A Comparison of Tokenization Impact in Attention Based and State Space Genomic Language Models

LeAnn M. Lindsey[1]✉, Nicole L. Pershing[2], Anisa Habib[1], W. Zac Stephens[3], Anne J. Blaschke[2], and Hari Sundar[4]

[1]Kahlert School of Computing, University of Utah, SLC, UT, USA
[2]Department of Pediatrics, School of Medicine, University of Utah, SLC, UT, USA
[3]Department of Pathology, School of Medicine, University of Utah, SLC, UT, USA
[4]Department of Computer Science, Tufts University, Boston, MA, USA

Genomic language models have recently emerged as powerful tools to decode and interpret genetic sequences. Existing genomic language models have utilized various tokenization methods including character tokenization, overlapping and non-overlapping k-mer tokenization, and byte-pair encoding, a method widely used in natural language models. Genomic models have significant differences from natural language and protein language models because of their low character variability, complex and overlapping features, and inconsistent directionality. These differences make sub-word tokenization in genomic language models significantly different from traditional language models.

This study explores the impact of tokenization in attention-based and state-space genomic language models by evaluating their downstream performance on various fine-tuning tasks. We propose new definitions for *fertility*, the token per word ratio, in the context of genomic language models, and introduce *tokenization parity*, which measures how consistently a tokenizer parses homologous sequences. We also perform an ablation study on the state-space model, Mamba, to evaluate the impact of character-based tokenization compared to byte-pair encoding. Our results indicate that the choice of tokenizer significantly impacts model performance and that when experiments control for input sequence length, character tokenization is the best choice in state-space models for all evaluated task categories except epigenetic mark prediction.

genomics | genomic language models | tokenization

Correspondence: ***leann.lindsey@utah.edu***

Introduction

Transformer-based large language models (LLMs) have had tremendous success in the field of natural language processing (NLP), driving advances in contextual understanding, content generation, and language translation. There are many similarities between natural language and genetic sequences, and the success of BERT (1) and GPT-2 (2) on NLP benchmarks led to transformer-based models being used as a new method to decipher the language of amino acid and nucleotide sequences. Language models trained on amino acid sequences were adopted quickly into the research community and have been used to accurately predict protein structure (3), predict effects of mutations on protein function (4), and generate functional protein sequences (5). Protein language models, however, are limited in their contextual understanding to the protein-coding regions of the genome.

Genomic language models (gLMs), i.e., language models trained on DNA, contain all of the information to decode both coding and non-coding regions (6), but have not yet surpassed existing methods on tasks such as identifying regulatory elements in the genome. Proponents argue that gLM embeddings can learn functional genomics annotations strictly from contextual information (7) and can capture complex dependencies in DNA sequences (8–10), which has enabled gLMs to even surpass protein models on select protein benchmarks (6). Others argue that the representational power of gLMs is limited, they perform only slightly better than traditional methods that use one-hot encoding, and fall short of true "foundational model" status(11, 12).

Genomic language tasks are more challenging than protein language tasks because of the inherent differences between DNA sequences and amino acid sequences. DNA has much lower character variability than amino acid sequences and contains different layers of information, such as overlapping regulatory features that can be context-dependent.

To unlock the potential that gLMs contain, significant research is needed to systematically evaluate the differences between gLMs, pLMs, and LLMs. In this study, we evaluate one variable, tokenization, to understand how current tokenization methods impact genomic language tasks.

Tokenization is the first step in the language model pipeline and is used to split an input sequence into tokens. A token represents either words, sub-words, or characters that are assigned numeric values and used as inputs to a neural network in order to learn context-specific embeddings. Sub-word tokenization methods such as byte-pair encoding (BPE) have become a de-facto standard in natural language models (13, 14), and significant research has been done to analyze and evaluate tokenizers in that field (15–20). Tokenization of genomic sequences, however, presents a unique challenge because no obvious "word" demarcations exist. It cannot be assumed that principles about tokenization that have been confirmed in natural languages will necessarily be true for genomic language models. Tokenizers may work as expected, or they may have limitations or unexpected consequences when used with genomic input.

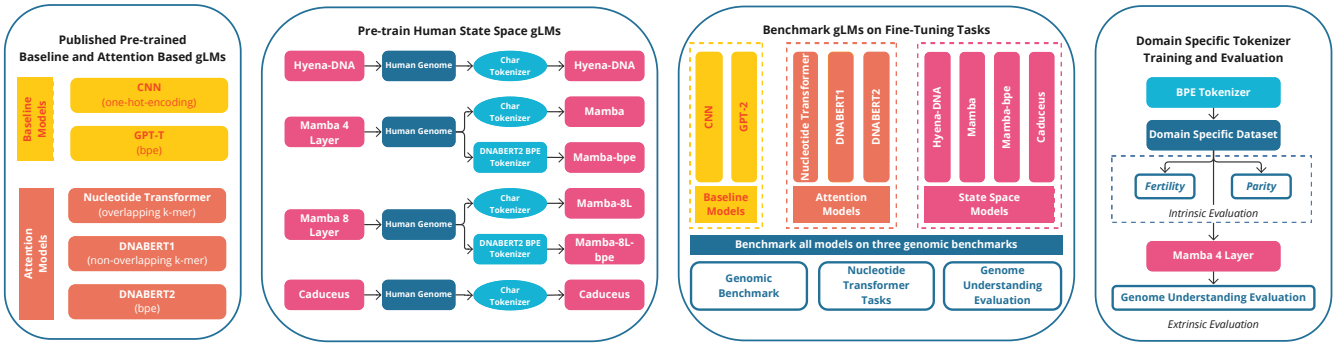

Fig. 1. Overview of experiments to compare the impact of tokenization in attention based and state space gLMs. Two baseline models, three pre-trained attention based models and three state space models were compared against three published genomic benchmarks comprising 45 downstream genomic tasks. Two sizes (4-Layer and 8-Layer) of the Mamba state-space models were trained using both BPE and character tokenization leaving all other model parameters fixed. Three additional BPE tokenizers were trained using domain-specific data from the yeast, mouse, and bacterial genomes and evaluated using the intrinsic metrics of *fertility* and *parity*, and the external metrics of downstream performance on the GUE benchmark.

Though the ideal tokenization method is still an open question in the field of natural language processing, there has been a significant research effort to understand the consequences of different tokenization choices. Systematic study of tokenization in the domain of gLMs remains limited.

