Say Anything but This: When Tokenizer Betrays Reasoning in LLMs

Anonymous ACL submission

Abstract

Large language models (LLMs) reason over discrete token ID sequences, yet modern subword tokenizers routinely produce non-unique encodings: multiple token ID sequences can detokenize to identical surface strings. This representational mismatch creates an unmeasured fragility wherein reasoning processes can fail. LLMs may treat two internal representations as distinct "words" even when they are semantically identical at the text level. In this work, we show that tokenization can betray LLM reasoning through one-to-many token ID mappings. We introduce a tokenization-consistency probe that requires models to replace designated target words in context while leaving all other content unchanged. The task is intentionally simple at the surface level, enabling us to attribute failures to tokenizer-detokenizer artifacts rather than to knowledge gaps or parameter limitations. Through analysis of over 11000 replacement trials across state-of-the-art open-source LLMs, we find a non-trivial rate of outputs exhibit phantom edits: cases where models operate under the illusion of correct reasoning, a phenomenon arising from tokenizer-induced representational defects. We further analyze these cases and provide a taxonomy of eight systematic tokenizer artifacts, including whitespace-boundary shifts and intra-word resegmentation. These findings indicate that part of apparent reasoning deficiency originates in the tokenizer layer, motivating tokenizer-level remedies before incurring the cost of training ever-larger models on ever-larger corpora.

1 Introduction

Recent breakthroughs in large language model (LLM) capabilities are predominantly attributed to scaling Transformer architectures with massive datasets and computational resources, yielding emergent abilities including reasoning, code generation, and few-shot generalization (Saunshi et al., 2025; Kaplan et al., 2020). However, the prevailing "scale explains everything" narrative overlooks a foundational component that mediates all model input and output: the tokenizer (Gastaldi et al., 2025). Tokenization constitutes the mandatory translation layer between the continuous, context-dependent nature of human language and the discrete symbolic representations required for computational processing. Early tokenizers employed word-level vocabularies, which suffered from out-of-vocabulary (OOV) limitations, where any word absent from the training corpus would be mapped to an unknown token, resulting in information loss (Choo and Kim, 2023). This limitation motivated a transition to open-vocabulary schemes. Accordingly, modern LLMs adopt subword tokenization including Byte Pair Encoding (BPE) (Sennrich et al., 2016), WordPiece (Wu et al., 2016), Unigram (Kudo, 2018), and SentencePiece (Kudo and Richardson, 2018). These algorithms decompose words into frequent substrings or character-level units, substantially solving OOV and rare word problem. Yet subword tokenization also introduces new challenges. By defining how text is segmented, the tokenizer can impose structural artifacts on the model's internal representation of language. The availability of numerous subtoken units enables different subtoken combinations to concatenate into the same word, establishing representational non-uniqueness. The core of our critique centers on the many-to-one nature of the tokenizer's mapping from token ID space to surface text. While the tokenization process is designed to be deterministic for a given input, the detokenization function is not injective. Consequently, models must learn that different subtoken decompositions of the same word represent semantically identical content despite possessing different token-level encodings. Simultaneously, research findings indicate that scaling model size and data volume alone does not resolve all reasoning deficiencies in LLMs. State-of-the-art models exhibit remarkable performance

across numerous benchmarks, yet they continue to exhibit systematic failures on certain reasoning tasks, particularly as problem complexity increases. This suggests that some limitations arise from factors beyond model scale, potentially from how information is represented or processed internally.

In this paper, we argue that tokenization is a hidden contributor to the reasoning failures. We introduce a simple yet diagnostic tokenization-consistency probe: models are instructed to perform straightforward word substitutions while preserving all other text unchanged. Success requires identifying marked spans, applying designated edits, and preserving unmarked regions. Under correct operation, an LLM should recognize that generating alternative token sequences which decode to the same word does not constitute a genuine edit and must be avoided. However, our empirical results reveal an alarmingly high rate of tokenizer-induced "phantom edits". In these cases, models output different token ID sequences for target words while the decoded surface text remains identical. The transformation appears successful from the model's internal token-based perspective (IDs differ), yet it is semantically null from the text-based evaluation perspective (strings match). This demonstrates that models are systematically misled by tokenizer properties: they "believe" they have successfully executed substitutions when no actual content change has occurred. Importantly, these failures do not reflect knowledge limitations or model scale, but rather expose a fundamental architectural constraint: **the inability to recognize that different token-level representations can encode identical linguistic content**.

