KL3M Tokenizers: A Family of Domain-Specific and Character-Level Tokenizers for Legal, Financial, and Preprocessing Applications

Michael J. Bommarito II
ALEA Institute
Stanford CodeX

Daniel Martin Katz
Illinois Tech - Chicago Kent Law
Bucerius Law School
ALEA Institute
Stanford CodeX

Jillian Bommarito
ALEA Institute

***Abstract*—We present the KL3M tokenizers, a family of specialized tokenizers for legal, financial, and governmental text. Despite established work on tokenization, specialized tokenizers for professional domains remain understudied. Our paper offers two main contributions to this area.**

First, we introduce domain-specific BPE tokenizers for legal, financial, and governmental text. Our kl3m-004-128k-cased tokenizer uses 9-17% fewer tokens than GPT-4o and Llama3 for domain-specific documents, despite having a smaller vocabulary. For specialized terminology, our cased tokenizer is even more efficient, using up to 83% fewer tokens for legal terms and 39% fewer tokens for financial terms.

Second, we develop character-level BPE tokenizers (4K, 8K, and 16K vocabulary sizes) for text correction tasks like OCR post-processing. These tokenizers keep consistent token boundaries between error-containing and correct text, making it easier for models to learn correction patterns.

These tokenizers help professional applications by fitting more text in context windows, reducing computational needs, and preserving the meaning of domain-specific terms. Our analysis shows these efficiency gains directly benefit the processing of long legal and financial documents. We release all tokenizers and code through GitHub and Hugging Face to support further research in specialized tokenization.[1]

I. Introduction

Tokenization — the process of converting raw text into discrete tokens — is a fundamental component of modern language models [1], [2] that has significant impacts on model performance [3]. While Byte-Pair Encoding (BPE) [4], [5] and similar subword tokenization approaches are widely used across domains, recent work has questioned their universal optimality, especially when trained on general corpora [6].

In this work, we present the results of our research on tokenizers we developed for the KL3M dataset and models. This research suggests that specialized domains like law and applications like OCR correction can significantly benefit from custom tokenization approaches.

As an example, consider a legal citation such as `Fed. R. Civ. P. 56(a)`. This citation refers to a critical rule in U.S. Federal court procedure that allows a party to ask the court to resolve a case or issue without a full trial. The presence of this citation is an important indicator for classification tasks, and when drafting, it is critical that motions and briefs properly invoke the rule. Similarly, financial documents are full of terms and abbreviations that are key to extraction tasks, such `EBITDA` or `diluted`.

Unfortunately, as highlighted in Table I, even the largest tokenizers from frontier labs fail to efficiently capture such language. For example, both `cl100k_base`, the tokenizer behind `gpt-4o`, and the LLaMA 3 tokenizer require three tokens for `EBITDA` or `diluted` each. In the case of legal citations, these tokenizers also fail to capture the fact that each token is an abbreviation, for example, by splitting the letter R from the abbreviating period.

While these differences may seem minor, anyone who has reviewed embedding layers or investigated inference pathologies like hallucination will appreciate the impact that such tokenizer issues can cause. Furthermore, tokenizer efficiency is a critical factor in the amount of text that can fit into a model's context window. While this has implications for the cost of training and inference generally, it is especially important for the legal and financial domain, where documents often have both more words and longer words than in other contexts.

As part of our research on datasets and models in the legal domain, we investigated a number of alternative approaches to tokenization that might address issues like the examples above. This research began with the `kl3m-001-32k` tokenizer and then branched into two separate groups of models: domain-specific BPE tokenizers and character-level BPE tokenizers.

Our domain-specific KL3M tokenizers (`kl3m-003-64k`, `kl3m-004-128k-cased`, `kl3m-004-128k-uncased`) are 9-17% more efficient than `cl100k_base`, the `gpt-4o` tokenizer, despite having a substantially smaller vocabulary. The cased variant in particular (`kl3m-004-128k-cased`) provides excellent performance across the legal and financial domains while maintaining case sensitivity, which is critical for many domain tasks.

Our character-level BPE tokenizers (`kl3m-004-char-4k`,

[1]For correspondence or assistance accessing our data and models: hello@aleainstitute.ai

TABLE I
TOKENIZATION COMPARISON ACROSS DOMAINS

Tokenizer	**Legal Text** `Fed. R. Civ. P. 56(a)`	**Financial Text** `EBITDA increased by 14.3%`
kl3m-004-128k-cased	["Fed.", " ", "R.", " ", "Civ.", " ", "P.", " 56", "(a)"]	["EBITDA", " increased", " by", " 14", ".", "3", "%"]
kl3m-004-char-8k-cased	["F", "ed", ".", " R", ".", " C", "iv", ".", " P", ".", " 56", "(", "a", ")"]	["EB", "IT", "DA", " in", "cre", "as", "ed", " by", " 14", ".", "3", "%"]
GPT-4o	["Fed", ".", " R", ".", " Civ", ".", " P", ".", " 56", "(a)"]	["EB", "IT", "DA", " increased", " by", " ", "14", ".", "3", "%"]
LLaMA 3	["Fed", ".", " R", ".", " Civ", ".", " P", ".", " 56", "(a)"]	["EB", "IT", "DA", " increased", " by", " ", "14", ".", "3", "%"]
GPT-2	["Fed", ".", " R", ".", " Civ", ".", " P", ".", " 56", "(a)"]	["E", "BIT", "DA", " increased", " by", " 14", ".", "3", "%"]
RoBERTa	["Fed", ".", " R", ".", " Civ", ".", " P", ".", " 56", "(a)"]	["E", "BIT", "DA", " increased", " by", " 14", ".", "3", "%"]

`kl3m-004-char-8k`, `kl3m-004-char-16k`), though less thoroughly researched, have been instrumental in training our OCR correction models, such as the 500M parameter `kl3m-004-correction-001` model.

The dual approach of domain-specific and character-level tokenizers within the KL3M family addresses complementary needs we faced in the KL3M project: efficient representation for the most common tasks and character-level precision for error correction in pretrain and RAG applications. Although our work focuses on legal, financial, and governmental domains, we believe similar approaches could potentially be relevant for other specialized fields.

All KL3M tokenizers are available on GitHub (https://github.com/alea-institute/kl3m-tokenizers and Hugging Face (https://huggingface.co/alea-institute), along with the source code, training data, and related models. The source code for this paper, including LaTeXand replication for figures and tables, is available at https://github.com/alea-institute/kl3m-tokenizer-paper/.