In this study, we investigate the following questions:

- Is sub-word tokenization in gLMs simply a form of compression, or does tokenization assist the model in capturing meaningful contextual relationships between tokens?
- Does the choice of tokenizer impact model performance?
- Are "words" (tokens learned by a BPE tokenizer) a meaningful unit of information in DNA?
- How do domain differences in training data impact tokenization in gLMs?

In order to investigate these questions, we compared three attention-based gLMs, three state space gLMs, and two baseline models, a simple three-layer Convolutional Neural Network (CNN) and the standard pre-trained GPT-2 model, on forty-five downstream tasks compiled from published genomic benchmarks (8, 21, 22). We also trained a 4-layer and an 8-layer Mamba state space model using both byte-pair encoding and character tokenization to gauge the direct impact of tokenization on model performance. We train three domain-specific tokenizers and compare the overlapping tokens in the tokenized vocabularies. Finally, we propose a new definition for the tokenization metric of *fertility*, the token per word ratio, in the context of gLMs, and introduce the concept of *tokenization parity* to measure how consistently a tokenizer parses homologous sequences.

The results of our study indicate that when sequence length is controlled for, the tokenization method chosen significantly impacts model performance, particularly on tasks that are currently more challenging for gLMs. We demonstrate that tokenizers trained on domain-specific data have moderate overlap in their vocabularies, with 25% of tokens shared between yeast, mouse, and human, but with significant variability. Our study also illustrates that for most tasks, character tokenization is the better choice when using state space models, where the context window is not as limited. However, when predicting epigenetic marks, byte-pair encoding performs better than character-based tokenization in state space models.

Additional contributions from this work include the release of pre-trained domain-specific BPE tokenizers, and pre-training datasets for bacteria, yeast, and mouse.

Related Work

Tokenizers. Existing genomic language models have been trained using three main tokenization methods: *character based tokenization*, where the sequences are tokenized by individual nucleotide, *k-mer tokenization*, where the input is tokenized into either overlapping or non-overlapping substrings of length k, and sub-word tokenization using *byte-pair encoding* (see Table 1).

Byte pair encoding (BPE) was originally developed as a method of compression and was later widely adopted by the NLP community as a method of tokenization for language models. BPE calculates the frequency of adjacent nucleotides and uses this to construct a vocabulary of the most frequently seen sub-words in a training corpus, which in natural language helps the model cope with out-of-vocabulary words.

BPE also provides significant advantages through text compression. The DNABERT2 tokenizer provides data compression of approximately 4-5x, significantly expanding the length of input sequences that can be used as input into the fixed context window. The non-overlapping k-mer tokenization used by the Nucleotide Transformer model also provides this same compression advantage.

Significant research has investigated the impact of tokenization on natural language models (17, 23, 24). We will review only work relevant to gLMs here.

Character-based tokenization in natural language models. Several natural language models have explored the idea of

Table 1. Tokenization of a sample input with nucleotide, k-mer and byte-pair encoding tokenization.

Tokenization Method	Tokens	Length	Compression
input sequence	GCTGGGTGCCTCTCGGGTGGGC	22	1.0
character based	G C T G G G T G C C T C T C G G G T G G G C	22	1.0
overlapping k-mer	GCTGGG CTGGGT TGGGCT GGGTGC GGTGCC GTGCCT TGCCTC GCCTCT ...	17	1.3
non-overlapping k-mer	GCTGGG TGCCTC TCGGGT GGGC	4	5.5
byte-pair encoding	GC TGGG TGCC TCTC GGGTG GGC	6	3.7

tokenization by byte or character. ByT5 (25), a variation of the T5 model (26), was able to obtain performance similar to a sub-word tokenized model by increasing the number of layers in the decoder stack by three times the number in the encoder stack. They also observed that byte tokenization is more resistant to noise and performs well on tasks sensitive to spelling. Similarly Clark et al. (27) used downsampling and a deep encoder stack in a character tokenized model, CANINE, to compensate for the lack of compression.

Several studies conclude that in the field of natural language, sub-word tokenization is more than simply compression (15, 17, 28) However, several recent articles have illustrated some of the potential hazards of sub-word tokenization (24, 29, 30). Kaiser et al. (18) studied the issue of tokenization inconsistency and demonstrated that models with consistent tokenization perform better on both in-domain and out-of-domain tasks.

Tokenization in biological language models. To date, the tokenization studies in the field of gLMs have been specific to one model and provided only summary results. The authors of DNABERT2 (8) compared BPE with k-mer tokenization over all GUE tasks and concluded that BPE resulted in significantly higher performance. They did not, however, control for input segment length. The authors of HyenaDNA (10) compared BPE with k-mer tokenization and concluded that using k-mer tokenization with state space models degraded the results. Additionally, Schiff et al. (9) observed that with k-mer tokenization small changes in the input sequence can result in dramatic changes in tokenization (9).

Dotan et al. (31), analyzed the effects of tokenization on biological language models and concluded that applying subword tokenization to biological sequences can both increase accuracy and decrease computational training time. However, their experiments focused on tokenization in protein language models, with only one experiment out of eight being performed on DNA. Additionally, they did not control for input sequence length. They used the same size context window in both experiments, thus, because of the compression factor of sub-word tokenization techniques, the experiments with sub-word tokenization contained significantly more information than those with character-based tokenization, which is known to significantly improve performance (32).

Token vocabulary impact in multilingual models. Multi-species gLMs are similar to multilingual models in that the genomes of species have different distributions of "words" in their tokenized vocabularies. Vulic et al (16) investigated the performance gap in models that had been trained by a multilingual tokenizer vs. a monolingual tokenizer. They found that languages that are adequately represented in the multilingual vocabulary show only a small performance difference but that replacing the multilingual tokenizer with a mono-language tokenizer, improved the results of the model on nearly every task.

Several studies indicate that the vocabulary size and training data distribution directly impact downstream model performance. The experiments in Galle et al. (33) show that given a fixed-size vocabulary, translation accuracy is correlated with lower fertility, the token per word ratio. Similarly, Kharitonov et al. (34) link a larger vocabulary size with lower fertility, which they conjecture allows a model to memorize information more easily.

There have not yet been similar experiments investigating the impact of fertility, optimal vocabulary size and the impact of domain-specific vocabularies on gLMs or between the attention and state-space model architectures.

Genomic Language Models. In the last four years, there has been rapid advancement in the field of genomic language models. DNABERT (35) was the first published transformer based gLM. It used the original BERT-base architecture and trained the human reference genome using overlapping k-mer tokenization with $k = [4, 5, 6]$.