Our analysis of over 11k replacement trials across ten state-of-the-art open-source LLMs reveals that a non-trivial fraction of reasoning failures stem not from knowledge limitations or model capacity constraints, but from tokenizer-induced representational non-uniqueness. We further characterize the mechanisms underlying these failures through a taxonomy of eight systematic artifact types, including whitespace-boundary shifts, whitespace detachment and reattachment, newline substitutions, intra-word resegmentation, and proper-noun and morphological segmentation ambiguities. This taxonomy reveals how standard vocabulary redundancies and alternative segmentation pathways systematically mislead token-space reasoning, providing a roadmap for future tokenizer-aware architectural improvements. Additionally, through a post-hoc token ID masking intervention, we demonstrate that when these spurious representational pathways are suppressed, models frequently reveal underlying reasoning capacities that were previously overshadowed. This suggests that the reasoning deficiency observed in state-of-the-art LLMs is partly a mirage, a byproduct of a mandatory translation layer that is both lossy and non-injective. Ultimately, this research motivates a paradigm shift toward more robust representational schemes to ensure that future language models can reason as reliably at the surface level as they do within their latent embedding spaces.

2 Related work

2.1 Tokenization in modern LLMs

Subword tokenization techniques such as BPE (Sennrich et al., 2016), WordPiece (Wu et al., 2016), and Unigram (Kudo, 2018) were developed to address out-of-vocabulary (OOV) and rare word challenges. These techniques form the tokenization backbone of modern state-of-the-art LLMs. Following the release of GPT-4 and the open-sourcing of OpenAI's tiktoken library, the field broadly converged toward high-performance BPE variants. Llama 3.2 and Qwen 3 employ Byte-Level BPE, an algorithm that constructs tokens from individual bytes to enable lossless encoding of multilingual text and code while mitigating the vocabulary explosion problem inherent in character-level BPE. This approach is further refined in the Mistral model family, which implements the "Tekken" tokenizer, a specialized BPE variant optimized for compression efficiency on source code and European languages. In contrast to this BPE-dominated landscape, Gemma 3 adopts the Unigram language model implemented via SentencePiece (Kudo and Richardson, 2018), utilizing an extensive vocabulary exceeding 256k tokens to support massively multilingual capabilities and enable direct integration of visual embeddings into the text stream. Despite their widespread success, fixed-vocabulary tokenizers have been criticized as a limiting inductive bias. These limitations have motivated "token-free" approaches that compute directly on bytes or characters as the atomic units of digital text (Clark et al., 2022; Xue et al., 2022; Tay et al., 2022).

2.2 Advances in LLM reasoning

LLMs have achieved significant reasoning improvements (Fang et al., 2024) through structured

prompting approaches, notably Chain-of-Thought (CoT) (Wei et al., 2022; Besta et al., 2025; Wang et al., 2023) and Zero-Shot CoT (Kojima et al., 2022), with subsequent work automating and refining these techniques through methods like Auto-CoT (Zelikman et al., 2024; Wei et al., 2022). Beyond prompting, augmenting LLMs with external tools and APIs further enhances their reasoning capabilities (Gou et al., 2024; Lu et al., 2023; Dong et al., 2025). Methods such as ReAct (Yao et al., 2023) interleave logical reasoning steps with executable action commands, while frameworks like Toolformer (Schick et al., 2023) enable models to invoke external resources including calculators, search engines, and specialized APIs, yielding improved performance on factual accuracy, numerical computation, and zero-shot reasoning tasks. Parallel research on retrieval-augmented generation (Tran et al., 2025) extends LLM capabilities by incorporating external knowledge bases that can be queried dynamically, effectively providing models with expandable memory beyond their parametric knowledge (Izacard et al., 2023).

2.3 Tokenization as a reasoning bottleneck

By fragmenting semantic continuity into discrete integer representations, tokenizers can constitute a bottleneck in LLM semantic understanding (Haslett and Cai, 2025). In quantitative reasoning, this appears as structural misalignment: greedy left-to-right subword tokenization can split integers and dates into arbitrary, non-semantic fragments, thereby obscuring the positional structure required for arithmetic and temporal inference (Singh and Strouse, 2024; Bhatia et al., 2025; Fatemi et al., 2024). Manipulating digit-grouping tokenizations reveal substantial accuracy shifts and altered error distributions in state-of-the-art LLMs, indicating that model "reasoning" frequently adheres to tokenizer-imposed structure rather than learned abstract computational principles. More broadly, small surface perturbations like typos, spacing changes, and rare Unicode characters can trigger different segmentations and produce behavioral changes (Chai et al., 2024). Proposed mitigations include exposing models to multiple valid tokenizations during training (Provilkov et al., 2020), and designing token-free architectures (Slagle, 2024).