II. Background and Related Work

A. Tokenization Evolution

Tokenization approaches in NLP have evolved from simple word-splitting [7] to sophisticated algorithms optimized for specific linguistic properties and computational requirements.[8]

Word-level tokenization, common in early NLP systems [7], faced significant challenges with out-of-vocabulary words and morphological variance. Character-level approaches [**?**] addressed vocabulary limitations but generated longer sequences and lost semantic coherence. The tokenization-free CANINE approach [9] eliminates the need for explicit tokenization but required specialized model architectures.

Modern NLP systems predominantly using subword tokenization, with Byte-Pair Encoding (BPE) [4], [5] became the de facto standard for large language models [1]. Alternative methods include SentencePiece [10], which offer language-agnostic tokenization such as WordPiece used in BERT [11]. These approaches balance vocabulary size constraints with morphological awareness.

B. Byte-Pair Encoding

Byte-Pair Encoding (BPE) was originally developed as a data compression algorithm [5] and later adapted for NLP tokenization [4]. The algorithm can be summarized as:

1) Start with a vocabulary of individual characters;
2) Identify the most frequent adjacent character pair and merge them into a new token;
3) Repeat this process iteratively until a desired vocabulary size or alternative stopping condition is reached.

BPE offers several key benefits, including its ability to effectively manage rare words and morphological variations by breaking them down into smaller, meaningful subword units. This subword tokenization approach also allows BPE to maintain a fixed vocabulary size, which is crucial for computational efficiency in NLP models. Furthermore, BPE is designed to preserve frequent words as single tokens, while less common words are segmented into subwords, striking a balance between efficient representation and the ability to handle out-of-vocabulary terms.

However, standard BPE approaches have limitations. Bostrom and Durrett [6] demonstrated that BPE can be suboptimal for language model pretraining due to its frequency-based merging, which may not align with the linguistic intuitions of typical users.

C. Domain-Specific Tokenization

While general-purpose tokenizers are designed for broad applicability, specialized domains can benefit significantly from tailored tokenization approaches.

In the biomedical domain, BioBERT [12] demonstrated that even with standard tokenization, domain-specific pretraining improves performance. Similarly, SciBERT [13] for scientific text and BERTweet [14] for social media text show that domain adaptation at the model level improves performance, but they do not specifically address tokenization challenges.

Legal and financial domains present unique tokenization challenges due to specialized terminology, citation formats, and document structures. Chalkidis et al. [15] introduced Legal-BERT, which adapted BERT for legal text but retained the original WordPiece tokenizer, focusing adaptation on the model parameters rather than the tokenization approach.

In the financial domain, Araci [16] proposed FinBERT for financial sentiment analysis, while Mansar et al. [17] organized the FinSim shared task focusing on financial terminology. However, these works primarily address model adaptation rather than tokenization optimization.

To date, we are not aware of any significant effort to build and evaluate domain-specific tokenizers for legal, regulatory and financial texts.

D. Character-Level and Other Approaches

Character-level models and tokenizers have seen renewed interest for specialized applications. Ma et al. [18] proposed CharBERT, which incorporates character-level information into the model architecture. Clark et al. [9] introduced CANINE, a tokenization-free encoder that operates directly on Unicode characters.

These approaches are particularly relevant for tasks requiring character-level understanding, such as spelling correction, OCR post-processing, and handling of noisy text. However, they typically require specific model architectures designed to work with character-level input, rather than providing character-aware tokenizers for existing architectures.

E. Tokenization Evaluation

Evaluating tokenizer performance presents unique challenges. Rust et al. [3] assessed how tokenizer quality impacts model performance for multilingual applications. The authors found that the choice of tokenizer significantly affects downstream performance, particularly for languages with complex morphology.

Most tokenizer evaluations focus on intrinsic measures like vocabulary coverage or extrinsic measures of downstream task performance. There is limited work on evaluating tokenizers specifically for domain-specific applications, with most evaluations focusing on general text or cross-lingual scenarios.

Our work builds on these foundations but differs in several important ways. Unlike previous approaches that adapt models but not tokenizers to domains, we directly address the tokenization challenges in legal and financial text.

III. Methodology

This section details our approach to designing and training the KL3M tokenizers, including the data sources and tokenizer design.

A. Data Sources

The KL3M tokenizers were trained on a diverse corpus of legal, financial, and governmental documents from our KL3M dataset. A fundamental principle of our approach was to use only data that is free from copyright or licensing restrictions, ensuring that the training data and resulting tokenizers can be used without restriction.

Primary data sources included:

- US government documents and websites produced by the executive or legislative branches under 17 USC 105
- EU government documents produced under CC-BY or Decision 2011/833
- US state and federal court opinions and associated documents
- Publicy-traded and registered company filings, including financial reports and legal agreements
- Granted patents filed with the USPTO

These datasets can be browsed under our Hugging Face account: https://huggingface.co/alea-institute.

B. Tokenizer Design

While our goal is to address a number of issues with traditional BPE tokenizers, we also wanted to ensure that our tokenizers could be easily used by ourselves and others. Therefore, we constrained our implementations to be compatible with the `tokenizers` BPE implementation, also available through the `transformers` library, to maximize compatibility with existing libraries and pipelines.

1) Original Tokenizer: kl3m-001-32k: Our first tokenizer, `kl3m-001-32k`, was designed as a test bed for our first models, `kl3m-002-170m` and `kl3m-003-1.7b`. In addition to its domain-specific training corpus, this tokenizer also featured a number of alterations:

- no space (Ġ) prefixing
- a small set of custom tokens, including limited whitespace, Markdown, HTML, JSON, XML, and numeric tokens
- special tokens for both MLM and CLM tasks
- power-of-2 padding

While we successfully trained models up to 1.7B on this tokenizer, our experience, especially with the decreased efficiency from the removal of space prefixing and struggles with OCR correction, led us to split our research into two families of tokenizers - domain-specific BPEs and character-level BPEs.

2) Domain-Specific BPE: While `kl3m-001-32k`'s small vocabulary had advantages for memory usage, we unsurprisingly found that 32K tokens was inefficient for many typical generative use cases. Furthermore, we found that custom tokens were extremely useful and increased reliability in a number of important use cases. As a result, we substantially increased the size of our vocabulary to 64K and 128K, increased the size and breadth of custom tokens, standardized on NFKC normalization, and introduced an uncased variant of our 128K tokenizer for embedding models.

3) Character Tokenizers: Conversely, for error correction and normalization tasks like OCR post-processing, we found that the 32K vocabularly was likely too large, requiring substantially more parameters to learn basic operations like character confusion or transposition. Given that we needed to use these models at the scale of pretrain corpora, model size and speed was an extremely important consideration.