The Nucleotide Transformer (NT), developed by InstaDeep and Nvidia (22), was a suite of models, with released models ranging from 500M to 2B parameters, and trained on a more diverse dataset. This included a model trained on over 3000 different human genomes and another model trained on 850 species. The NT model uses non-overlapping k-mer tokenization (see Table 1).

Dnabert2 (8) used Bert-base architectures, but added optimizations like Flash Attention (36) and low-rank adaptation of language models (LORA) (37) to help the model scale and allow longer input sequences. It was trained on a multi-species dataset that included the human genome and the genomes of 135 other species. The tokenization method was changed from k-mer to byte-pair encoding (BPE). The authors observed that overlapping k-mer tokenization resulted in information leakage which limited the model's ability to generalize. DNABERT2 outperformed DNABERT1 on most benchmarks, but since the model underwent several changes, including a significantly different pre-training dataset, it is not clear how much of the impact on model performance was due to the tokenization change.

State Space models like S4 (38), Hyena (39) and Mamba (40) were developed to target tasks that required an understanding of long-distance sequence dependencies. These models replace the self-attention block in transformers with a new block that allows the model to reduce the amount of contextual information saved. They were inspired by early

Table 2. Details of all models used in the benchmarking experiments. A simple 3-layer CNN was used as a baseline model. The OpenAI/GPT-2 pre-trained model was used as a second baseline. The Time and Hardware columns specify the hardware used and total time needed for pretraining.

Model	Architecture	Parameters	Tokenization	Max Context Window (nt)	Pre-training Data	Pre-training Time	Pre-training Hardware
CNN	CNN	464 K	one hot encoding	N/A	N/A	N/A	N/A
GPT-2	Attention	124 M	BPE	1024	WebText	4 days	256 Tesla P100
Nucleotide Transformer	Attention	500 M	non-overlap k-mer	512	3202 Genetically Diverse Human Genomes	14 days	16x 8 Nvidia A100
DNABERT1	Attention	117 M	overlap k-mer	512	Hg38 Human Reference Genome	25 days	8 Nvidia 2080 Ti
DNABERT2	Attention	117 M	BPE	∼2500	Human Genome and 135 Other Species	14 days	8 Nvidia 2080 Ti
HyenaDNA	Space State	13.1 M	char	1M	Hg38 Human Reference Genome	2-6 hrs	1 Nvidia A100
Mamba	Space State	1.8 M	char	131K	Hg38 Human Reference Genome	4-12 hrs	4 Nvidia A100
Caduceus	Space State Equivariant	3.9 M	char	131K	Hg38 Human Reference Genome	4-12 hrs	4 Nvidia A100

mathematical models called state space models, but have mathematical similarites to both convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Unlike attention-based models that have quadratic time complexity, Hyena has subquadratic time complexity (39), and the selective state model Mamba has linear time complexity (40). This significant reduction in time complexity allows the model to significantly expand the context window, and reduces overall training time. On standard NLP benchmarks, Mamba matches the performance of attention-based models that are significantly larger (40). The natural language version of Mamba was trained using BPE tokenization, but the genomic language model Mamba was trained using character-based nucleotide tokenization (9) (see Table 1).

Caduceus (9) is a recently released model that uses a unique model architecture. It trains both the forward and reverse complement sequence input on parallel Mamba layers and combines the results to create an equivariant model. Model differences are summarized in Table 2. Although rapid advancements have been made in gLM, the characteristics of optimal training datasets, the impact of different tokenization methods, and the strengths and weaknesses of different models remain unclear. This understanding is critical to optimize the choice of tokenization and architecture for different genomic tasks and is a primary contribution of this work.

Materials & Methods

gLM Models. We tested two baseline models, a simple three-layer CNN (21) trained using one-hot encoding and the pre-trained GPT-2 model from OpenAI. Although it was not trained on DNA, it performed significantly better than random so we include it as a second baseline comparison. GPT-2 uses BPE tokenization, and no changes were made to the model or tokenizer to adapt them to DNA for these experiments.

We compared the transformer-based models, Nucleotide Transformer (22), DNABERT1 (35), and DNABERT2 (8), against three of the state space genomic language models, HyenaDNA (10), Mamba (40), and Caduceus (9). To directly compare the effect of a tokenizer on a state space model, we created a four-layer and eight-layer Mamba model and trained both using character and BPE tokenization.

Data. All human datasets used were previously published in the HyenaDNA, Mamba-char, Mamba-BPE and Caduceus models (41).

In order to test whether domain-specific tokenizers could improve downstream model performance, similar to the performance improvement seen in Galle et al. (33) with multi-lingual vs. mono-lingual tokenizers, we created additional domain-specific datasets for mouse, yeast, and bacteria. (See Supplementary Material for details).

Benchmarks. We benchmarked all models against three published genomic fine-tuning benchmarks: the Nucleotide Transformer Tasks (22), the Genomic Benchmark (21), and the Genome Understanding Evaluation (8). All fine-tuning tasks were performed ten times, and the average accuracy and Mathews Correlation Coefficient (MCC) were reported. A small hyperparameter search was performed for the state space models, since they are extremely sensitive to these parameters. Details are available in the Supplementary Material. Differences in these benchmarking results and published results we attribute to the gLMs high variability and dependence on initial seed values.

Downstream Performance Evaluation. We report the results of each model's performance using accuracy and the Matthews correlation coefficient.

Tokenizers. Our experiments on tokenization were implemented using the `transformers` library from *HuggingFace* (42). We limited this study to token vocabularies of size 4096 because of the experiments by Zhou et al. (8) that recommended tokenized vocabularies of this size.

Intrinsic Evaluation Metrics. Tokenizers are commonly evaluated on both intrinsic and extrinsic metrics. Intrinsic metrics are those that are independent of downstream model performance and are either characteristics of the tokenizer or generated tokens. Extrinsic metrics are performance metrics of a model on a specific task.

Table 3. A human transcription factor sequence from the GUE dataset tokenized by five different BPE tokenizers that have been trained on different datasets. The fertility, F, can be understood to represent the number of words per ten nucleotides in a DNA sequence. A lower fertility provides more compression and generally indicates that a vocabulary is a better fit for a corpus.