3 Method

3.1 Task setup overview

LLMs execute instructions by operating over token ID sequences produced by tokenizers. Tokenization poses a particularly strict challenge: the same surface word may be associated with different token IDs at input encoding versus output decoding, due to non-unique segmentations that detokenize to identical text. This property establishes a natural framework for probing tokenization-consistency reasoning. We define the task formally: given a text with marked spans and an instruction to replace each marked span with an alternative, the model must 1) identify the target spans, 2) apply the edits, and 3) preserve all non-target text. Our probe evaluates whether models recognize that different token ID combinations can decode to identical strings and therefore represent identical semantics despite divergent internal representations. This setting isolates failures arising from the interaction between the tokenizer's discrete alphabet and the model's learned rules, thereby evaluating reasoning under the tokenizer's lossy and non-unique encoding. In contrast to standard benchmarks where reasoning failures may reflect insufficient model parameters, knowledge gaps, or sampling variance, our task specifies an unambiguous objective where failures arise because the model's internal processing depends on token IDs that lack one-to-one correspondence with surface text.

Figure 1 depicts the full experimental workflow. The selector randomly samples $p\%$ of non-stop words from each document and marks them by placing them inside brackets. We then prepend an explicit replacement instruction to the document and pass the combined text through the tokenizer. The tokenizer maps the input to a token-ID sequence that serves as the LLM's input. Finally, the model's output token IDs are detokenized to produce the output string. We evaluate replacement performance by comparing input and output texts together with their aligned token-ID sequences. Using evidence from both the surface form and the token IDs, we categorize model behavior into three classes: **1)** ***Unchanged***, where the model does not replace the target and the output token ID matches the input token ID; **2)** ***Replaced***, where the model performs a successful substitution and the output token ID differs from the input token ID; and **3)** ***Different***, where the model outputs different token ID(s) that nevertheless decode to the same word,

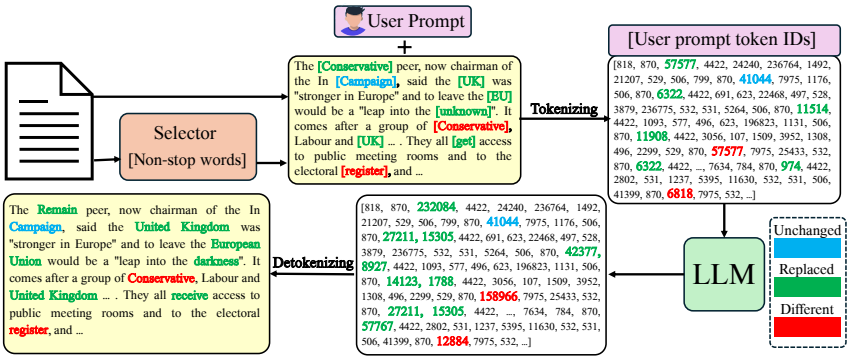

Figure 1: The overview of the proposed tokenization-consistency probe.

yielding the same surface string as the original input word, despite token-id variation. In the example shown in Figure 1, the model successfully replaces several targets (e.g., get → receive), indicated in green. As expected, successful substitutions are accompanied by changes in the corresponding token IDs. Conversely, the model failed to replace the word "Campaign", which retains the same token ID in the output sequence, indicated by blue highlighting. Most critically, the red color highlights the cases where the model outputs a different token-ID representation than the input, but detokenization produces the same surface word in the final text. This phenomenon arises because tokenization is not injective, distinct token-ID sequences can map to an identical surface form. We provide a more detailed explanation of how the tokenizers enable such token-level changes without surface-level edits in Section 4.2.

3.2 Experimental setup

Dataset: we sample news articles from the XSUM dataset (Narayan et al., 2018) and retain texts between 100 and 600 words. For each selected article, we randomly sample 5% of non-stop words as replacement targets and designate these spans by enclosing them in brackets, producing a bracket-annotated version of the original text. LLMs receive the bracket-annotated document along with explicit instructions to replace only bracketed words while maintaining all other text in its original form.

