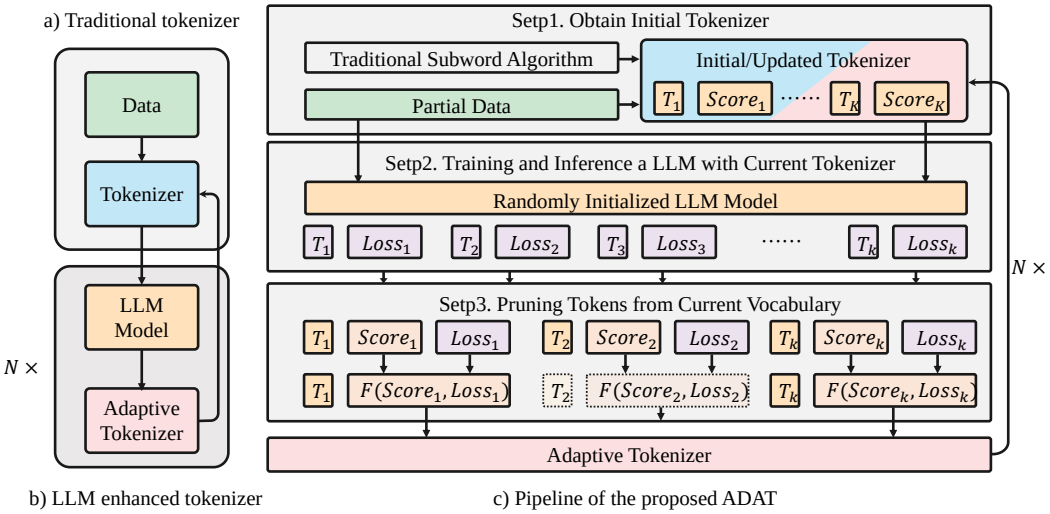

Figure 1: Illustration of the proposed ADAT pipeline. (a) The traditional tokenizer algorithm that directly extracts vocabulary from data. (b) The framework of the LLM-enhanced tokenizer, iteratively refining vocabulary based on model feedback. (c) Overview of ADAT, encompassing initial tokenizer acquisition, training and inference to derive token losses, token pruning based on scores and losses.

the data distribution is unbalanced or contains too much noisy data, it can lead to poor generalization of the tokenizer and negatively affect the performance of LLMs. For instance, Neural [17] requires a pre-training dataset curated with space-separated tokens and two carefully crafted heuristics to improve the ground label of the dataset. Therefore, stringent requirements for the quality of training data limit their widespread application.

In addition, some existing works have applied the concept of adaptive tokenizers in several fields, such as neural machine translation [26], domain adaptation [31], and text generation [23]. These task-adaptive tokenizers integrate tokenizers generated from different data distributions, focusing on how to combine the task-specific tokenizer with the other one. In contrast, our proposed model, ADAT, is designed to learn a general tokenizer. Therefore, the purpose of ADAT is different from that of the aforementioned adaptive tokenizers.

3 Adaptive Tokenizers

For Large Language Models (LLMs), two critical aspects are accuracy and inference speed, both of which are deeply intertwined with the design of the tokenizer. Specifically, an optimal vocabulary V with maximum size N in an objective dataset D can be described by the following optimization problem:

$$\min_{V} \textbf{Length}(D_o, V) - \lambda \textbf{Acc}(D, M, V), \quad |V| \leq N, \tag{1}$$

where M denotes the trained LLM, **Length** and **Acc** denotes the sequence length and accuracy (or the performance) using vocabulary V in dataset D, and the λ is a hyper-parameter to balance the two terms. This problem incorporates two primary objectives: given a fixed vocabulary size, the first is to maximize inference speed given a fixed vocabulary size, and the second is to maximize model accuracy. However, existing tokenizer schemes typically focus on one aspect over the other; for instance, traditional frequency-based schemes emphasize speed, while end-to-end approaches prioritize accuracy. To address this, we propose a method that optimizes both aspects. Therefore, we introduce an improved approach based on traditional frequency statistics, termed "adaptive tokenizers." This method aims to refine the balance between vocabulary efficiency and performance, thereby enhancing both the speed and accuracy of the model.

https://doi.org/10.52202/079017-3608

3.1 Unigram Model

Traditional tokenization methods generally fall into two categories: one approach, exemplified by Byte Pair Encoding (BPE) and WordPiece, starts with a small set of symbols and incrementally builds a larger vocabulary by merging the most frequent adjacent pairs. The other approach, typified by the Unigram method, begins with a large initial vocabulary which is progressively pruned based on token utility, a method found to generally offer superior performance due to its probabilistic foundation.

The Unigram model operates on the principle that the probability of a sentence is determined by the individual probabilities of its tokens. Here we briefly review the Unigram model. Initially, a large vocabulary V is established. This extensive initial set includes potentially every unique word or subword unit observed in the training corpus, ensuring that the vocabulary can cover all possible textual inputs. The process of refining the vocabulary involves several key steps repeated in cycles: 1. **Probability Estimation**: For each token x_i in the current vocabulary, we calculate its probability $p(x_i)$ based on its frequency of occurrence in the corpus. 2. **Loss Calculation**: We then compute the loss for each token, which is determined by how much the overall loss of the model would decrease if that token were removed. The loss function is calculated as:

$$\mathcal{L}_P(V) = \sum_{s=1}^{|V|} \log(p(X^{(s)}) = \sum_{i=1}^{|V|} \log(\sum_{\mathbf{x} \in S((X^{(s)})}^{P} (\mathbf{x})), \tag{2}$$

where $S(X)$ is a set of segmentation candidates built from the input sentence X, and $\mathbf{x} = (x_1, ..., x_K)$ is a subword sequence that $P(\mathbf{x}) = \prod_{i=1}^{K} p(x_i)$. The loss for each token x_i is then formulated as $\mathcal{L}_P(x_i) = \mathcal{L}_P(V) - \mathcal{L}_P(V - x_i)$. 3. **Token Pruning**: Tokens are ranked according to their calculated loss. Finally, a proportion of tokens contributing the most to increasing the overall loss is pruned from the vocabulary.