To address this, we developed specialized character-level tokenizers with 4K, 8K, and 16K vocabulary sizes. These tokenizers rely on a modified training technique that constrains the maximum merged token length to emphasize character-level patterns. The 4K and 8K tokenizers have a maximum token length of three characters, while the 16K tokenizer allows up to four characters per token.

Character-level tokenizers are particularly valuable for text error correction in legal and financial documents, where errors can significantly alter meaning. Errors in these domains frequently occur in predictable patterns from multiple sources including OCR, manual transcription, and user entry:

- Character confusions: similar-looking characters (e.g., "c"/"e", "5"/"S", "0"/"o", "l"/"1")
- Spacing errors: inappropriate spaces or joined words (e.g., "S tates" instead of "States")
- Character transpositions: reversed character order (e.g., "Teh" instead of "The")
- Domain-specific substitutions: legal/financial symbols (e.g., "§"/"S", "¶"/"P")
- Typographical errors: common keyboard-based mistakes (e.g., adjacent key hits, double letters)
- Phonetic errors: spelling based on pronunciation (e.g., "eksept" instead of "except")

Drawing on character-aware approaches like CharBERT [18] and CANINE [9], but with important modifications, our character tokenizers are optimized for different use cases:

- **kl3m-004-char-4k-cased:** Optimized for pure character-level models and fine-grained spelling correction, similar to character-based approaches in [18]. This tokenizer provides granular character-by-character tokenization ideal for learning exact substitutions (e.g., "c" $\rightarrow$ "e") in text errors.
- **kl3m-004-char-8k-cased:** Balanced approach for general text error correction, with slightly larger character groupings that efficiently handle both character-level errors and common error patterns in various document types.
- **kl3m-004-char-16k-cased:** Incorporates domain-specific character sequences for specialized correction tasks while maintaining character-level precision, suitable for more nuanced, domain-specific correction in legal and financial documents.

Unlike standard BPE tokenizers that treat text errors as unknown tokens or fragment them inconsistently, our character tokenizers maintain stable token boundaries between incorrect and correct forms. This consistency creates a more direct mapping for transformer models to learn correction patterns, aligning with findings from Wang et al. [19] on structure-preserving tokenization.

C. Custom Tokens

A key aspect of the KL3M tokenizers is the deliberate inclusion of domain-specific and format-specific tokens that might not emerge naturally from BPE training on a general corpus. By explicitly adding these custom tokens, we can steer both tokenizer training and downstream models more easily.

These custom tokens are grouped into the following categories:

- **Whitespace**: Combinations and repetitions of spaces, tabs, newlines, and carriage returns
- **Markdown**: Common Markdown elements, especially related to document headings, formatting, and lists
- **HTML**: Common HTML tags, including opening and closing tag substrings and attributes
- **JSON**: Common JSON tokens
- **XML**: Common XML tokens
- **Years**: Years from 1776 to 2050
- **Numbers**: Numbers from 1-999
- **Enumerations**: Enumerations such as Roman numerals, including in parenthetical format (e.g., `(iv)`)
- **Citations**: Common legal citations and related tokens derived from the Free Law Project's `reporters-db`

While the 32K tokenizer included some of these custom token groups, the 64K and 128K tokenizers contain many more custom tokens.

D. Power-of-2 Padding

All KL3M tokenizers are padded to align with powers of 2 during the final stage of training. In the event that the standard BPE training algorithm stopped before hitting the target vocabulary size, additional whitespace combination tokens (e.g., repeating spaces or newlines) are added until the vocabulary is an exact power. This provides enhanced efficiency opportunities for storage, computation, and search.

IV. Evaluation Methodology

This section describes our methodology for evaluating the performance of KL3M tokenizers compared to established tokenizers like `gpt-4o`, `LLaMA3`, and `gpt-2`. We assess three key dimensions: tokenization efficiency, domain term representation, and token size distribution.

A. Datasets and Tokenizers

We evaluated performance across five diverse datasets selected to represent different domains:

- **US Code**: Federal statutes with specialized legal terminology and structure
- **Congressional Hearings**: Formal political dialogue and legislative terminology
- **Court Documents**: Judicial language with complex citation patterns
- **SEC Filings**: Financial disclosure documents with business and accounting terminology

- **General Content**: Non-specialized texts for baseline comparison

Each dataset contains 20-100 documents, providing a representative sample of each domain's linguistic characteristics. Documents were preprocessed to remove markup while preserving text structure.

We compared the following tokenizers:

- **KL3M Standard**: `kl3m-004-128k-cased`, `kl3m-004-128k-uncased`, kl3m-003-64k
- **KL3M Character**: kl3m-004-char-4k-cased, kl3m-004-char-8k-cased, kl3m-004-char-16k-cased
- **Comparison**: `gpt-4o`, `LLaMA3`, `gpt-2`

B. Evaluation Metrics

1) Tokenization Efficiency: We measure efficiency using the tokens per character (TPC) ratio across datasets:

$$\text{TPC} = \frac{\text{Number of tokens}}{\text{Number of characters}} \tag{1}$$

TPC represents the inverse of compression ratio, with lower values indicating higher efficiency as fewer tokens are needed to represent the same text. This directly impacts computational requirements for processing text.

2) Domain Term Representation: We evaluated how effectively each tokenizer represents domain-specific terminology by measuring token counts for common legal and financial terms. This analysis reveals how well tokenizers capture specialized language patterns, which affects model performance when processing domain-specific content.

3) Token Size Distribution: We analyzed the distribution of token lengths (in characters) across each tokenizer's vocabulary, categorizing tokens as short (1-2 characters), medium (3-6 characters), or long (7+ characters). This distribution provides insights into tokenization strategies and their implications for efficiency and semantic preservation.

4) Character Tokenizer Evaluation: For the specialized KL3M character tokenizers, we conducted additional analyses comparing their tokenization patterns on text containing errors typical of OCR and manual transcription processes. This evaluation demonstrates how consistent character-level tokenization affects error correction capabilities.

V. Results

This section presents our evaluation results comparing KL3M tokenizers with `gpt-4o`, `LLaMA3`, and `gpt-2` tokenizers across two key dimensions: tokenization efficiency and domain-specific terminology representation. A detailed analysis of token size distribution is available in Appendix A.