Tokenizer	F	Tokenized Sequence
DNABERT2	1.68	`GCTG GTCTCGAA CTCC TGACC TCAAGTGA TCC GCCTG CCTT GGCCTCCCAAA GTGCTGGGATTACAGG ... CGGCC GGGGA TCTGA GT`
Hg38	1.09	`GC TGG TCTCGAACTCCTGACCTC AAGTGATCC GCCTGCC TTGGCCTCCCAAAGTGCTGGGATTACAGGC ... GCCCGGCC GGGG ATCTGAG T`
Mouse	2.38	`GC TGG TCTCG AACTCC TGACC TCAAG TGATCC GCC TGCC TTGGCC TCCC AAAG ... ACTGC GCCC GGCC GGGG ATC TGAG T`
Yeast	2.48	`GCTGG TCTC GAAC TCC TGACC TCAAG TGATCC GCC TGCC TTGGCC TCCC AAAG .... ACTGC GCCC GGCC GGGG ATC TGAG T`
Bacteria	2.18	`GCTGG TCTC GAAC TCC TGACC TCAA GTGA TCCGCC TGCCTT GGCC TCCCAAA GTGC TGGGA ... TGCGCCC GGCC GGGGA TCTGA GT`
GPT-2	5.25	`G CT GG T CT C GA ACT CC TG AC CT C AA GT G AT CC G CC TG CC .... CC ACT GC G CCC GG CC GGGG AT CT G AG T`

Table 4. Overview of MCC scores across models summarized by task category and benchmark. The highest performing model in each model type is highlighted in bold. The highest overall score is underlined. The benchmarks included are: Genomic Benchmark (GB) (21), Nucleotide Transformer Tasks (NTT) (22), GUE (Genome Understanding Evaluation) (8)

	CNN	GPT-2	NT	DNABERT1	DNABERT2	HyenaDNA	Mamba-char	Mamba-bpe	Caduceus
EMP	0.707	0.658	0.740	0.747	**0.771**	0.769	0.738	0.783	**0.806**
Enhancers	0.695	0.656	**0.792**	0.740	0.750	0.705	0.743	0.742	**0.763**
Promoters	0.885	0.822	**0.903**	0.899	0.894	0.881	0.872	0.841	**0.888**
Splice Sites	0.940	0.583	0.951	**0.958**	0.942	0.936	0.835	0.659	**0.944**
TF	0.746	0.695	0.761	0.806	**0.838**	0.827	0.830	0.771	**0.852**
GB	0.806	0.692	**0.829**	0.754	0.773	0.831	0.855	0.845	**0.856**
NTT	0.781	0.694	0.827	0.828	**0.838**	0.819	0.797	0.782	**0.842**
GUE	0.795	0.728	0.812	0.840	**0.853**	0.834	0.813	0.762	**0.854**
All Benchmarks	0.791	0.707	0.822	0.819	**0.832**	0.827	0.814	0.786	**0.849**
	Baseline		*Attention Models*			*State Space Models*			

Fertility. Fertility is a metric commonly used in the NLP community to evaluate a tokenizer's performance (43). It is defined as

$$F = \frac{T_A}{W_A} \tag{1}$$

where T_A is the number of tokens in an input corpus A and W_A is the number of words in A. It can be understood to mean the average number of tokens used to represent each word in the corpus.

Since genomic input has no inherent word breaks, we provide a new definition for fertility in the context of biological language model tokenizers.

$$F = \frac{T_A * 10}{C_A} \tag{2}$$

where T_A is the number of tokens in the corpus, and C_A is the total number of characters in the corpus. We multiply by ten to place the ratio in the same order of magnitude as traditional tokenizer fertility. It can be understood to represent the average number of tokens used to represent every ten nucleotides in the training corpus. Fertility was calculated using the training set of each fine-tuning dataset.

Lower fertility indicates higher compression of the corpus, and a vocabulary that better fits the training data will have lower fertility than a vocabulary that does not fit the training data well. (see Table 3). This mismatch increases the cost to train a model and makes it more challenging for a model to learn long-distance contextual relationships (23).

Parity. In multilingual models, parity is a measure of the ratio between the number of tokens in a sentence in language A with the number of tokens generated for the same sentence in language B. It is a measure of the consistency of a tokenizer between multiple languages. Since DNA does not have spaces, clear word breaks, or punctuation, different start positions of a segment of DNA can impact the consistency of tokenization of the DNA. Ideally, a tokenizer would be "shift-invariant" so that if two samples with an overlapping segment were tokenized, the overlapping segments should be tokenized the same way in both samples. This is valuable in a genomic setting because homologous sequences can be found across species, and if they are not tokenized the same way, it could impede the ability of the model to learn. The experiments by Kaiser et al (18) provide compelling evidence that tokenization inconsistency can degrade model performance.

To calculate tokenization parity, we created overlaps of 25% of the total sequence length in the training data from the GUE benchmark. We then tokenized the sequences, calculated the longest common sub-sequence (LCS), and divided the overlap length by the number of nucleotides in the LCS to obtain a percentage. The percentage is 100% when the overlapping nucleotides have been tokenized exactly the same way in both sequences.

Vocabulary Comparisons. We trained the BPE tokenizer on human, mouse, yeast and bacterial genomes to create tokenizers of vocabulary size 4096 (see Methods), and used an UpSet plot (44) to identify the intersections (tokens in common) in the vocabularies between the genomes (see Figure 4), as well as the word length distributions in the vocabularies and the intersections.

Performance in Domain-Specific Models. A comparison between the performance of mouse and human domain-specific models against the mouse transcription factor prediction task is included in Supplementary Note 7. In the BPE trained models, there was an average 0.029 improvement in accuracy, and a 0.059 score improvement in the MCC score. There was not a corresponding improvement in the character