LLM models: we evaluate ten state-of-the-art open-source LLMs spanning four major model families and accessed through Hugging Face: Gemma (Gemma3-270M, Gemma3-1B, Gemma3-4B, Gemma3-12B), Llama (Llama3.2-1B, Llama3.2-3B, Llama3.1-8B), Mistral (Ministral-8B, MistralSmall-24B), and Qwen (Qwen3-4B, Qwen3-30B). To ensure comparability across models, we employ a standardized sampling configuration for all generation tasks, and set Top-p, Top-k, and Temperature to 0.9, 50, and 1.0, respectively.

4 Result and discussion

4.1 Model size is not always the bottleneck for reasoning

While parameter scaling is often assumed to improve reasoning capacity, our experiments show that it is not always sufficient to ensure semantic understanding and reasoning in targeted word-replacement tasks. Figure 2 aggregates outcomes from over 11k replacement trials, categorizing model performance into our three distinct classes (*Unchanged*, *Replaced*, and *Different*).

If parameter insufficiency were the primary failure mode, we would anticipate monotonic performance improvements with model size within a given architecture family. However, our observations contradict this expectation. Within the Qwen3 and Gemma3 families, some larger variants achieve replacement success that is comparable to or even lower than that of smaller counterparts. Moreover, even in cases where larger models behave better overall (higher *Replaced* and lower *Unchanged* rates), a non-trivial portion of outputs still fall into the *Different* category. These outcomes are consistent with a tokenizer-induced representational defect. For some word forms, the mapping from token-ID sequences to rendered text is effectively many-to-one, such that different subtoken segmentations decode to the same surface string. Consequently, models may execute legitimate substitutions at the token level (replacing one valid tokenization with an alternative) while producing no perceptible modification at the character level, cre-

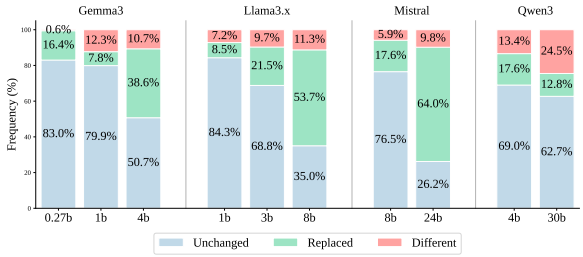

Figure 2: Distribution of outcomes across LLM families and parameter scales for the word-replacement task. The "Different" category (red) highlights tokenizer-induced phantom edits, which persist across all model sizes and families.

ating the appearance of replacement failure. The model therefore "believes" the replacement occurred due to token-ID changes, even though the decoded output remains identical. Critically, because this discrepancy originates in the tokenizer-detokenizer architecture rather than in the model's parametric representation, increasing model capacity offers no systematic solution to this fundamental misalignment.

4.2 Tokenizer-induced phantom edits

We examine token-level input-output differences within the *Different* class to identify systematic tokenizer artifacts that can mislead the model into believing it reasons correctly for word replacement. These errors arise from the non-unique representational property inherent to subword tokenization schemes, wherein a single surface string may admit multiple valid segmentations. This structural ambiguity establishes a many-to-one mapping from token ID sequences to character strings. Therefore, models may modify token IDs through boundary shifts, whitespace insertion/deletion, or alternative segmentations yet the final decoded text remains identical. We refer to such cases as phantom edits, syntactically valid token-space modifications that collapse to an unchanged surface form, obscuring the semantic equivalence across segmentations and hindering the model's ability to reason reliably about word identity. Below, we present our taxonomy of these errors.

4.2.1 Error 1: whitespace-boundary shift

A prominent error mode stems from tokenizer's vocabulary that duplicate a word with (shown in this paper as ␣) and without a leading whitespace (e.g., "February" vs. "␣February"), which are assigned distinct token IDs and thus distinct representations. In this mode, models substitute a space-prefixed token with its non-prefixed counterpart, producing a token ID change that decodes to the same surface string after whitespace normalization during detokenization.