A. Tokenization Efficiency

Tokenization efficiency directly impacts both computational resource requirements and context window utilization for transformer models. We measure efficiency using tokens per character (TPC), which represents the inverse of compression ratio—lower values indicate better efficiency as fewer tokens are needed to encode the same text. Table II presents a comprehensive comparison across datasets:

As demonstrated in Table II, while `kl3m-004-128k-uncased` achieves the lowest TPC values across all datasets, our primary `kl3m-004-128k-cased` model shows nearly equivalent efficiency with particularly significant gains in domain-specific content:

- **Overall efficiency**: Our cased model delivers 9% average improvement over `gpt-4o` and `LLaMA3` (0.2417 vs. 0.2636/0.2647)
- **US Code**: 17% improvement over `gpt-4o` and `LLaMA3` (0.3181 vs. 0.3716/0.3717)
- **Congressional Hearings**: 8% improvement (0.2292 vs. 0.2482/0.2475)
- **SEC Filings**: 9% improvement (0.1816 vs. 0.1976/0.1992)

These efficiency gains directly translate to practical benefits: for a typical 100K-character legal document, `kl3m-004-128k-cased` would require approximately 24,170 tokens compared to 26,360 tokens for `gpt-4o`—a difference that can significantly impact context window utilization. Appendix C provides the absolute token counts for each tokenizer across all tested datasets, showing that `kl3m-004-128k-cased` requires 8-10% fewer tokens than `gpt-4o` and `llama3` for the same content. More detailed visualizations of these efficiency metrics can be found in Appendix B.

Figure 1 reveals that KL3M tokenizers achieve superior efficiency despite having vocabulary sizes comparable to other tokenizers. This indicates that vocabulary composition and token distribution—not just vocabulary size—are critical factors for tokenization efficiency.

B. Domain Term Representation

Efficient representation of domain-specific terminology directly impacts how accurately and efficiently models can process specialized content. A key advantage of domain-optimized tokenizers is their ability to represent technical terms with fewer tokens, preserving semantic integrity. Tables III and IV demonstrate how KL3M tokenizers achieve substantial improvements in this critical dimension:

The data reveals significant differences in how tokenizers handle specialized terminology:

- **Legal terminology**: `kl3m-004-128k-cased` encodes terms like "certiorari" and "habeas corpus" in just 1-2 tokens, while `gpt-4o` and `LLaMA3` require 3-5 tokens. For complex legal citations (e.g., "42 U.S.C. § 1983"), our cased tokenizer requires 5 tokens compared to 9-10 tokens for other tokenizers.
- **Financial terminology**: Terms like "EBITDA" are represented as a single token in `kl3m-004-128k-cased` but require 3-5 tokens in other tokenizers. Even for complex terms like "Basel III compliance," our cased tokenizer achieves a 25-40% reduction in token count.
- **Overall efficiency**: As shown in Table IV, `kl3m-004-128k-cased` requires an average of

TABLE II
TOKEN EFFICIENCY (TOKENS PER CHARACTER) ACROSS DATASETS
Note: Lower values indicate more efficient tokenization (fewer tokens per character).

Dataset	kl3m-004-128k-cased	kl3m-004-128k-uncased	kl3m-003-64k	gpt-4o	llama3	gpt2
Congressional Hearings	0.2292	**0.2231**	0.2808	0.2482	0.2475	0.3765
Court Documents	0.2741	**0.2692**	0.2907	0.2971	0.2986	0.3524
General Content	0.2057	**0.2033**	0.2223	**0.2033**	0.2066	0.2116
SEC Filings	0.1816	**0.1804**	0.1915	0.1976	0.1992	0.3720
US Code	0.3181	**0.3171**	0.3296	0.3716	0.3717	0.3595
Average	0.2417	**0.2386**	0.2630	0.2636	0.2647	0.3344

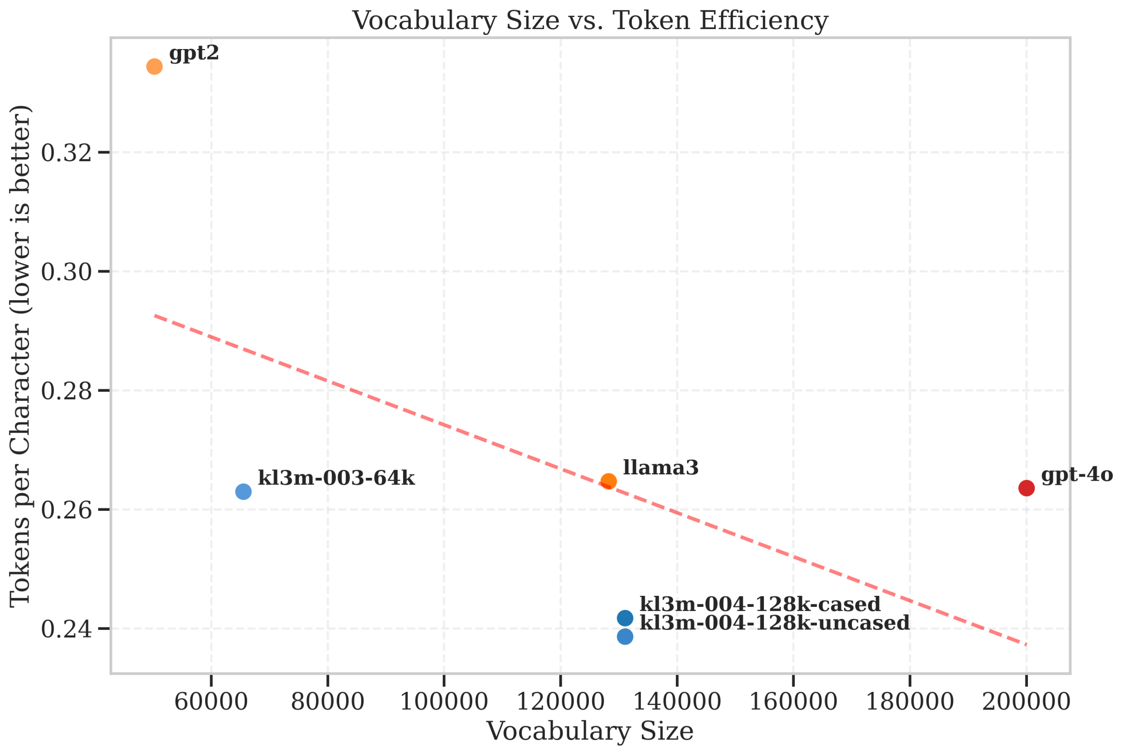

Fig. 1. Relationship between vocabulary size and tokenization efficiency. KL3M tokenizers achieve superior efficiency despite having comparable vocabulary sizes to other tokenizers.

just 3.65 tokens for domain-specific terms, compared to 4.85 for `gpt-4o` (33% more), 6.00 for `LLaMA3` (64% more), and 5.55 for `gpt-2` (52% more).