3.2 LLM-Enhanced Tokenization

To enhance the integration of Large Language Models (LLMs) with our tokenization process, we have developed a simple but effective refined method for calculating the loss associated with each token, incorporating insights directly from the LLM's performance metrics. This approach aims to optimize the tokenizer's vocabulary to better align with the LLM's understanding and generation of text. The framework is illustrated in Figure 1.

Our revised loss calculation method integrates the traditional Unigram model's frequency-based loss with a performance-driven loss derived from an LLM. Specifically, we first train an LLM M in the vocabulary D using a training dataset T. This model is designed to capture the linguistic nuances relevant to the tasks it is trained for, providing a robust framework for assessing token utility. For each token x_i in the vocabulary, we measure its individual contribution to the model's error using a cross-entropy loss function. The loss for each token is calculated as:

$$\mathcal{L}_M(x_i) = \sum_{x_i \in T} CE(M(x_{i-1}), x_i). \tag{3}$$

Here, CE represents the cross-entropy function, $M(x_{i-1})$ is the LLM's output given the previous token x_{i-1}, and x_i is the actual next token. This formula assesses how well the LLM predicts each token following its predecessor, providing a direct measure of each token's impact on model performance. Finally, The cross-entropy loss for each token is then combined with the traditional Unigram frequency-based loss. This combined loss ensures that tokens are evaluated not only on their frequency of occurrence but also on their actual contribution to the LLM's task performance. The final loss for pruning the vocabulary is given by:

$$\mathcal{L}(x_i) = F(\mathcal{L}_P(x_i), \mathcal{L}_M(x_i)), \tag{4}$$

where $F(\cdot, \cdot)$ is a function to balance the importance of frequency-based loss and LLM-driven loss, which will be discussed in experiments. Using this enhanced loss metric, we iteratively refine the vocabulary by pruning tokens that contribute the least to the combined loss, thus optimizing the vocabulary for both general language understanding and specific task performance. This process continues until the vocabulary is compact enough to manage while still being comprehensive enough to support the LLM effectively.

Random sampling. In the training of Large Language Models (LLMs), ensuring that each token within the set vocabulary receives equal and substantial training is crucial to prevent loss bias due to uneven training. While iterating over all possible tokenizations of the training corpus would ideally provide the most comprehensive learning experience, this approach is computationally prohibitive due to the immense variety of potential segmentations. To address this, we adopt the classic Viterbi algorithm [40] to perform randomized tokenization of the training data. This method allows for a diverse and balanced exposure of all tokens within the vocabulary to the learning process. The Viterbi algorithm efficiently determines the most probable tokenization paths through a probabilistic model of token occurrence, which significantly reduces the computational overhead compared to exhaustive methods. By leveraging this approach, our LLM can learn each token in the vocabulary more uniformly, enhancing the overall robustness and performance of the model.

Loss momentum. In the iterative process of training Large Language Models, maintaining the stability of the vocabulary is crucial to ensure consistent learning outcomes. To achieve this, we propose a momentum-based improvement for calculating the loss during each iteration. Specifically, the loss for iteration j of token x_i, denoted as $\mathcal{L}^j(x_i)$, is not solely computed based on the current data but is also weighted by the loss from the previous iteration L^{j-1}. This approach allows for a smoother convergence and mitigates fluctuations in training dynamics. The formula for updating the loss at each iteration is given by:

$$\mathcal{L}^j_{\text{momentum}}(x_i) = \beta \mathcal{L}^{j-1}_{\text{momentum}}(x_i) + \mathcal{L}^j(x_i), \tag{5}$$

where β is the momentum coefficient that controls the extent to which the previous loss influences the current loss. This methodology not only stabilizes the vocabulary updates across iterations but also enhances the model's ability to generalize from the training data by reducing the variability in loss across successive training epochs.

4 Experiments

In this section, we outline the comprehensive experimental framework designed to assess the effectiveness of our proposed tokenizer, **Ada**ptive **T**okenizer (**ADAT**), in comparison to established methods such as Byte Pair Encoding (BPE) [32] and the Unigram model [19]. These evaluations utilize the Pythia [3] suite of models at various scales, leveraging a substantial corpus to ensure robust and generalizable results.

4.1 Experimental Setup

Model Framework We deploy the Pythia framework [3] for its lightweight design and adaptability across different computational setups. Pythia's flexibility facilitates reproducibility and consistent assessment of performance, making it an ideal choice for evaluating the scalability and efficiency of various tokenization strategies across model sizes of 70M, 160M, and 410M parameters.

Data Corpus The study utilizes a substantial corpus extracted from The Pile [14], consisting of 56GB of raw data across 91 files. We specifically excluded subsets from DM_Mathematics and Github to ensure the relevance and quality of the data. The remaining data, approximately 16 billion tokens after a random shuffle, was tokenized using a Unigram [19] tokenizer with a vocabulary size of 50,000 tokens. A detailed enumeration of the data files used is available in Supp. A.8.

Baseline Methods Our investigation compares four tokenization methods: Bytepiece [35], Byte Pair Encoding (BPE) [32], Unigram [19], and our proposed **ADAT**. These tokenizers were selected based on their established efficacy in handling large corpora and their theoretical implications for processing complex linguistic data.