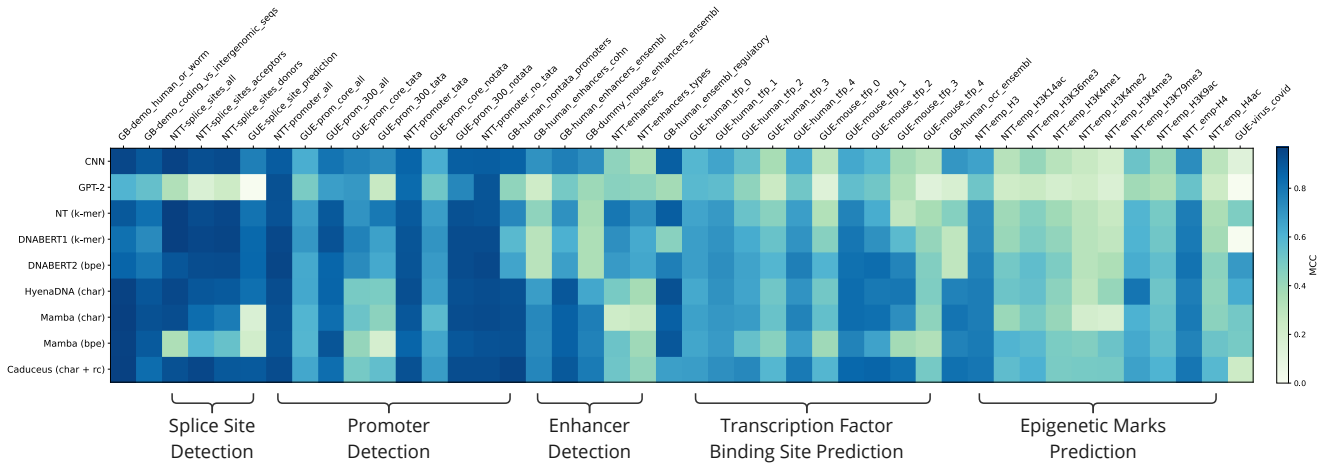

Fig. 2. Model Performance (MCC) on all Benchmark Tasks, grouped by task category and sorted by task difficulty. MCC score is visualized as a blue-green color gradient, with darker blue indicating better performance. The two baseline models, CNN and GPT-2 are at the top, followed by the three attention-based models and then the four state space models at the bottom. The MCC scores of the CNN model can be used as a proxy for the difficulty of the task, with tasks that have lower average MCC scores on the right and higher average MCC scores on the left.

tokenized models. The character tokenized models, however, did have overall better performance than the BPE models.

Correlations. In order to determine the impact of variables such as the input sequence length, number of training samples, we calculated the Spearman's rank correlation between the variables.

The adjusted mutual information score (45) was calculated to determine the relationship between the variables fertility and parity with the performance metrics, accuracy and MCC score. Mutual information measures the dependency between two variables and is zero if the two variables are independent, with higher values indicating a stronger relationship between the two variables. The mutual information is adjusted to account for relationships that may occur because of chance.

Results

Genomic Benchmarks Evaluation. The results for all three genomic benchmarks can be seen in Figure 2. The model with the highest average MCC score was Caduceus with 0.849, followed by DNABERT2 with 0.832 and then Hyena-DNA with 0.827. The baseline CNN model came within 0.025 of the MCC score of the best-performing model on the NTT splice site prediction tasks, the GUE prom core tata task, and the GB coding vs intergenomic discrimination task. The largest gap between the CNN and the highest performing gLM was on the GUE Covid variant classification task with nine classes where the difference in performance as measured by MCC was 0.668. The average MCC score for the CNN model on all tasks was 0.791. The GPT-2 model, on the other hand, had a better than random performance but did not approach any of the gLMs in MCC scores. The light band on the second row of the heatmap illustrates that the GPT-2 baseline model has relatively poor performance across most tasks. It had an average MCC score of 0.707.

On discrimination tasks, such as classifying a segment as human/worm or coding/intergenomic, the state space models

Table 5. Spearman's Rank Correlation between sequence length and number of samples and model performance on the promoter category tasks.

Variable 1	Variable 2	Spearman's Rank Correlation	p-value
Sequence Length	Accuracy	0.541	0.106
Sequence Length	MCC	0.617	0.057
Number of Samples	Accuracy	0.569	0.086
Number of Samples	MCC	0.483	0.157

had slightly higher MCC scores. While on splice site detection, all models performed very well, except the two Mamba models, with the Mamba-bpe model performing significantly worse than the Mamba-char model.

The promoter detection category results seemed more strongly correlated with the individual benchmark dataset than the type of promoter task. MCC scores on the NTT promoter tests were consistently higher for all models than the GUE and GB promoter tasks. There were two tasks, both GUE benchmark promoter identification with tata boxes, where the attention-based models outperformed the state space models, while the GB human non-tata promoter task showed opposite results, with the state-space models having a higher MCC score. The promoter tasks vary significantly in their input sequence length and number of samples, so we calculated the Spearman's rank correlation between sequence length, number of samples, accuracy, and MCC scores (see Figure 5). We observed a positive correlation between segment length and both MCC and accuracy, as well as a positive correlation between the number of samples and both MCC and accuracy. As a comparison, when we computed the same metrics on the full dataset, the Spearman's rank correlation between sequence length and both model performance metrics is negative (see Table 8 in the Supplementary Material).

Similarly, the enhancer detection tasks show inconsistent results, with state space models performing better than attention on the GB enhancer tasks, while the attention models perform better on the NTT enhancer tasks. Whether there are intrinsic differences in the selection methods for benchmark

Table 6. Adjusted Mutual Information between Parity, Fertility, Accuracy and MCC scores in Models using BPE tokenization.

Adjusted Mutual Information			
		Accuracy	MCC
Fertility	DNABERT2	0.023	0.031
	Mamba-bpe	-0.029	0.031
Parity	DNABERT2	-0.036	0.091
	Mamba-bpe	0.150	0.091

dataset positive and negative samples that benefit attention-based versus state-space gLM models remains to be explored. This understanding is critical for optimizing the predictive abilities of gLM in novel datasets and for new questions.

On the tasks that were the most challenging for the gLMs, transcription factor binding site detection, epigenetic mark prediction, and Covid variant prediction, the state space models performed better on average than attention-based models.

Byte Pair Encoding vs Character Tokenization. The results of the direct comparison between byte-pair encoding and character tokenization in the state space model Mamba can be seen in Figure 3. Markers that appear above the diagonal line indicate that character tokenization has better performance on these tasks. Markers that appear below the diagonal indicate that BPE tokenization has better performance on these tasks. In the Nucleotide Transformer tasks, the Mamba-char model has better performance on the splice site prediction task, and the Mamba-bpe model has better performance on the epigenetic marks prediction tasks. On the Genomic Benchmark, there is minimal difference between the character and BPE models. The GUE benchmark shows that the Mamba-char model has better performance on transcription factors and promoters and there are no tasks where the Mamba-BPE model has consistently better results than Mamba-char.

Consistent with the results in the ByT5 experiments (25), adding more layers to the model improves the performance of the character tokenized model, with the highest scores in most of the Epigenetic tasks moving from the Mamba-bpe column to the Mamba-char column when additional model layers were added. (see Table 7 in Supplementary Material).

Vocabulary Evaluation. To investigate how well the BPE tokenization was able to learn the words (tokens) and vocabularies (token sets) of the language of DNA, we analyzed the token set intersections between the domain-specific vocabularies as well as the token length distributions. The intersections between the domain-specific vocabularies are consistent with the idea that there is a common vocabulary in DNA. For example, the yeast, mouse and human vocabularies share 25% of their most common tokens (see Figure 4. But, there is also high variability between genomes. Only 3% of the tokens are shared by all five of the domain-specific tokenizers. The token length distribution shows that DNABERT2, mouse and human all have more tokens with a length greater than 20 nt in their token sets than bacteria and yeast.