4.2.2 Error 2: whitespace detachment/reattachment

We identify two related error modes driven by whitespace-sensitive vocabulary design. First, a token with leading space like "␣Saturday" is converted into an explicit space token together with the same word token, e.g., ["␣", "␣Saturday"], which we term **whitespace reattachment**. Second, a token such as ["␣Guy"] is split into an explicit space token and the word token without a leading space, ["␣", "Guy"], which we call **whitespace detachment**. Both errors originate from the tokenizer vocabulary's redundant encoding of words, wherein both space-prefixed and non-prefixed versions of the same word exist as distinct entries, enabling multiple valid segmentations that decode to identical surface strings.

4.2.3 Error 3: newline/whitespace substitution

We also find cases where a space-prefixed token is transformed into a newline token plus the space-free word token, as in ["␣However"]→["\n", "However"]. From the model's representational perspective, this constitutes a valid replacement operation: the output token IDs differ from the input, signaling successful task completion.

4.2.4 Error 4: intra-word resegmentation

During tokenization, the tokenizer often prefers a single vocabulary entry that fully matches the word (i.e., an unsplit token). However, because the vocabulary contains multiple subword units that can be compositionally combined to form the same lexical item, models may substitute an atomic token with a multi-token sequence comprising its constituent subpieces, unaware that this decomposition yields an identical surface string upon decoding. For instance, when replacing "␣unbelievable", the model may generate the three-token sequence ["un", "bel", "ievable"]. This transformation converts a single token ID [66917] into a sequence of distinct IDs [602, 8145, 179086], producing entirely different token identifiers and consequently divergent vector representations, despite maintaining surface-level equivalence.

4.2.5 Error 5: proper-noun segmentation ambiguity

Proper nouns and rare words are typically tokenized through subword decomposition, as atomic representations often do not exist in the vocabulary, for example, [“ Jub’, “ilee”]. During inference, models may substitute the original segmentation with alternative valid decompositions that redistribute characters across token boundaries, such as [“J”, “ub”, “ilee”]. Our analysis also reveals a systematic pattern wherein models isolate initial capital letters as separate tokens before applying subword segmentation to the remainder, for instance, transforming [“ Dorm”, “er”] into [“D”, “orm”, “er”]. Notably, while these alternative segmentations may share some common token IDs (e.g., “er”), the overall token ID sequences differ from the input, leading the model to interpret this resegmentation as a successful replacement despite producing identical decoded output.

4.2.6 Error 6: morphological boundary surfacing

Tokenizers exploit morphological regularities by representing common affixes as distinct tokens, enabling compositional word construction, for instance, segmenting [“ repaid”] as [“re”, “paid”]. However, because such morphemic pieces appear across many lexical items, they are not uniquely tied to the intended word-level meaning (e.g., “paid” in this context). Consequently, when models perform morphological resegmentation during replacement, they may erroneously interpret the combination of generic morphemic tokens as constituting semantically distinct content from the original atomic word, despite their decode-time equivalence.

4.2.7 Error 7: acronyms split

Acronyms behave differently from ordinary words because they can originate from multiple underlying expansions and often admit several valid tokenizations that detokenize to the same surface string. Since many tokenizers include subtokens for single capital letters as well as multi-letter capital sequences, the model may assemble an acronym using different capitalized units and treat this as a meaningful change in its internal reasoning, even though the decoded acronym is unchanged (e.g., “ HIV”→[“H”, “IV”]). Note that we do not expect acronyms, often uniquely grounded in a document, to be replaced with different acronyms. Instead, a reasonable replacement would be to expand them into their full phrases (like the example presented in Figure 1). However, the predominant pattern involves resegmentation among all-capital token variants, where distinct token ID sequences mislead the model into inferring it has generated new lexical content despite surface-level equivalence.

4.2.8 Error 8: plural/possessive tail tokens

The single-letter token ‘s’, like other isolated alphabetic characters, lacks inherent semantic content and functions primarily as a compositional subtoken within larger units, for example, “ smooth” may be segmented as [“s”, “mooth”]. As a consequence, plural forms and possessives may be resegmented into a base token plus the isolated “s”, producing token-ID sequences that differ substantially from the original unsplit tokenization (e.g., “ rights”→[“right”, “s”], and [“ Clement”, “s”]→[“C”, “lements”]). The resulting token-space divergence can mislead the model into treating the output as a meaningful modification, thereby reinforcing the tokenizer fault and obscuring erroneous reasoning about word identity.