Evaluation Metrics The effectiveness of each tokenization strategy is rigorously evaluated using several metrics. These include Perplexity (PPL), which measures the model's predictive accuracy, and Compression Rate(refer to A.1), assessing how efficiently the tokenization process reduces vocabulary size while preserving linguistic diversity. We calculate PPL for all models on PG19 [29] dataset. Specifically, we use its test set and the first 2048 tokens for each book. Furthermore, we use the Language Model Evaluation Harness [15] to run five-shot evaluations on eight common language

 https://doi.org/10.52202/079017-3608

Table 1: Performance Comparison of Different Tokenization Methods.

Metric	BPE	BytePiece	+ADAT(Ours)	Unigram	+ADAT(Ours)
PPL	22.31	71.5	67.19(-4.31)	16.52	**6.97(-9.55)**
ARC-C	17.32 ± 1.11	18.69 ± 1.14	18.94 ± 1.15	$\mathbf{19.54} \pm 1.16$	18.46 ± 1.12
ARC-E	37.58 ± 0.99	33.80 ± 0.97	33.71 ± 0.97	37.04 ± 0.99	$\mathbf{40.57} \pm 0.99$
Boolq	61.28 ± 0.85	42.12 ± 0.87	$\mathbf{62.20} \pm 0.85$	53.06 ± 0.87	61.19 ± 0.85
Lambda	10.89 ± 0.43	8.80 ± 0.39	13.55 ± 0.48	17.27 ± 0.53	$\mathbf{17.97} \pm 0.52$
LogiQA	23.04 ± 1.65	20.28 ± 1.58	22.27 ± 1.63	23.20 ± 1.66	$\mathbf{24.22} \pm 1.70$
PIQA	59.25 ± 1.15	57.83 ± 1.15	56.96 ± 1.16	$\mathbf{60.50} \pm 1.14$	59.93 ± 1.14
SciQ	66.60 ± 1.49	54.01 ± 1.58	51.90 ± 1.58	68.10 ± 1.47	$\mathbf{72.40} \pm 1.44$
SST-2	51.26 ± 1.69	49.08 ± 1.69	50.23 ± 1.69	49.77 ± 1.69	$\mathbf{54.24} \pm 1.69$
Winogrande	49.96 ± 1.41	50.31 ± 1.41	49.41 ± 1.41	51.46 ± 1.40	$\mathbf{51.62} \pm 1.40$
Avg. (%)	41.91	37.21	$39.91(+2.70)$	42.22	$\mathbf{44.51}(+\mathbf{2.29})$

modeling benchmarks: Lambada (OpenAI) [25], PIQA [4], WinoGrande [1], ARC-Easy [10], ARC-Challenge [10], SciQ [18], LogiQA [22], and SST-2 [33, 41], to provide a comprehensive insight into each method's capabilities.

By analyzing the impact of tokenization on model scalability and the influence of vocabulary size variations, this study aims to enhance our understanding of how tokenization strategies can optimize language models for efficiency and linguistic performance. The findings are expected to contribute significantly to the development of more robust and adaptable language processing tools, catering to a wide array of NLP applications. The Pythia models are trained using a corpus of 15B tokens, where training the 70M model consumes approximately 48 GPU hours with FlashAttention [11]. The models used for loss calculation require additional 2 GPU hours.

4.2 Tokenization Methods Evaluation

In this section, we examine the effects of different tokenization strategies on the training effectiveness. The core objective is to explore how variations in the vocabulary, induced by different tokenization methods, affect model training and performance.

The Pythia-70M model is selected due to its moderate size and efficiency, which help mitigate the complexities associated with larger model architectures. It is initialized with random weights and undergoes a single training epoch using pre-training data. This data is processed with vocabularies generated from 1/10th of the training corpus (approximately 1.5 billion tokens), each containing 50,000 tokens—a size consistent with the Pythia [3] setup.

Baseline tokenization methods including BPE, Unigram, and BytePiece, generate vocabularies consisting of 50,000 tokens directly from initial data that approximately one-tenth of the training corpus (about 1.5 billion tokens). In contrast, for the proposed ADAT method, initial vocabularies are generated using either BytePiece or Unigram with 150,000 tokens. These are then methodically refined down to 50,000 tokens over 5 iterative steps, matching the baseline vocabulary size. At each step, a randomly initialized model is trained on approximately 0.3 billion tokens from the initial dataset. Subsequently, the model performs inference on a subset of 0.1 billion tokens, during which token loss is calculated. This loss data, when combined with token frequency using the formula $\frac{a}{\lambda \log(b+1)}$, guides the vocabulary pruning process. An ablation study on the combination methods will be discussed in Section 4.6.4.

As illustrated in Table 1, our method achieves its best performance when initialized with the Unigram vocabulary, recording a score of 44.51. This score represents a considerable improvement of 2.29 points over the standard Unigram model and surpasses the BPE model by 2.6 points. Additionally, our approach shows a notable enhancement of 2.7 points when utilizing BytePiece as the initial vocabulary. Although the BytePiece vocabulary generally exhibits inferior baseline results, our method effectively elevates its performance, indicating robustness across both high-quality (Unigram) and lower-quality (BytePiece) vocabularies. These results not only affirm the efficacy of our method but also demonstrate its adaptability to different initial conditions, thereby validating its potential for broad adeptness on diverse vocab initialization.

Table 2: Evaluation on Different Scale Model Size.