This more efficient representation offers two key advantages: (1) it reduces computational overhead by requiring fewer tokens for domain-specific content, and (2) it preserves semantic integrity by keeping domain concepts as atomic units rather than fragmenting them into semantically meaningless subwords. This preservation of semantic boundaries is particularly valuable for fine-tuning models on domain-specific tasks where conceptual precision is critical.

C. Character Tokenizer Performance

Our specialized KL3M character tokenizers (4K, 8K, and 16K variants) represent a distinct approach optimized for text normalization and error correction. Unlike standard tokenizers, these are designed with deliberately constrained vocabulary sizes and maximum token lengths (2-5 characters), resulting in more consistent tokenization patterns that facilitate character-level transformations.

Comparative analysis across these variants revealed that:

- The 4K variant provides maximum granularity for pure character-level operations
- The 8K variant offers an optimal balance between granularity and efficiency for general text correction
- The 16K variant incorporates larger domain-specific character sequences beneficial for specialized correction tasks

This approach is particularly valuable for legal and financial documents where errors from OCR, transcription, or manual

TABLE III
TOKEN COUNT COMPARISON FOR DOMAIN-SPECIFIC TERMINOLOGY ACROSS TOKENIZERS

Domain	Term	kl3m-004-128k-cased	kl3m-004-128k-uncased	gpt-4o	llama3	roberta-base	gpt2
Legal	11 U.S.C. §362(a)	**6**	**6**	10	11	15	13
	res judicata	**2**	**2**	3	5	6	4
	stare decisis	**3**	**3**	4	5	7	5
	habeas corpus	**2**	**2**	4	5	5	3
	certiorari	**1**	**1**	3	4	5	3
	de novo review	**3**	**3**	**3**	4	6	4
	28 C.F.R. §14.2(a)	**8**	**8**	12	13	16	14
	42 U.S.C. §1983	**5**	**5**	9	10	11	9
	Fed. R. Civ. P. 12(b)(6)	**10**	**10**	14	15	16	14
	prima facie	**2**	**2**	3	5	6	4
Financial	EBITDA	**1**	**1**	3	4	5	3
	P/E ratio	4	4	**3**	4	6	4
	10-K filing	4	4	**3**	4	6	4
	SEC Form 8-K	**5**	**5**	**5**	6	7	**5**
	quarterly dividend	**2**	**2**	3	4	5	3
	year-over-year growth	6	6	**4**	5	8	6
	Basel III compliance	**3**	**3**	4	5	6	4
	GAAP accounting	**2**	**2**	3	4	5	3
	ROI analysis	**2**	**2**	**2**	3	5	3
	market capitalization	**2**	**2**	**2**	4	5	3

TABLE IV
AVERAGE TOKEN COUNT BY DOMAIN ACROSS TOKENIZERS

Tokenizer	Legal Terms	Financial Terms	Overall
kl3m-004-128k-cased	**4.20**	**3.10**	**3.65**
kl3m-004-128k-uncased	**4.20**	**3.10**	**3.65**
gpt-4o	6.50	3.20	4.85
llama3	7.70	4.30	6.00
roberta-base	9.30	5.80	7.55
gpt2	7.30	3.80	5.55

entry can significantly alter meaning and where preserving exact formatting is critical.

1) Text Error Correction Example: Table V demonstrates how the character-level tokenizers process text errors compared to standard tokenizers. Consider the text "Thc Vnited S tates 5enate is nesp0nslbe for the" (an error-containing version of "The United States Senate is responsible for the," representative of OCR mistakes, transcription errors, or user typos).

As shown in Table V, the character-level tokenizers precisely tokenize each character or small character group in the error text. This fine-grained tokenization allows models to:

- Make direct character-to-character mappings (e.g., recognizing "Thc" $\rightarrow$ "The")
- Detect and correct transposition errors (e.g., "nesp0nslbe" $\rightarrow$ "responsible")
- Handle insertion/deletion errors (e.g., "S tates" $\rightarrow$ "States")
- Correct character confusions and substitutions (e.g., "5enate" $\rightarrow$ "Senate", "0" $\rightarrow$ "o")

As Table V illustrates, the key advantage of KL3M character tokenizers compared to standard BPE approaches is their consistent tokenization patterns between error-containing and correct text forms. While standard tokenizers like `gpt-4o` produce radically different token boundaries when handling errors, our character tokenizers maintain stable segmentation patterns that enable more effective learning of correction mappings.

This structured approach to character-level tokenization benefits several practical use cases:

- **OCR post-processing**: Correcting scanning errors in legal documents and financial statements where errors can significantly alter meaning
- **Transcription normalization**: Standardizing court tran-

TABLE V
CHARACTER-LEVEL TOKENIZATION OF TEXT ERRORS FROM TEST DATA

Tokenizer	**Error Text** *Thc Vnited S tates 5enate is nesp0nslbe for the*	**Correct Text** *The United States Senate is responsible for the*	**Notes**
kl3m-004-char-4k-cased	["Th", "c", " V", "n", "it", "ed", " S", " t", "at", "es", " 5", "en", "ate", " is", " n", "esp", "0", "ns", "l", "be", " f", "or", " t", "he"]	["The", " Un", "it", "ed", " St", "at", "es", " S", "en", "ate", " is", " re", "sp", "ons", "ib", "le", " f", "or", " t", "he"]	Preserves character positioning
kl3m-004-char-8k-cased	["Th", "c", " V", "nit", "ed", " S", " t", "at", "es", " 5", "en", "ate", " is", " n", "esp", "0", "ns", "l", "be", " f", "or", " t", "he"]	["The", " Un", "it", "ed", " St", "at", "es", " S", "en", "ate", " is", " re", "sp", "ons", "ib", "le", " f", "or", " t", "he"]	Balances character groups
kl3m-004-char-16k-cased	["Th", "c", " V", "n", "ited", " S", " t", "ates", " 5", "en", "ate", " is", " n", "esp", "0", "ns", "l", "be", " for", " the"]	["The", " Un", "ited", " St", "ates", " Sen", "ate", " is", " re", "sp", "ons", "ible", " for", " the"]	Larger character groupings
gpt-4o	["Th", "c", " V", "n", "ited", " S", " t", "ates", " ", "5", "en", "ate", " is", " n", "esp", "0", "n", "sl", "be", " for", " the"]	["The", " Un", "ited", " States", " Sen", "ate", " is", " respons", "ible", " for", " the"]	Less consistent boundaries

scripts, congressional records, and regulatory filings where formatting consistency is critical

- **Document digitization**: Converting legacy documents where character-level errors are common but systematic
- **Legal citation standardization**: Ensuring consistent formatting of case citations, statutory references, and regulatory citations

The efficiency gains from this approach are particularly valuable in large-scale document processing workflows where both accuracy and computational efficiency are critical considerations.