This phenomenon underscores a critical tokenizer design limitation: the morphological marker ‘s’ (denoting possession or plurality) and the alphabetic character ‘s’ (functioning as a compositional subtoken in larger units) are conflated into a single vocabulary entry despite serving fundamentally different linguistic roles. This representational conflation enables the problematic resegmentations described above, as the tokenizer cannot distinguish between morphological and compositional uses of the same character sequence, permitting models to substitute one role for another while remaining unaware of the semantic distinction.

4.3 Subtokens’ realignment patterns

In this experiment, we sample non-stop words with character lengths ranging from 3 to 15 characters, mark them within news articles, and present these annotated documents to Gemma3-4B model. Our analysis centers on the “Different” class exclusively. For each instance in this class, we quantify two values: the number of subtokens produced by the tokenizer for the input target word, and the number of subtokens used for the corresponding surface word in the model output. we aggregate these pairs into the 2D grid of Figure 3. Each cell represents a specific fragmentation transition (e.g., 1→2 subtokens). We label each cell with the frequency of “Different” instances mapped to that transition and

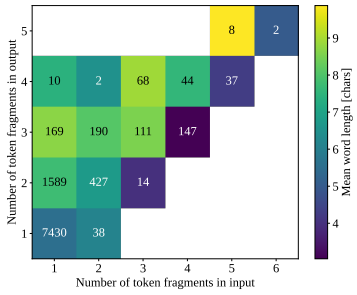

Figure 3: Heatmap of subtoken fragmentation transitions for "Different" class instances using the Gemma3-4B model. The x and y axes denote the number of token fragments in the input and output, respectively. Cell values indicate the frequency of each transition, while the color scale represents the mean character length of the words.

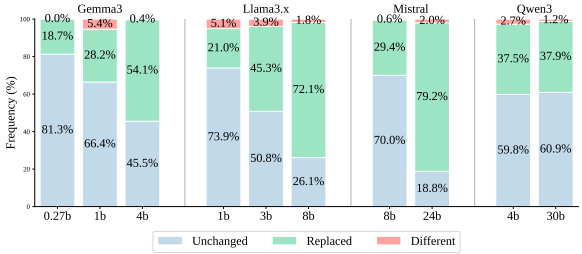

Figure 4: Distribution of outcomes across LLM families after applying the token-ID masking intervention.

use a color scale to represent the average character length of the words in that cell.

Finding 1. Cells above the diagonal line indicate that subtoken splitting (increasing subtoken count) is common while merging (decreasing subtoken count) is rare. Aggregating across the grid, we observe that approximately 78% of cases preserve the same subtoken count ($y{=}x$), about 19.7% increase the count ($y{>}x$, representing splits), and only 2.3% decrease it ($y{<}x$, representing merges). This pronounced asymmetry suggests that models more frequently edit by expanding representations than by compressing them. This pattern raises two critical questions. First, if a more concise tokenization exists, why does the tokenizer sometimes assign a more fragmented input encoding (below-diagonal cases)? Second, why does the model more frequently expand representations at inference than compress them, effectively "over-complicating" the representations (above-diagonal cases)? These patterns suggest that LLMs rarely succeed in "repairing" suboptimal segmentations, introduced upstream by the tokenizer.

Finding 2. Most input words possess atomic token representations, meaning they have unique IDs in the tokenizer vocabulary and theoretically require no subtoken decomposition. However, this theoretical property does not hold in practice: words are frequently fragmented into valid subtokens during inference despite having atomic representations available. Conversely, the existence of duplicate single-token variants causes systematic model failures. Notably, the (1→1) cell is the single most common transition, accounting for approximately 72.2% of all "Different" instances. This reveals that the majority of phantom edits preserve subtoken count while altering only token ID identity. This pattern reveals that the substantial existence of whitespace-variant token pairs (Error Type 1), tokens differing solely in leading whitespace, constitutes a primary failure mechanism. Models fail to recognize that these variants encode identical semantics, implying that their learned vector representations are not sufficiently close in embedding space to support equivalence recognition.

Finding 3. Mean character length generally increases with segmentation depth, though the relationship is not strictly monotonic. The brightest region appears near the (5→5) cell, consistent with the intuition that longer words often require more subtokens. However, certain high-token-count cells exhibit darker coloring (indicating shorter mean character length), supporting the hypothesis that token-count inflation can arise from tokenizer artifacts rather than lexical complexity. Examples include whitespace tokens, newline characters, and morpheme decompositions (with extra white spaces), non-lexical structural elements that increase token count without proportionally increasing character length.