Metric	70M		160M		410M	
	Unigram	ADAT	Unigram	ADAT	Unigram	ADAT
PPL	16.52	**6.97(-9.55)**	13.97	**6.19(-7.78)**	10.92	**5.78(-5.14)**
ARC-C	**19.54**±1.16	18.46 ± 1.12	18.69 ± 1.14	**18.94**±1.15	**20.82**±1.19	19.29 ± 1.21
ARC-E	37.04±0.99	**40.57**±0.99	39.52 ± 1.00	**42.87**±1.01	44.65 ± 1.02	**46.69**±1.03
Boolq	53.06 ± 0.87	**61.19**±0.85	**58.56**±0.86	57.68 ± 0.86	54.80 ± 0.87	**60.81**±0.87
Lambda	17.27 ± 0.53	**17.97**±0.52	19.06 ± 0.55	**25.02**±0.60	27.81 ± 0.62	**28.94**±0.66
LogiQA	23.20 ± 1.66	**24.22**±1.70	**25.65**±1.71	25.04 ± 1.65	23.20 ± 1.66	**24.32**±1.68
PIQA	**60.50**±1.14	59.93 ± 1.14	60.83 ± 1.14	**61.86**±1.14	63.38 ± 1.12	**64.61**±1.11
SciQ	68.10 ± 1.47	**72.40**±1.44	72.10 ± 1.42	**79.60**±1.28	80.70 ± 1.25	**83.50**±1.14
SST-2	49.77 ± 1.69	**54.24**±1.69	52.06 ± 1.69	**52.78**±1.69	50.69 ± 1.69	**54.71**±1.69
Winogrande	51.46 ± 1.40	**51.62**±1.40	49.88 ± 1.41	**50.69**±1.41	52.41 ± 1.40	**52.93**±1.41
Avg	42.22	**44.51**	44.04	**46.05**	46.50	**48.42**

4.3 Scalability

We examine the scalability of a proposed tokenization method that tailors the vocabulary to model size, unlike the static Unigram method which maintains a consistent vocabulary across various model capacities. The scalability of the tokenization methods is tested using the Pythia framework configured at three different levels of computational complexity: 70M, 160M, and 410M parameters. For each model size, our method generates an optimized vocabulary specific to that configuration, allowing us to analyze how adjustments in vocabulary affect performance as model size increases. In contrast, the Unigram method employs a uniform 50,000-word vocabulary across all sizes, serving as a baseline. We gauge performance using Perplexity (PPL) and scores from benchmark datasets designed to assess the linguistic capabilities of each model under various conditions, providing insights into the efficiency and adaptability of the tokenization methods at scale. For training larger models, the same volume of data will lead to insufficient warm-up, potentially resulting in a slight decline in the accuracy of loss computations used for determining token priority. As a result, we increase the data volume for training the loss calculation model according to the size of model.

The results of this experimental framework, as presented in Table 2, indicate substantial performance variations across different model sizes employing varied tokenization strategies. Specifically, average performance scores across all evaluated metrics demonstrate consistent improvements with increases in model sizes: ADAT achieves a score of 44.51 in the 70M model, significantly surpassing Unigram's 42.22; 46.05 compared to 44.04 in the 160M model; and 48.32 versus 46.50 in the 410M model. These findings highlight the superior efficacy of ADAT in managing diverse model volumes compared to the more static approach of Unigram, which exhibits limited scalability with increasing model size. Remarkably, the performance of the 70M model using ADAT exceeded that of the Unigram on the 160M model by nearly 5%, illustrating the substantial enhancement and ability of our method to bridge a parameter gap of over double. Furthermore, the performance of our 160M model approaches that of the 410M model, emphasizing the robust adaptability of the ADAT method across varying computational scales.

Table 3: Cross-Model Adaptability of Vocabularies.

Model Size	Unigram	70M-Model Vocabulary	410M-Model Vocabulary
70M	42.22	44.51	42.62
160M	44.04	45.03	45.83
410M	46.50	47.66	48.42

4.4 Cross-Model Adaptability

This experiment evaluates the adaptability of vocabularies generated by our proposed tokenization method across various configurations of the Pythia model, particularly assessing whether vocabularies optimized for one model size can effectively scale to others. We initially create vocabularies using the 70M and 410M configurations. These are then used to train models at both scales to evaluate performance in downstream tasks, allowing us to assess how vocabularies designed for a specific

https://doi.org/10.52202/079017-3608

Table 4: Impact of Vocabulary Size on Model Performance Across Different Model Sizes.

Model Size	Vocabulary Size	Tokenization Method	Accuracy	Perplexity (PPL)
70M	50,000	**Unigram**	42.22	16.52
		ADAT	44.51(+2.29)	6.97
	30,000	**Unigram**	40.93	32.53
		ADAT	43.33(+2.40)	7.38
160M	50,000	**Unigram**	44.04	13.97
		ADAT	46.05(+2.01)	6.19
	30,000	**Unigram**	43.08	15.21
		ADAT	45.26(+2.14)	6.11

size perform when applied to both smaller and larger models, thus examining their cross-model adaptability.

Table 3 illustrates the cross-model adaptability of vocabularies across different model sizes. By applying vocabularies derived from different model sizes to various models, we observe that vocabularies generated by the 70M and 410M models surpass the performance of the standard Unigram model. This indicates the adaptability of the ADAT vocabularies across different model sizes. Furthermore, we note that the vocabulary from the 410M model achieves only a marginal improvement of 0.4 when applied to the 70M model, significantly less than the 2.29 increase afforded by the 70M model's vocabulary. This suggests that the vocabularies selected by ADAT possess a strong capacity for targeted optimization, enabling the selection of tokenization strategies that are specifically tailored to the characteristics of different models.

4.5 Model and Vocabulary Size

The experiment aims to assess the impact of different tokenizer strategies on model performance across two vocabulary sizes, comparing a standard 50,000-token set with a reduced 30,000-token set. We utilize two configurations of the Pythia model—70M and 160M—to explore how vocabulary size influences model efficiency. Each model is tested using both the standard Unigram and our proposed tokenization method. This setup allows us to directly observe the effects of reduced vocabulary sizes on the performance dynamics, providing insights into how smaller vocabularies impact the computational efficiency and efficacy of language models.