VI. DISCUSSION

Our evaluation results demonstrate the advantages of domain-specific tokenization for legal, financial, and governmental text. This section discusses the practical implications of our findings and examines key limitations and challenges of specialized tokenizers.

A. Practical Benefits for Professional Applications

The efficiency advantages of KL3M tokenizers translate to several practical benefits:

- **Expanded effective context window:** When processing legal or financial documents with thousands of domain-specific terms and citations, the 9-17% overall efficiency improvement and up to 83% improvement for specific terminology substantially expands the effective context window. This allows models to process more complete documents without truncation.
- **Reduced computational costs:** The reduced token count directly translates to lower computational requirements for training and inference. For applications processing millions of legal or financial documents, this efficiency can yield significant computational savings.
- **Enhanced cross-document reasoning:** By representing citations and references more efficiently and coherently, KL3M-004-128k-cased enables better tracking of references across documents, which is crucial for legal research, financial analysis, and regulatory compliance tasks.

For fine-tuning applications, our domain-specific tokenizers also deliver notable benefits. When fine-tuning general models on legal or financial texts, tokenization mismatches between pre-training and fine-tuning data can disrupt performance. By using tokenizers specifically designed for professional content during both pre-training and fine-tuning, this mismatch can be reduced or eliminated, leading to better specialized model performance.

B. Limitations and Challenges

1) Domain Specificity vs. Generality: A core limitation of specialized tokenizers lies in the inherent trade-off between domain specificity and broader applicability. By allocating vocabulary space to domain-specific terms, these tokenizers leave less room for general terms, which can enhance performance on targeted content but may compromise effectiveness on more general text. While our analysis showed that KL3M-004-128k-cased performed well on general content, more extensive evaluation across diverse general text types would be needed to fully assess this trade-off.

Furthermore, KL3M tokenizers, designed specifically for legal and financial domains, may prove less efficient when applied to other specialized fields such as medicine or chemistry, limiting their cross-domain utility. Additionally, like any fixed-vocabulary tokenizer, they may struggle to efficiently handle novel terminology that emerges after the training data is set, potentially reducing their adaptability to evolving language.

2) Implementation Challenges: Several technical challenges surfaced during implementation. One key issue was striking the right balance between cased and uncased variants; although we provide both options, the best choice depends heavily on the specific application requirements. Another difficulty arose in custom token selection, where identifying domain-specific tokens required considerable expert knowledge, suggesting that more systematic methods could enhance the process.

Furthermore, while we obtained high-quality, copyright-free legal and financial texts for training, acquisition of additional content may be a significant limitation in subsequent efforts to further improve tokenizer quality and performance. The quality and representativeness of training data remain crucial factors in tokenizer performance.

3) Ecosystem Compatibility: Although our implementation ensures compatibility with the Hugging Face ecosystem, broader compatibility issues exist. Certain model architectures rely on assumptions about tokenization that may not align with the design of specialized tokenizers, posing integration challenges. Additionally, using our tokenizers with existing pre-trained models requires careful alignment or adjustments to embedding layers to ensure proper functionality.

The NLP ecosystem tends to prioritize support for a handful of widely used tokenizers, which could restrict the availability of tools and libraries compatible with our specialized approach. Despite these challenges, the significant efficiency advantages demonstrated in our evaluation suggest that overcoming these implementation and compatibility issues would be worthwhile for professional applications working with legal and financial documents.

C. Future Work

Several promising directions for future work emerge from our research:

- **Downstream task evaluation:** Providing public and reproducible experiments on downstream training tasks like masked language modeling (MLM) and causal language modeling (CLM) would quantify the impact of domain-specific tokenization on model quality, with particular emphasis on the OCR/error correction capabilities of our character-level tokenizers.
- **Tokenizer swapping:** Investigating the impact and methodology of tokenizer swapping for established pre-trained models like `LLaMA3` could enable existing models to benefit from domain-specific tokenization without complete retraining, potentially offering an efficient path to domain adaptation.
- **Custom token extensions:** Further refinement of the custom token selection process could enhance domain coverage, particularly through development of systematic methods to identify high-value domain-specific tokens that maximize efficiency gains across diverse professional documents.
- **Non-English professional language support:** Extending our approach to common non-English professional languages (e.g., EU legal frameworks in French and German, international financial reporting terminology) would address the growing need for multilingual domain-specific NLP capabilities in global regulatory and business contexts.

VII. Conclusion

In this paper, we presented the KL3M family of tokenizers for legal, financial, and governmental text processing. Our research demonstrates significant advantages of domain-specific tokenization for professional applications, with five key contributions:

1) A methodology for domain-specific tokenization that balances learned tokens with curated domain-specific additions, building on foundational BPE work [4].
2) A dual-approach tokenizer family with standard BPE variants (64K-128K vocabulary) for efficient representation and character-level variants (4K-16K vocabulary) for OCR correction and text normalization.
3) Quantitative efficiency improvements: Our analysis shows that `kl3m-004-128k-cased` achieves excellent tokenization efficiency with an average of 0.2417 tokens per character across datasets, representing a 9% improvement over `gpt-4o` (0.2636) and `LLaMA3` (0.2647) despite being 35% smaller than `gpt-4o`. While our uncased variant offers slightly better efficiency (0.2386 tokens per character), the cased model maintains important case distinctions necessary for many professional applications.
4) Evidence that domain-specific tokenization preserves semantic integrity of specialized terminology and citation patterns critical to professional understanding, with `kl3m-004-128k-cased` requiring an average of just 3.65 tokens for domain-specific terms compared to 4.85 for `gpt-4o` (33% more tokens) and 6.00 for `LLaMA3` (64% more tokens).
5) A rigorous evaluation framework for tokenizer performance in specialized domains, extending previous evaluation approaches [3].

The efficiency advantages of `kl3m-004-128k-cased` are particularly pronounced for specialized content. For US Code, our cased tokenizer achieved 0.3181 tokens per character compared to 0.3716 for `gpt-4o` (a 17% improvement) and 0.3717 for `LLaMA3` (a 17% improvement). For SEC filings, our cased model achieved 0.1816 tokens per character compared to 0.1976 for `gpt-4o` (a 9% improvement) and 0.1992 for `LLaMA3` (a 10% improvement). This demonstrates that even against state-of-the-art tokenizers like `gpt-4o` and `LLaMA3`, domain-specialized tokenizers can provide substantial efficiency gains.