4.4 Superficial remedy via token-ID masking

In practice, rectifying tokenizer deficiencies would require modifying tokenization rules and vocabulary, followed by re-training model so it can internalize the revised token-ID mappings. Token embeddings, output projections, and internal activation statistics are co-adapted to the original segmentation and ID assignments; thus, simply swapping token IDs or altering merge rules without re-optimization typically disrupts the learned mapping between subword units and semantics, leading to broad performance degradation. Complete tokenizer redesign and LLM re-training remain largely inaccessible in academic settings due to computational constraints. Therefore, we adopt a lightweight post-hoc intervention. We mask the

specific output token IDs that empirically trigger phantom edits in "Different" instances. This procedure does not modify the tokenizer, the embedding space, or the model's learned representations. It only constrains the decoding space so the model cannot assign probability mass to the offending tokens observed in "Different" samples, thereby forcing alternative generations. We interpret this approach as a superficial remedy, a diagnostic and mitigation layer that suppresses a known failure mode without resolving the underlying non-uniqueness and tokenizer-model co-adaptation. This intervention provides dual value: it 1) provides evidence that a meaningful fraction of errors is mediated by tokenization artifacts rather than model capacity, and 2) provides a low-cost workaround when retraining with a revised vocabulary is infeasible.

Figure 4 presents LLM performance following the blocking of problematic token IDs. Compared to the baseline shown in Figure 2, the updated distribution reveals dramatic suppression of the "Different" failure mode, with percentages declining to approximately $0-5\%$ across all models. However, the persistence of small "Different" percentages indicates that alternative token ID sequences still deceive models, which fail to recognize identical underlying semantics. This confirms that removing undesired token sequences cannot guarantee complete resolution of the issue, as the fundamental many-to-one mapping property of tokenization remains. Interestingly, blocking problematic token IDs also produces reductions in "Unchanged" rates for most LLMs. This effect is consistent with decoding dynamics: suppressing a subset of high-probability "phantom-edit" tokens forces the model to renormalize and redistribute probability mass across remaining options, which can elevate truly correct replacements and increase the chance of producing a valid substitution.

Overall, the results indicate that once tokenizer-driven artifact tokens are blocked, models more reliably produce semantically distinct substitutions. This demonstrates that models inherently possessed the reasoning capacity to generate appropriate replacements, but this capacity was systematically suppressed by the presence of high-probability tokenizer artifacts. This suggests that tokenizer-induced phantom edits constitute a "path of least resistance" in the probability landscape. When this path is blocked, models show decreased input copying ("Unchanged") and are instead forced to engage genuine semantic reasoning processes, successfully identifying alternative words that were previously overshadowed by the substantial probability mass allocated to phantom tokens.

5 Conclusion and future work

This work demonstrates that LLM reasoning deficiencies are not purely a function of capacity constraints or knowledge gaps, but are fundamentally compromised by the tokenizer layer. By exposing "phantom edits" via our consistency probe, we show that the non-injective mapping from token ID space to surface text creates a profound structural vulnerability. Models routinely operate under the illusion of successful execution because distinct token-level representations collapse into identical character strings.

A plausible next step is to test whether tokenization non-uniqueness affects reasoning behaviors beyond our replacement probe. Concretely, one could construct controlled "equivalence interventions" that replace an input's token ID sequence with an alternative sequence detokenizing to the identical surface string, then evaluate whether downstream behaviors (like chain-of-thought stability, multi-step arithmetic accuracy) exhibit sensitivity to these representationally distinct but semantically identical inputs. Such experiments would quantify the extent to which reasoning pipelines are sensitive to representational choice rather than semantics, helping to separate genuine cognitive limitations from artifacts introduced by the tokenizer's many-to-one mapping. A second research direction involves tokenizer-aware training objectives that enforce representational consistency across equivalent encodings. This can be achieved by modifying training procedures to use averaged (or pooled) representations across multiple token ID sequences corresponding to the same surface token or string, encouraging models to internalize their equivalence and reduce phantom-edit pathways. Additionally, the tokenizer can be redesigned to shrink equivalence classes, for example, introducing a dedicated $<$ `start_of_sentence` $>$ token for no-leading-space variants can eliminate duplicate forms of the same word (with vs. without leading space) and shrink the equivalence classes that enable spurious token-level edits ("token"= [$<$ `start_of_sentence` $>$,"▁token"]).