The experimental results, as presented in Table 4, support the hypothesis that changes in vocabulary size can significantly affect model performance, with different impacts observed across varying model sizes. For large language models, it is common for models of vastly different sizes to utilize vocabularies of similar or identical sizes [3, 39]. This practice can lead to issues of performance or efficiency. Our method offers a more effective solution by tailoring tokenization strategies to the specific sizes of models, thereby mitigating these challenges. For the 70M model, ADAT achieved a notable improvement in accuracy from 42.22 to 44.51 (+2.29) and a substantial reduction in perplexity from 16.52 to 6.97 when using a 50,000-word vocabulary. Even with a reduced vocabulary of 30,000, ADAT enhances accuracy to 43.33 (+2.40) and decreases perplexity to 7.38, suggesting robustness against vocabulary size reduction. In contrast, the 160M model, which has a greater parameter capacity, also shows improvements with ADAT: accuracy increases from 44.04 to 46.05 (+2.01), and perplexity drops sharply from 13.97 to 6.19 for the 50,000 vocabulary size. With a 30,000 vocabulary, accuracy still increases to 45.26 (+2.14), and perplexity remains low at 6.11, underscoring that larger models not only handle vocabulary reductions well but also benefit significantly in terms of computational efficiency and model quality.

4.6 Ablation Study

This ablation study is structured into three distinct parts to explore how variations in the inference corpus size used for calculating token loss, initial vocabulary sizes, momentum strategy, and balance function $F(a, b)$ influence the efficacy of our proposed tokenization method on a 70M parameter model. More ablation results can be referred to in the supplementary.

Table 5: Ablation Studies Results.

Infer Data Volume		**Initial Vocabulary Size**		**Momentum**		**Balance** $F(a, b)$	
Tokens	**Acc.**	**Init Size**	**Acc.**	**Methods**	**Acc.**	**Methods**	**Acc.**
1M	43.13	75k	43.42	Unigram	42.20	$a - \lambda b$	42.70
10M	43.74	100k	43.78	ADAT+By	44.51	$log(a) - \lambda b$	43.23
100M	44.51	150k	44.51	-w/o Mnt.	43.16	$a/\lambda \log(b + 1)$	44.51

4.6.1 Corpus Size used in Loss Calculation

We conducted an experimental study to investigate the effects of varying corpus sizes on the accuracy of token loss calculations. The experiment assessed the performance of models trained on different sizes of inference data, specifically 1 million (1M), 10 million (10M), and 100 million (100M) tokens. The results are summarized in the table 5.

These results indicate a direct correlation between the volume of the corpus used during the loss calculation phase and the overall accuracy of token loss estimates. When smaller corpora are used, a significant number of tokens are absent, resulting in numerous instances where loss values cannot be computed. Furthermore, the precision of token loss estimations tends to decrease with smaller data sets. Even with just 1M tokens, there was a noticeable improvement over the baseline unigram vocabulary accuracy of 42.22. This enhancement became more pronounced with larger data volumes, reaching an increase of 44.51 in accuracy with 100M tokens.

4.6.2 Initial Vocabulary Size

This segment of our study assesses the effect of different initial vocabulary sizes on model performance. Adjusting the vocabulary from a baseline of 150,000 tokens to either 100,000 or 75,000 tokens, we explore the influence of vocabulary scale on training outcomes. The results, detailed in Table 5, illustrate the trade-offs associated with varying vocabulary sizes.

From the experiment, it is evident that models equipped with a larger initial vocabulary of 150,000 tokens tend to achieve lower Perplexity and higher Accuracy, indicating a robust ability to capture diverse linguistic nuances that significantly enhance performance. In contrast, reducing the vocabulary size to 75,000 tokens results in increased perplexity and decreased accuracy, highlighting a potential compromise in linguistic detail that adversely affects model functionality, especially in complex linguistic scenarios.

4.6.3 Momentum Strategy

This experiment evaluates the impact of incorporating a momentum strategy into our tokenization algorithm's vocabulary pruning process. The performance of vocabularies pruned under both the momentum and non-momentum conditions is directly compared in Table 5.

The results in Table 5 demonstrate that the momentum approach significantly enhances model accuracy, with a notable improvement from 43.16% to 44.51% in the ADAT method with Unigram initialization vocabulary when momentum is applied. Similarly, the Unigram method shows a baseline performance of 42.20% accuracy. These results confirm that integrating momentum allows for a more refined pruning process by effectively utilizing historical performance data to make more informed decisions, thereby preserving valuable linguistic features.

4.6.4 Balance Strategy

In this ablation study, we investigate various functions to balance token frequency and loss value in our tokenization algorithm. The primary objective is to adhere to the principle that tokens with higher frequency and lower loss should be assigned higher priority. Given the significant difference in their magnitudes, we explored subtraction and division methods. We evaluated three functions, detailed in Table 5. Here, λ is a scaling factor introduced to adjust the balance between frequency and loss, and we set it as 1 in practice.

The results demonstrate that the subtraction methods $a - \lambda b$ and $a - \lambda \log(b)$ yielded accuracies of 42.70 and 43.23 respectively. These results indicate relatively poor performance, even with

 https://doi.org/10.52202/079017-3608

the logarithmic transformation applied to the score $\log(a) - \lambda b$. This underperformance is likely attributable to the significant disparity in the magnitudes of frequency and loss values, which the subtraction methods struggle to reconcile effectively. In contrast, the division method $\frac{a}{\lambda \log(b+1)}$ significantly outperformed the subtraction approaches with an accuracy of 44.51. This superior performance suggests that the division method more naturally balances the influence of frequency and loss by scaling the loss logarithmically before the division, thereby mitigating the impact of numerical range discrepancies. This method's ability to integrate frequency and loss without requiring additional adjustments for scale disparities results in more stable and effective prioritization of tokens.