Our domain term analysis provides even more compelling evidence for the value of specialized tokenization. For legal terminology, `kl3m-004-128k-cased` required an average of just 4.20 tokens per term, while `gpt-4o` needed 6.50 tokens (55% more) and `LLaMA3` required 7.70 tokens (83% more). For financial terms, the advantage persisted with KL3M requiring 3.10 tokens compared to 3.20 for `gpt-4o` and 4.30 for `LLaMA3` (39% more). Key legal terms like "certiorari" required just 1 token in KL3M compared to 3 tokens in `gpt-4o` and 4 tokens in `LLaMA3`, while "11 U.S.C. § 362(a)" required 6 tokens in KL3M versus 10 tokens in `gpt-4o` and 11 tokens in `LLaMA3`.

These efficiency gains directly translate to context window utilization, computational efficiency, and lower inference costs. For large legal or financial documents with thousands of domain-specific terms and citations, the 9-17% overall efficiency improvement can substantially expand the effective context window, allowing more complete document understanding without truncation. This advantage becomes particularly important for long-document reasoning and question answering, where parsing and understanding references across a lengthy document is essential.

Domain-specific tokenization represents an important frontier in NLP research. Building on domain-specific model work like Legal-BERT [15] and FinBERT [16], our results suggest that tokenization customization can provide substantial benefits complementary to model specialization. Future work could extend to multilingual legal systems, additional professional domains, and dynamic vocabulary adaptation approaches.

All KL3M tokenizers are publicly available through the Hugging Face Hub under the `alea-institute/` account to support researchers and practitioners working with legal, financial, and governmental text. The complete source code, including tokenizer training scripts and evaluation tools, is available on GitHub at https://github.com/alea-institute/kl3m-tokenizers. This open-source release enables full reproducibility and further extension of our work by the research community. Our work establishes that tokenization—far from being a solved problem—remains critical for domain-specific optimization as language models transform professional workflows in specialized domains.

VIII. Acknowledgements

We drafted and revised this paper with the assistance of large language models. All errors or omissions are our own.

References

[1] 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*, vol. 33, pp. 1877–1901, 2020.

[2] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[3] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych, "How good is your tokenizer? on the monolingual performance of multilingual language models," in *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)*, 2021, pp. 3118–3135.

[4] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1715–1725.

[5] P. Gage, "A new algorithm for data compression," *The C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[6] K. Bostrom and G. Durrett, "Byte pair encoding is suboptimal for language model pretraining," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 4617–4624.

[7] J. J. Webster and C. Kit, "Tokenization as the initial phase in nlp," in *COLING 1992 volume 4: The 14th international conference on computational linguistics*, 1992.

[8] S. J. Mielke, Z. Alyafeai, E. Salesky, C. Raffel, M. Dey, M. Gallé, A. Raja, C. Si, W. Y. Lee, B. Sagot *et al.*, "Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp," *arXiv preprint arXiv:2112.10508*, 2021.

[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*, vol. 10, pp. 73–91, 2022.

[10] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018, pp. 66–71.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[12] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[13] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3615–3620.

[14] D. Q. Nguyen, T. Vu, and A. Tuan Nguyen, "Bertweet: A pre-trained language model for english tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 9–14.

[15] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "Legal-bert: The muppets straight out of law school," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 2898–2904.

[16] D. Araci, "Finbert: Financial sentiment analysis with pre-trained language models," *arXiv preprint arXiv:1908.10063*, 2019.

[17] Y. Mansar, J. Kang, and I. E. Maarouf, "The finsim-2 2021 shared task: Learning semantic similarities for the financial domain," in *Companion Proceedings of the Web Conference 2021*, 2021, pp. 288–292.

[18] W. Ma, Y. Cui, C. Si, T. Liu, S. Wang, and G. Hu, "Charbert: Character-aware pre-trained language model," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 39–50.

[19] C. Wang, X. Liu, Z. Chen, H. Hong, J. Tang, and D. Song, "Deepstruct: Pretraining of language models for structure prediction," in *Findings of the Association for Computational Linguistics: ACL 2022*, 2022, pp. 803–823.

Appendix

A. *Token Size Distribution*

The distribution of token lengths provides important insights into tokenizer design and potential efficiency. Figure 2 presents a line plot comparing the percentage of vocabulary by token length across tokenizers.

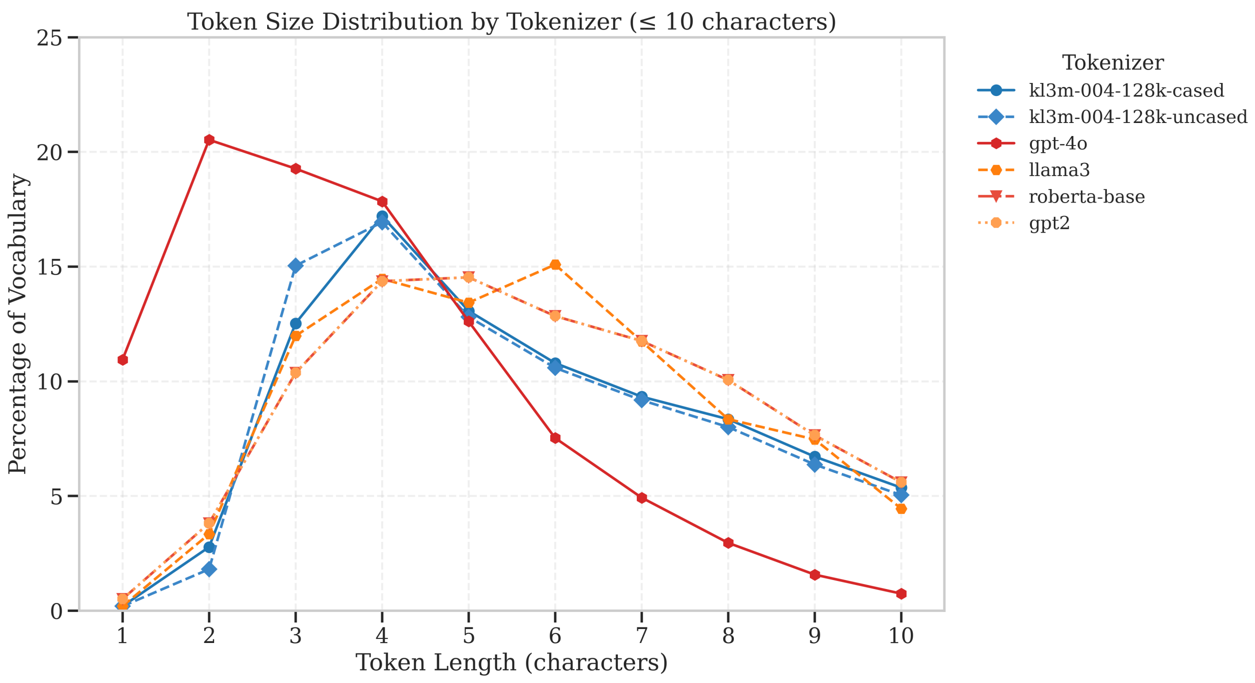

Fig. 2. Percentage of vocabulary by token length across tokenizers. KL3M tokenizers show higher percentages of medium-length tokens (3-6 characters) compared to other tokenizers.