Limitations

Although our tokenization-consistency probe successfully exposes tokenizer-induced "phantom edits", the evaluation task is inherently restricted to localized word substitutions within context. While this simplified setup is advantageous for isolating tokenizer-detokenizer artifacts from parametric knowledge gaps , it does not directly measure how representational non-uniqueness propagates through or degrades complex, multi-step downstream pipelines, such as long-form generation, multi-step arithmetic, or chain-of-thought stability. Furthermore, the post-hoc token-ID masking intervention introduced in this work functions strictly as a superficial, inference-time remedy rather than a structural architectural fix. While this intervention successfully demonstrates that models retain a latent reasoning capacity that is heavily suppressed by high-probability artifact pathways , it achieves this by simply constraining the model's active decoding space. It does not resolve the fundamental, upstream non-injective mapping of the tokenizer itself, nor does it address the deep co-adaptation between the vocabulary definitions and the learned model embeddings. A thorough resolution of these representational defects necessitates a comprehensive tokenizer redesign and full model re-training, an objective that remains largely inaccessible under the typical computational constraints of academic research settings.

Acknowledgments

The authors acknowledge the use of AI tools solely for language polishing, proofreading, and paraphrasing to enhance the clarity of the text. All core concepts, experimental designs, and research results were entirely conceived and executed by the authors, and no original content or scientific contributions were generated by automated systems.

References

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, Guangyuan Piao, Nils Blach, Piotr Nyczyk, Marcin Copik, Grzegorz Kwaśniewski, Jürgen Müller, Lukas Gianinazzi, Ales Kubicek, Hubert Niewiadomski, Aidan O'Mahony, Onur Mutlu, and Torsten Hoefler. 2025. Demystifying chains, trees, and graphs of thoughts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(12):10967–10989.

Gagan Bhatia, Maxime Peyrard, and Wei Zhao. 2025. Date fragments: A hidden bottleneck of tokenization for temporal reasoning. *Preprint*, arXiv:2505.16088.

Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. Tokenization falling short: On subword robustness in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA. Association for Computational Linguistics.

Sanghyun Choo and Wonjoon Kim. 2023. A study on the evaluation of tokenizer performance in natural language processing. *Applied Artificial Intelligence*, 37(1):2175112.

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025. Toolstar: Empowering llm-brained multi-tool reasoner via reinforcement learning. *Preprint*, arXiv:2505.16410.

Meng Fang, Shilong Deng, Yudi Zhang, Zijing Shi, Ling Chen, Mykola Pechenizkiy, and Jun Wang. 2024. Large language models are neurosymbolic reasoners. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17985–17993.

Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. 2024. Test of time: A benchmark for evaluating llms on temporal reasoning. *Preprint*, arXiv:2406.09170.

Juan Luis Gastaldi, John Terilla, Luca Malagutti, Brian DuSell, Tim Vieira, and Ryan Cotterell. 2025. The foundations of tokenization: Statistical and computational concerns. *Preprint*, arXiv:2407.11606.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. 2024. Tora: A tool-integrated reasoning agent for mathematical problem solving. *Preprint*, arXiv:2309.17452.

David A. Haslett and Zhenguang G. Cai. 2025. How much semantic information is available in large language model tokens? *Transactions of the Association for Computational Linguistics*, 13:408–423.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. 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*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 43447–43478. Curran Associates, Inc.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *Preprint*, arXiv:1808.08745.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. 2025. Reasoning with latent thoughts: On the power of looped transformers. *Preprint*, arXiv:2502.17416.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *Preprint*, arXiv:2402.14903.

Kevin Slagle. 2024. Spacebyte: Towards deleting tokenization from large language modeling. In *Advances in Neural Information Processing Systems*, volume 37, pages 124925–124950. Curran Associates, Inc.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2022. Charformer: Fast character transformers via gradient-based subword tokenization. *Preprint*, arXiv:2106.12672.

Hieu Tran, Zonghai Yao, Zhichao Yang, Junda Wang, Yifan Zhang, Shuo Han, Feiyun Ouyang, and Hong Yu. 2025. RARE: Retrieval-augmented reasoning enhancement for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18305–18330, Vienna, Austria. Association for Computational Linguistics.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2023. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717–2739, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, and 12 others. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Preprint*, arXiv:1609.08144.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2024. Star: Self-taught reasoner bootstrapping reasoning with reasoning. In *Proc. the 36th International Conference on Neural Information Processing Systems*, volume 1126.