4.7 Analysis of the Compute Costs

To prove that the proposed method is feasible in practice. We analyzed the empirical runtime introduced by ADAT. To measure runtime, we used 8 NVIDIA A100 GPUs, an Intel 8378A CPU, and PyTorch 2.1.2 with CUDA 12.1. The tokenizer optimization involves 5 epochs, where each epoch consists of training the LLM on a 0.3B corpus, followed by inference on a 0.1B corpus, and concludes with a vocabulary pruning step (90 seconds for a 100K tokens vocabulary). Therefore, the total computational cost of the tokenizer optimization process is calculated as:

$$5 \times (0.3B \text{ training} + 0.1B \text{ inference} + \text{ pruning time}) = 1.5B \text{ training} + 0.5B \text{ inference} + 450s.$$

ADAT introduces an additional training cost of 1.5B tokens and an inference cost of 0.5B tokens, along with minimal vocabulary pruning time. Compared to the hundreds of billions or even trillions of tokens required for LLM training, these computational costs are negligible. As shown in Table 6, the full-scale training of the LLM incurs significantly higher computational costs. For instance, when training models with a 16B and 60B corpus, the tokenizer optimization accounts for only 4.17% and 1.04% of the total training time, respectively. The Pythia-70M model takes 510 GPU hours to train with the full Pile dataset [3], and exceeds the tokenizer optimization's computational cost by over 255 times. Therefore, the additional computational cost introduced by our method is minimal, making it feasible in practice.

Table 6: Runtime of ADAT optimization and training models.

	Tokenizer Optimization	**Training on 16B**	**Training on 60B**	**Pythia Report**
Runtime	2 GPU hours	48 GPU hours	192 GPU hours	510 GPU hours

5 Limitations

The adaptive nature of our proposed tokenizer method introduces variations in tokenizers across different model sizes, leading to inconsistent vocabularies. This inconsistency complicates tasks such as knowledge distillation and speculative decoding, which rely on the assumption of a uniform vocabulary across both smaller and larger models.

6 Conclusion

In this paper, we have presented a novel approach to tokenizer design that integrates key aspects of both accuracy and inference speed, addressing the inherent limitations found in existing tokenizer schemes. By innovating beyond the traditional frequency-based and end-to-end methodologies, our adaptive tokenizer framework strategically optimizes vocabulary construction, ensuring both rapid processing and high precision in language modeling tasks. Our results demonstrate that the adaptive tokenizer significantly enhances the performance of Large Language Models (LLMs) across various benchmarks, providing a balanced solution that does not sacrifice speed for accuracy or vice versa. Future work will focus on refining these adaptive tokenization techniques, exploring further integration with neural network architectures, and expanding their applicability to a broader range of languages and complex linguistic tasks.

Acknowledgements

This work was supported by the Australian Research Council under Projects DP240101848 and FT230100549.

References

[1] Winogrande: An adversarial winograd schema challenge at scale. 2019.

[2] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[3] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[4] Y. Bisk, R. Zellers, R. L. Bras, J. Gao, and Y. Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[5] B. Boecking, N. Usuyama, S. Bannur, D. C. Castro, A. Schwaighofer, S. Hyland, M. Wetscherek, T. Naumann, A. Nori, J. Alvarez-Valle, et al. Making the most of text semantics to improve biomedical vision–language processing. In *European conference on computer vision*, pages 1–21. Springer, 2022.

[6] K. Bostrom and G. Durrett. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*, 2020.

[7] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] E. Bursztein, M. Zhang, O. Vallis, X. Jia, and A. Kurakin. Retvec: Resilient and efficient text vectorizer. *Advances in Neural Information Processing Systems*, 36, 2023.

[9] J. H. Clark, D. Garrette, I. Turc, and J. Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

[10] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

[11] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] P. Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

[14] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser, and C. Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[15] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou. A framework for few-shot language model evaluation, 12 2023.

[16] N. Godey, R. Castagné, É. de la Clergerie, and B. Sagot. Manta: Efficient gradient-based tokenization for robust end-to-end language modeling. *arXiv preprint arXiv:2212.07284*, 2022.

[17] M. M. Islam, G. Aguilar, P. Ponnusamy, C. S. Mathialagan, C. Ma, and C. Guo. A vocabulary-free multilingual neural tokenizer for end-to-end task learning. *arXiv preprint arXiv:2204.10815*, 2022.

[18] M. G. Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions. 2017.

[19] T. Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*, 2018.

 https://doi.org/10.52202/079017-3608

[20] T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[21] W. Ling, I. Trancoso, C. Dyer, and A. W. Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

[22] J. Liu, L. Cui, H. Liu, D. Huang, Y. Wang, and Y. Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

[23] S. Liu, N. Deng, S. Sabour, Y. Jia, M. Huang, and R. Mihalcea. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15264–15281, 2023.

[24] P. Nawrot, J. Chorowski, A. Łańcucki, and E. M. Ponti. Efficient transformers with dynamic token pooling. *arXiv preprint arXiv:2211.09761*, 2022.

[25] D. Paperno, G. Kruszewski, A. Lazaridou, Q. N. Pham, R. Bernardi, S. Pezzelle, M. Baroni, G. Boleda, and R. Fernández. The lambada dataset, Aug 2016.

[26] C. Park, S. Eo, H. Moon, and H.-S. Lim. Should we find another model?: Improving neural machine translation performance with one-piece tokenization method without model modification. In *Proceedings of the 2021 conference of the north american chapter of the association for computational linguistics: human language technologies: Industry papers*, pages 97–104, 2021.

[27] A. Radford, R. Jozefowicz, and I. Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

[28] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[29] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019.

[30] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[31] V. Sachidananda, J. S. Kessler, and Y.-A. Lai. Efficient domain adaptation of language models via adaptive tokenization. *arXiv preprint arXiv:2109.07460*, 2021.