TABLE VI
TOKEN SIZE DISTRIBUTION (PERCENTAGE OF VOCABULARY BY CHARACTER LENGTH). NOTE: FOR TIKTOKEN MODELS LIKE `gpt-4o`, ONLY A SUBSET OF TOKENS CAN BE INDIVIDUALLY DECODED, SO STATISTICS ARE BASED ON A PARTIAL SAMPLE.

Length	kl3m-004-128k-cased	kl3m-004-128k-uncased	gpt-4o	llama3	roberta-base	gpt2
1	0.2%	0.2%	10.9%	0.2%	0.5%	0.5%
2	2.8%	1.8%	20.5%	3.3%	3.8%	3.8%
3	12.5%	15.0%	19.3%	12.0%	10.4%	10.4%
4	17.2%	16.9%	17.8%	14.5%	14.4%	14.4%
5	13.1%	12.8%	12.6%	13.4%	14.5%	14.5%
6	10.8%	10.6%	7.5%	15.1%	12.8%	12.8%
7	9.3%	9.2%	4.9%	11.7%	11.8%	11.8%
8	8.3%	8.0%	3.0%	8.4%	10.1%	10.1%
9	6.7%	6.4%	1.6%	7.5%	7.6%	7.7%
10	5.4%	5.0%	0.7%	4.5%	5.6%	5.6%
Total ≤ 5	45.8%	46.8%	81.2%	43.4%	43.6%	43.6%
Total 6-10	40.6%	39.2%	17.7%	47.1%	47.9%	47.9%
Total ≤ 10	86.3%	86.0%	99.0%	90.5%	91.5%	91.5%

KL3M tokenizers, particularly the kl3m-004-128k variants, have a more balanced distribution with a higher percentage of medium-length tokens (3-6 characters). In contrast, `gpt-4o` has a significantly higher percentage of short tokens (1-2 characters), with over 31

B. *Tokenization Efficiency Visualizations*

Tokenization efficiency, measured as tokens per character (TPC), represents the inverse of compression ratio. Lower TPC values indicate better compression and higher efficiency, as fewer tokens are needed to encode the same amount of text. Figure 3 provides a visual comparison of tokenization efficiency across datasets for different tokenizers.

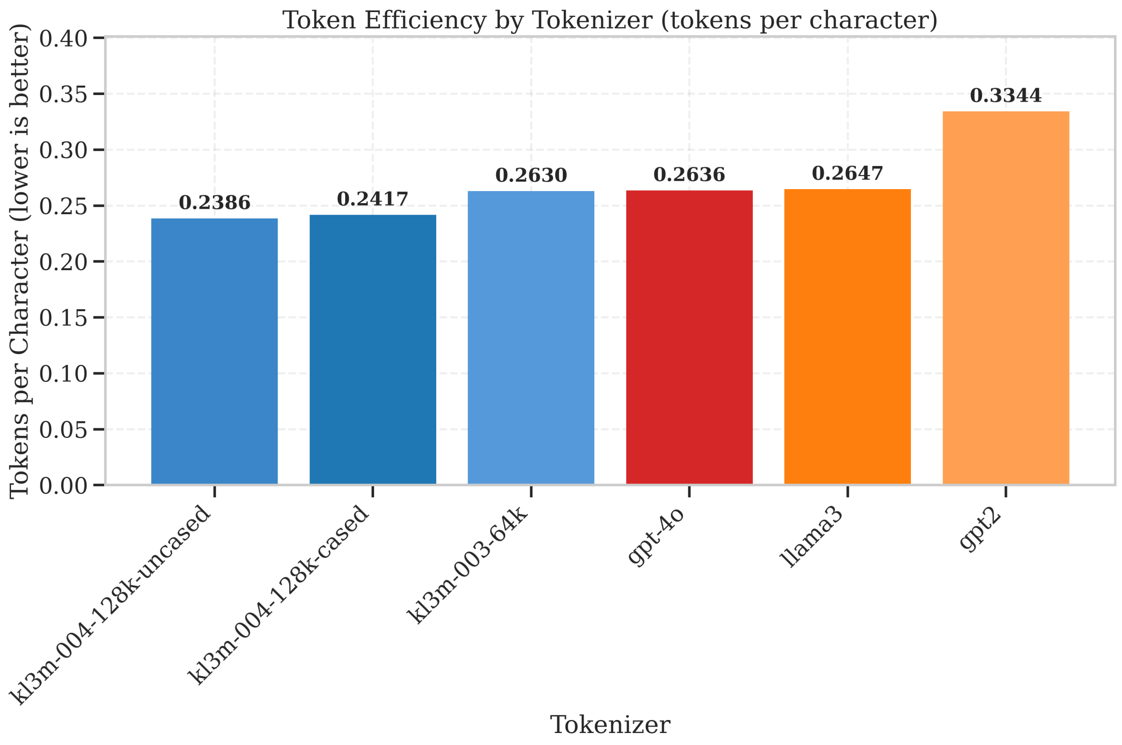

Fig. 3. Tokenization efficiency (tokens per character) across datasets. Lower values indicate higher efficiency. KL3M tokenizers consistently demonstrate higher efficiency, particularly for domain-specific content like US Code and Congressional Hearings.

C. *Total Token Count Comparison*

Table VII presents the absolute token counts for each tokenizer across the different datasets, providing a concrete measure of the efficiency improvements offered by the KL3M tokenizers.

TABLE VII
TOTAL TOKEN COUNT ACROSS DATASETS
Note: Lower values indicate more efficient tokenization (fewer tokens to represent the same text).

Dataset	kl3m-004-128k-cased	kl3m-004-128k-uncased	kl3m-003-64k	gpt-4o	llama3	gpt2
Congressional Hearings	308,169	**299,954**	377,458	333,702	332,791	506,115
Court Documents	23,682	**23,260**	25,116	25,674	25,798	30,454
General Content	22,424	**22,164**	24,237	22,167	22,522	23,068
SEC Filings	243,580	**242,057**	256,913	265,078	267,267	499,126
US Code	45,099	**44,957**	46,732	52,684	52,689	50,962
Total	642,954	**632,392**	730,456	699,305	701,067	1,109,725