Fertility Evaluation. We calculated the fertility on the GUE datasets using the tokenizers trained on human, mouse, yeast, and bacterial genomes, as well as the DNABERT2 tokenizer. The average fertility was 2.13 with a standard deviation of 0.11. In NLP models, a lower fertility indicates that a tokenizer better fits the corpus. For example, the average fertility of the GPT-2 tokenizer is 5.24, which is significantly higher than any of the DNA trained tokenizers. Average fertility scores for domain-specific tokenizers differed between datasets, with the epigenetic mark prediction tasks having very low and stable fertility using any tokenizer, while the transcription factor datasets had fertility with higher variance (see Figure 4b). This suggests that there is something intrinsic to the dataset category or the actual task being evaluated that affects fertility values. Though differences in fertility scores of the DNA trained domain-specific tokenizers are very small, the mouse tokenizer has the lowest fertility score on several of the mouse transcript factor task, and the yeast tokenizer has the lowest fertility on the epigenetic marks prediction task, which is from a yeast dataset. The small differences in fertility are consistent with the idea that there is a common DNA vocabulary that is learned by BPE tokenization, and that most any DNA trained tokenizer will provide approximately the same level of compression. The adjusted mutual information score between fertility, accuracy and MCC score indicates that in gLMs, fertility has little meaningful impact on accuracy or MCC score (see Table 6).

Parity Evaluation. Average parity scores on each dataset range from 40% to 95%, and the variability is different depending on the task category. Transcription factors have wide variance, whereas the epigenetic marks prediction task has a minimal variance (see Figure 4c). The parity score is also correlated with the length of the input sequence (Spearman correlation = 0.65. $R^2 = 0.42$). Longer input sequences with longer overlaps have higher parity, and shorter sequences with shorter overlaps have lower parity. The low parity scores we observe indicate that there is significant variation in the tokenization of homologous sequences, which could be impeding the ability of gLMs that use BPE to fully understand the language of DNA. The adjusted mutual information score between parity, accuracy, and MCC score indicates that there is a small but measurable relationship between parity and both accuracy and MCC score (see Table 6).

Discussion. Our results suggest that there is not one gLM that outperforms all other models. Additionally, there are inherent differences between the abilities of state-space and attention-based models that result in different performance in different task categories.

We observe that gLMs are not able to significantly outperform the simple CNN model on promoter tasks, splice site tasks and many of the GB tasks, compared to the other two benchmarks. These tasks are similar in that they have high MCC scores by all models. On tasks that seem more challenging for the models, such as the COVID variant detection, transcription factor detection, and epigenetic mark prediction, we find that the majority of the gLMs significantly

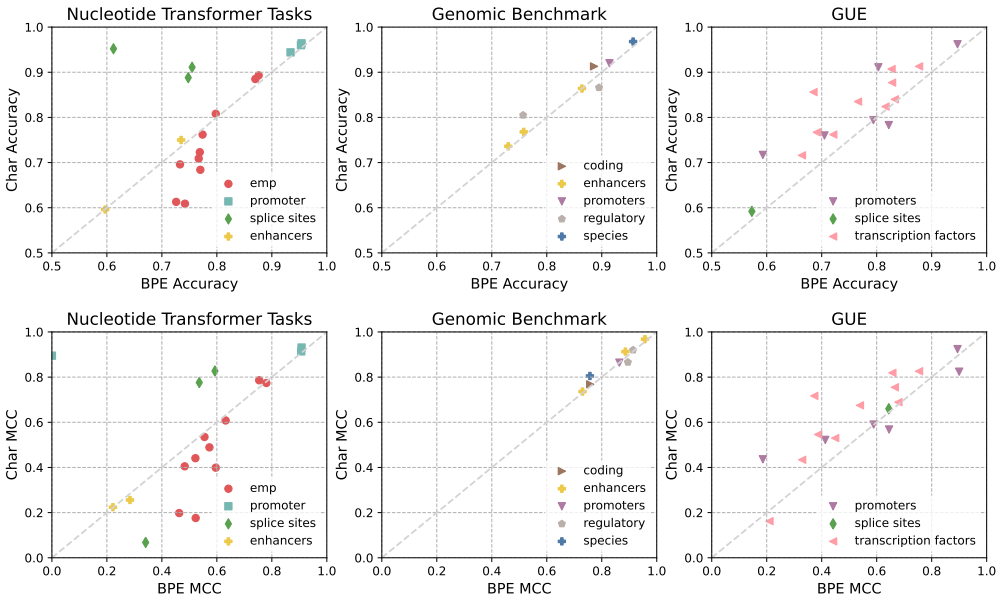

Fig. 3. Tokenization comparison in Mamba on all three genomic benchmarks. Markers below the diagonal line indicate that the Mamba model with BPE tokenization has higher performance than character-based tokenization on the same dataset. Accuracy differences are shown in the top row while MCC differences are shown in the bottom row. Markers are colored by task category.

outperform the simple CNN model.

When using attention-based models, tokenization methods that compress the input, thereby increasing the total information per sample given to a model and significantly reducing the computational cost to train, are preferred. In state-space models, where a limited context window is not a concern, our experiments indicate that character-based tokenization are the best choice for all genomic language tasks except epigenetic mark prediction. We also observe that a slight increase in the depth of the model can improve performance when using character-based tokenization.

Inconsistency in model performance in the same task category across benchmark datasets, such as the results in the enhancer and promoter detection datasets, indicates that these differences could be artifacts of the dataset. The strong Spearman's rank correlations between input sequence length, number of samples in the training dataset and model performance indicate that the selection of data to be included in genomic benchmarks can skew the perception of performance. Nevertheless, there are some clear trends that can be observed in Figure 2.

We observe that fertility has little to no impact on model performance but there is a modest relationship between parity and model performance, suggesting that low parity could potentially impede model progress for BPE models. The results from this study were limited to vocabularies of size 4096, but larger token vocabularies may have different properties. We also believe that more complex, curated datasets are needed to create additional genomic benchmarks so that both the strengths and weaknesses of gLMs can be better understood.

We hope that these experiments provide a new perspective in gLM research, as well as new metrics that can be used to help build and evaluate more powerful genomic language models. We encourage future research to focus on understanding how gLMs differ from traditional LLMs, so that appropriate architecture changes can be made to improve performance.

Competing interests

No competing interest is declared.