[32] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[33] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[34] M. N. Sreedhar, X. Wan, Y. Cheng, and J. Hu. Local byte fusion for neural machine translation. *arXiv preprint arXiv:2205.11490*, 2022.

[35] J. Su. Bytepiece: A more pure and effective tokenizer. `https://github.com/bojone/bytepiece`, 2023.

[36] I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1017–1024, 2011.

[37] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu, and D. Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. *arXiv preprint arXiv:2106.12672*, 2021.

[38] A. Thawani, S. Ghanekar, X. Zhu, and J. Pujara. Learn your tokens: Word-pooled tokenization for language modeling. *arXiv preprint arXiv:2310.11628*, 2023.

[39] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[40] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.

[41] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In T. Linzen, G. Chrupała, and A. Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[42] X. Wang, S. Ruder, and G. Neubig. Multi-view subword regularization. *arXiv preprint arXiv:2103.08490*, 2021.

[43] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

 https://doi.org/10.52202/079017-3608

A Appendix

A.1 Compression Rate results.

The compression rate of the proposed model and baselines across Pythia model[2]-70M models are shown in Table 7.

Table 7: Performance comparison of different tokenization methods.

Metric	BPE	BytePiece	ADAT+By	Unigram	ADAT+U
Compression Rate	4.38	4.81	4.98	3.96	2.91

A.2 Results of ADAT in the 1B model.

To further demonstrate the scalability of our proposed ADAT method, we expanded our experimental results on larger models and larger corpus. Specifically, we add results(shown in Table 8) from training a 1B model on 60B corpus. The results demonstrate that on larger models and with more data, ADAT continues to show substantial improvements over the baseline, indicating that ADAT has strong scalability.

Table 8: Results in the 1B model.

Metric	Unigram	ADAT
Avg	49.11	51.20

A.3 Analysis of Differences Between Vocabularies

To illustrate the differences between the vocabulary results obtained by ADAT and those from unigram, we calculated the overlap ratio of the two token sets as follows:

$$Ratio = |A_{vocab} \cap B_{vocab}|/|A_{vocab} \cup B_{vocab}|. \tag{6}$$

Where $\cap$ and $\cup$ denote the intersection and union of two sets, respectively, we present the overlap ratios between the tokenizers obtained using ADAT and Unigram on different models under the same vocabulary size setting, as well as the overlap ratios between these tokenizers themselves in Table 9. As shown in Table 1, there are significant differences between the vocabulary generated by ADAT and that generated by unigram. This disparity arises because ADAT is not entirely data-driven in its vocabulary generation process. By incorporating the loss from the LLM, ADAT can simultaneously focus on enhancing the performance of the LLM. Additionally,the overlap ratio between ADAT-160M and ADAT-410M is higher than that between ADAT-160M and ADAT-70M. This also indirectly explains why, as shown in Table 3, the tokenizer generated by ADAT-410M is more suitable for Pythia-160M compared to the tokenizer generated by ADAT-70M.

Table 9: Comparison between Vocabulary of ADAT and Unigram.

Unigram vs.		ADAT	
Model size	**Ratio**	**Model size**	**Ratio**
ADAT-70M	0.18	70M | 160M	0.71
ADAT-160M	0.19	70M | 410M	0.69
ADAT-410M	0.21	160M | 410M	0.84

To compare the vocabulary obtained by ADAT with the initial vocabulary, we sorted the 100k initial vocabulary by score in descending order and calculated the percentage of tokens, pruned by ADAT,

[2]with Apache-2.0 license

that fall into each score interval. As shown in Table 10, ADAT-70M-50K tokens are most densely distributed not in the 0-25% interval, indicating that ADAT relies not only on token frequency but also on the prediction difficulty of tokens in the LLM during training.

Table 10: Comparison with Initial Vocabulary with vocabulary size 100k.

Vocabulary	**0-25%**	**25-50%**	**50-75%**	**75-100%**
Unigram-50k	54.75%	23.95%	8.11%	13.19%
ADAT-70M-50k	21.43%	31.60%	32.72%	14.24%

A.4 Impact of Training Epochs

We investigate the effects of varying the number of training epochs for developing the vocabulary. Specifically, the model is trained using vocabularies that have been developed over 3, 5, and 7 epochs. The model is initialized with random weights for each training session to evaluate the immediate impact of the epoch variation.

As shown in Table 11, the results from varying the number of training epochs suggest a clear relationship between training duration and vocabulary efficacy. Before the epoch reaches a certain value (5 epochs), increasing the number of epochs benefits accuracy, indicating a more refined and effective vocabulary. However, accuracy does not significantly improve with further increases in the number of epochs. This suggests that, given the specified vocabulary size, ADAT can quickly and efficiently learn an appropriate vocabulary without the need for prolonged training over many epochs. Therefore, in our experiments, the default number of training epochs is set to 5.

Table 11: Impact of Training Epochs on Model Performance.

Training Epochs	**Accuracy**
3	43.67
5	44.51
7	44.58

A.5 Expanded Analysis on Infer Data Volume Tokens and Initial Vocabulary Size

we have expanded the analysis of 'infer data volume tokens' and 'initial vocabulary size'—both variables are explored within and beyond settings in Table 5. The expanded results are displayed in the table 12. We observe that an inference data volume of 100M tokens is sufficient, with larger volumes yielding only marginal improvements. Regarding the initial vocabulary size, increasing it to 150K is important to enhance performance. However, when it increases to 200K, the score shows almost no improvement, indicating that the 150K vocabulary likely already includes most of the potential final candidate tokens. Therefore, further increasing the initial vocabulary size will not bring additional benefits.

Table 12: Results for different Infer Data Volume Tokens and Initial Vocabulary Sizes.