Author contributions statement

L.L, N.P., A.B., and H.S conceived the experiments, L.L. and A.H conducted the experiments. L.L. wrote the manuscript. L.L, N.P, W.Z.S., A.B and H.S analysed the results and reviewed the manuscript.

Acknowledgments

This work is supported by funds from the National Science Foundation (NSF: #1636933, #1920920, #2222322). This work used Bridges-2 at Pittsburgh Supercomputing Center and Delta at the National Center for Supercomputing Applications (NCSA) through allocation BIO230092 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.
We would like also to acknowledge and thank Yair Schiff for his technical support for the state space models and Zhihan Zhou for his technical support for DNABERT2.

1. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. arXiv:1810.04805 [cs].
2. Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. *OpenAI blog*, 2019.
3. Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. doi: 10.1126/science.ade2574. Publisher: American Association for the Advancement of Science.

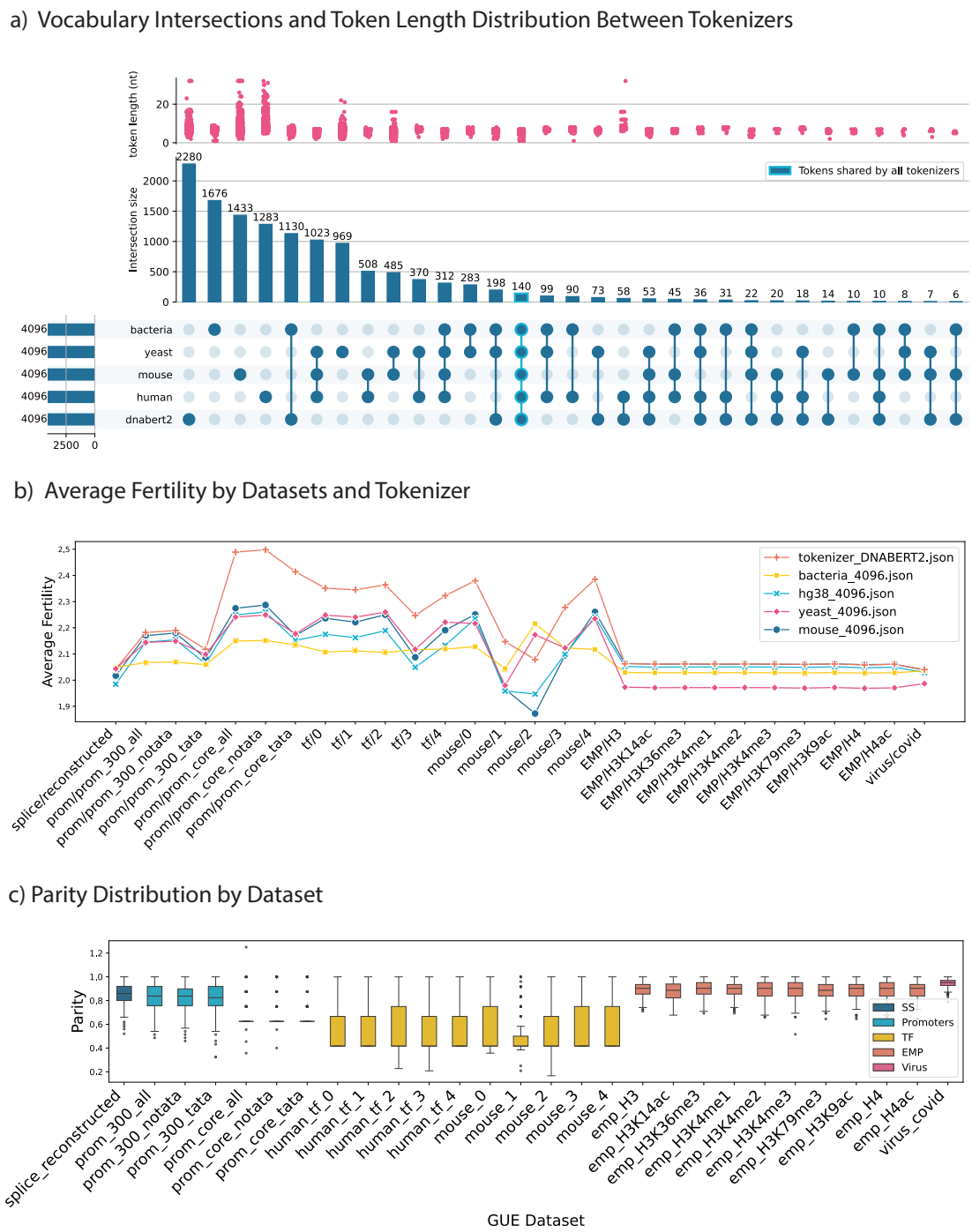

Fig. 4. Domain Specific Vocabulary, Fertility and Parity Evaluation. a) UpSet plot showing the intersection in word vocabulary (common words) in the domain-specific tokenizers (blue bar graph). Each tokenizer has been trained on a domain-specific dataset and learned a domain-specific vocabulary, which consists of the most common 4096 "words" (tokens) in that dataset. The first column (2280) indicates the number of words that are unique to the DNABERT2 tokenizer, while column five (1130) indicates the token set intersection between bacteria and DNABERT2. The word length distribution is shown above (pink dot graph). b) Comparison of the average fertility of all domain-specific BPE tokenizers on the GUE Dataset. Fertility, the token per word ratio, is lower when a token set is a good fit for the data. c) Parity distribution on all of the GUE datasets using the DNABERT2 tokenizer, colored by task category. The Parity metric measures a tokenizer's ability to tokenize a DNA segment in the same way, in different contexts. In this case, two adjacent segments of DNA that have 25% overlap. A parity of 100% indicates that the overlap was tokenized exactly the same in both cases. A parity score of 50% would indicate that only half of the overlapping sequence was tokenized into the same tokens.

4. Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alex Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. In *Advances in Neural Information Processing Systems*, volume 34, pages 29287–29303. Curran Associates, Inc., 2021.

5. Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, August 2023. ISSN 1546-1696.

doi: 10.1038/s41587-022-01618-2. Publisher: Nature Publishing Group.
6. Sam Boshar, Evan Trop, Bernardo P. de Almeida, Liviu Copoiu, and Thomas Pierrot. Are Genomic Language Models All You Need? Exploring Genomic Language Models on Protein Downstream Tasks, May 2024. Pages: 2024.05.20.594989 Section: New Results.
7. Melissa Sanabria, Jonas Hirsch, Pierre M. Joubert, and Anna R. Poetsch. DNA language model GROVER learns sequence context in the human genome. *Nature Machine Intelligence*, pages 1–13, July 2024. ISSN 2522-5839. doi: 10.1038/s42256-024-00872-0. Publisher: Nature Publishing Group.
8. Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome, June 2023.
9. Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-Directional Equivariant Long-Range DNA Sequence Modeling. *arXiv preprint arXiv:2403.03234*, 2024.
10. Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, Stefano Ermon, Stephen A. Baccus, and Chris Ré. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution, November 2023.
11. Ziqi Tang and Peter K. Koo. Evaluating the representational power of pre-trained DNA language models for regulatory genomics, March 2024. Pages: 2024.02.29.582810 Section: New Results.
12. Rebecca Boiarsky, Nalini Singh, Alejandro Buendia, Gad Getz, and David Sontag. A Deep Dive into Single-Cell RNA Sequencing Foundation Models. *bioRxiv*, 2023. doi: 10.1101/2023.10.19.563100.
13. Martin Berglund and Brink van der Merwe. Formalizing BPE Tokenization. *Workshop on Non-Classical Models for Automata and Applications*, 388:16–27, September 2023. doi: 10.4204/eptcs.388.4. ARXIV_ID: 2309.08715 MAG ID: 4386699927 S2ID: 7d1d3cdcf59232ceb74e6235a59f18b487e66a86.
14. Anh Khoa Ngo Ho and François Yvon. Optimizing Word Alignments with Better Subword Tokenization. In *The 18th biennial conference of the International Association of Machine Translation*, Proceedings of the 18th Biennial Machine Translation Summit :Volume 1: Research Track, Miami (virtual), United States, August 2021.
15. Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization Is More Than Compression, February 2024. arXiv:2402.18376 [cs].
16. Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3118–3135, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243.
17. Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. Toward a Theory of Tokenization in LLMs, April 2024. arXiv:2404.08335 [cs].
18. Kaiser Sun, Qi Pan, Yuhao Zhang, Lan Liu, William Yang Wang, and Zhiheng Huang. Tokenization Consistency Matters for Generative Models on Extractive NLP Tasks. *Conference on Empirical Methods in Natural Language Processing*, December 2022. doi: 10.48550/arxiv.2212.09912. ARXIV_ID: 2212.09912 MAG ID: 4312090143 S2ID: 2f272ca3394656835bf32a30b80c6f8ba1c8f4c4.
19. Aaditya K. Singh and DJ Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs. *arXiv.org*, 2024. doi: 10.48550/arxiv.2402.14903. ARXIV_ID: 2402.14903 S2ID: 49c45d2a2773c537804c38d69cde67e00fbad6fe.
20. Vin Sachidananda, Jason S Kessler, and Yi-An Lai. Efficient Domain Adaptation of Language Models via Adaptive Tokenization. *SUSTAINLP*, 2021. doi: 10.18653/v1/2021.sustainlp-1.16. ARXIV_ID: 2109.07460 S2ID: 3c477f0e43660ce1ba39c111df312929da37f81f.
21. Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, May 2023. ISSN 2730-6844. doi: 10.1186/s12863-023-01123-8.
22. Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics, March 2023.
23. Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Schulze Buschhoff, Charvi Jain, Alexander Arno Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. Tokenizer Choice For LLM Training: Negligible or Crucial?, October 2023.
24. Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Guochao Jiang, Jiaqing Liang, and Deqing Yang. Tokenization Matters! Degrading Large Language Models through Challenging Their Tokenization, May 2024. arXiv:2405.17067 [cs].
25. 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].
26. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter 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.
27. Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. CANINE: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, January 2022. ISSN 2307-387X. doi: https://doi.org/10.48550/arXiv.2103.06874. arXiv:2103.06874 [cs].
28. Ximena Gutierrez-Vasques, Christian Bentz, and Tanja Samardžić. Languages Through the Looking Glass of BPE Compression. *Computational Linguistics*, 49(4):943–1001, December 2023. ISSN 0891-2017. doi: 10.1162/coli_a_00489.
29. Cagri Toraman, Eyup Halit Yılmaz, Furkan Sahinuc, and Oguzhan Ozcelik. Impact of Tokenization on Language Models: An Analysis for Turkish. *ACM Trans. Asian Low Resour. Lang. Inf. Process.*, 22(4):1–21, March 2023. doi: 10.1145/3578707. ARXIV_ID: 2204.08832 MAG ID: 4360951492 S2ID: 0de580957d23dd65e31b6c95e6bc5d15bc15c57d.
30. Janghoon Yang. Dictionary-based Sentiment Analysis at Subword Level. In *2024 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*, pages 8–13, July 2024. doi: 10.1109/IAICT62357.2024.10617606. ISSN: 2834-8249.
31. Edo Dotan, Gal Jaschek, Tal Pupko, and Yonatan Belinkov. Effect of tokenization on transformers for biological sequences. *Bioinformatics*, 40(4):btae196, April 2024. ISSN 1367-4811. doi: 10.1093/bioinformatics/btae196.
32. Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. Unpacking Tokenization: Evaluating Text Compression and its Correlation with Model Performance, June 2024. arXiv:2403.06265 [cs].
33. Matthias Gallé. Investigating the Effectiveness of BPE: The Power of Shorter Sequences. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1375–1381, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1141.
34. Eugene Kharitonov, Marco Baroni, and Dieuwke Hupkes. How BPE Affects Memorization in Transformers, December 2021. arXiv:2110.02782 [cs].
35. Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, August 2021.
36. Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, June 2022. arXiv:2205.14135 [cs].
37. Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, October 2021. arXiv:2106.09685 [cs].
38. Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces, August 2022. arXiv:2111.00396 [cs].
39. Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena Hierarchy: Towards Larger Convolutional Language Models, April 2023. arXiv:2302.10866 [cs].
40. Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv preprint arXiv:2312.00752*, 2023.
41. Ziga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R. Ledsam, Agnieszka Grabska-Barwinska, Kyle R. Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203, October 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01252-x.
42. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art Natural Language Processing, July 2020. arXiv:1910.03771 [cs].
43. Stephen Della Pietra, Mark Epstein, Salim Roukos, and Todd Ward. Fertility models for statistical natural language understanding. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98/EACL '98, pages 168–173, USA, July 1997. Association for Computational Linguistics. doi: 10.3115/976909.979639.
44. Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. UpSet: Visualization of Intersecting Sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992, December 2014. ISSN 1077-2626. doi: 10.1109/TVCG.2014.2346248.
45. Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, June 2009. Association for Computing Machinery. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553511.