Infer Data Volume Tokens	75K	100K	150K	200K
1M	42.89	43.07	43.13	43.20
10M	43.19	43.39	43.74	43.77
100M	43.42	43.78	44.51	44.56
1000M	43.45	43.83	44.53	44.57

A.6 Details of Model Parameters

The detailed parameters of the 3 different model sizes applied in our experiment are shown in Table 13.

https://doi.org/10.52202/079017-3608

Table 13: Specifications of different LLMs used in the paper.

Model Size	Layers	Model Dim	Heads	Learning Rate	Batch Size
70M	6	512	8	10.0×10^{-4}	32
160M	12	768	12	2.5×10^{-4}	16
410M	24	1024	16	2.5×10^{-4}	16

A.7 Societal Impact

The adaptive tokenizer we propose is an important component of large language models. The proposed tokenizer is fine-tuned by closely monitoring the model's perplexity, enabling the language model to perform well on various tasks, such as machine translation and question answering. However, it may also face the same issues as existing subword tokenizers [19, 43], such as privacy leakage. For instance, the tokenizer might segment sensitive information, like names, addresses, or identity identifiers, into results of tokens that could be recognized in text generation. Therefore, we recommend that the training corpus undergo preprocessing to remove private information. In addition, in real-world applications, the tokenizer should be employed in conjunction with privacy-preserving technologies and specially configured filtering rules. In specific scenarios, such as medical diagnostics and legal consultations, human experts should be involved to review the tokenization results.

A.8 List of Training Dataset

The specific corpus used for training tokenizers is from The Pile[3], and the detailed list of files is shown below.

- pile_ArXiv_025.json
- pile_ArXiv_069.json
- pile_ArXiv_070.json
- pile_ArXiv_092.json
- pile_ArXiv_098.json
- pile_ArXiv_123.json
- pile_ArXiv_124.json
- pile_ArXiv_133.json
- pile_ArXiv_134.json
- pile_ArXiv_157.json
- pile_Books3_015.json
- pile_Books3_016.json
- pile_Books3_052.json
- pile_Books3_057.json
- pile_Books3_071.json
- pile_Books3_083.json
- pile_Books3_084.json
- pile_Books3_093.json
- pile_Books3_115.json
- pile_Books3_134.json
- pile_Books3_173.json
- pile_Books3_197.json
- pile_Books3_203.json
- pile_Books3_235.json
- pile_Books3_242.json
- pile_Books3_247.json
- pile_Enron_Emails_004.json
- pile_FreeLaw_031.json
- pile_FreeLaw_083.json
- pile_FreeLaw_104.json
- pile_Gutenberg_PG-19_044.json

[3]URL:https://pile.eleuther.ai/

- pile_Gutenberg_PG-19_049.json
- pile_OpenSubtitles_008.json
- pile_OpenSubtitles_031.json
- pile_OpenSubtitles_037.json
- pile_OpenWebText2_011.json
- pile_OpenWebText2_050.json
- pile_OpenWebText2_063.json
- pile_OpenWebText2_108.json
- pile_OpenWebText2_118.json
- pile_OpenWebText2_132.json
- pile_OpenWebText2_157.json
- pile_OpenWebText2_162.json
- pile_OpenWebText2_212.json
- pile_OpenWebText2_216.json
- pile_OpenWebText2_242.json
- pile_OpenWebText2_245.json
- pile_OpenWebText2_256.json
- pile_Pile-CC_001.json
- pile_Pile-CC_024.json
- pile_Pile-CC_069.json
- pile_Pile-CC_076.json
- pile_Pile-CC_106.json
- pile_Pile-CC_120.json
- pile_Pile-CC_133.json
- pile_Pile-CC_181.json
- pile_Pile-CC_209.json
- pile_Pile-CC_211.json
- pile_Pile-CC_237.json
- pile_Pile-CC_254.json
- pile_Pile-CC_259.json
- pile_PubMed_Abstracts_037.json
- pile_PubMed_Abstracts_049.json
- pile_PubMed_Abstracts_054.json
- pile_PubMed_Central_028.json
- pile_PubMed_Central_053.json
- pile_PubMed_Central_067.json
- pile_PubMed_Central_069.json
- pile_PubMed_Central_085.json
- pile_PubMed_Central_123.json
- pile_PubMed_Central_125.json
- pile_PubMed_Central_132.json
- pile_PubMed_Central_149.json
- pile_PubMed_Central_165.json
- pile_PubMed_Central_173.json
- pile_PubMed_Central_215.json
- pile_PubMed_Central_220.json
- pile_Stack_Exchange_055.json
- pile_USPTO_Backgrounds_012.json
- pile_USPTO_Backgrounds_027.json
- pile_USPTO_Backgrounds_031.json
- pile_USPTO_Backgrounds_051.json
- pile_Ubuntu_IRC_001.json
- pile_Ubuntu_IRC_017.json
- pile_Ubuntu_IRC_021.json
- pile_Wikipedia_en_006.json
- pile_Wikipedia_en_009.json
- pile_Wikipedia_en_043.json
- pile_Wikipedia_en_053.json
- pile_Wikipedia_en_070.json

https://doi.org/10.52202/079017-3608

- pile_YoutubeSubtitles_008.json

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in the Limitations section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

https://doi.org/10.52202/079017-3608

Justification: No theoretical results are included in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the information needed to reproduce the experimental results is included in the Experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The list of the dataset is included in the appendix. We will release codes after completing the necessary preparations.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details are included in the Experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the experiment results with std error.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/079017-3608

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The sufficient information on the computer resources is included in Experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms in every respect with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Both potential positive societal impacts and negative societal impacts are discussed in Societal Impact section of the Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Included all citations.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

https://doi.org/10.52202/079017-3608

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.