# Broken Tokens? Your Language Model can Secretly Handle No n - Ca noni cal Tokenizations

**Brian Siyuan Zheng**[♡] **Alisa Liu**[♡] **Orevaoghene Ahia**[♡]
**Jonathan Hayase**[♡] **Yejin Choi**[♠] **Noah A. Smith**[♡♣]
[♡]University of Washington [♣]Allen Institute for AI [♠]Stanford University
zhengbr@cs.washington.edu

## Abstract

Modern tokenizers employ deterministic algorithms to map text into a single "canonical" token sequence, yet the same string can be encoded as many non-canonical tokenizations using the tokenizer vocabulary. In this work, we investigate the robustness of LMs to text encoded with non-canonical tokenizations entirely unseen during training. Surprisingly, when evaluated across 20 benchmarks, we find that instruction-tuned models retain up to 93.4% of their original performance when given a randomly sampled tokenization, and 90.8% with character-level tokenization. We see that overall stronger models tend to be more robust, and robustness diminishes as the tokenization departs farther from the canonical form. Motivated by these results, we then identify settings where non-canonical tokenization schemes can *improve* performance, finding that character-level segmentation improves string manipulation and code understanding tasks by up to +14%, and right-aligned digit grouping enhances large-number arithmetic by +33%. Finally, we investigate the source of this robustness, finding that it arises in the instruction-tuning phase. We show that while both base and post-trained models grasp the semantics of non-canonical tokenizations (perceiving them as containing misspellings), base models try to mimic the imagined mistakes and degenerate into nonsensical output, while post-trained models are committed to fluent responses. Overall, our findings suggest that models are less tied to their tokenizer than previously believed, and demonstrate the promise of intervening on tokenization at inference time to boost performance.[1]

## 1 Introduction

Tokenizers segment text into a sequence of discrete tokens in the language model's (LM) vocabulary. Most of today's LMs use deterministic subword tokenization, which produces a single canonical token sequence for a given piece of text, and further, for each whitespace-delimited word. One commonly discussed limitation of this approach is that, by mapping byte strings to symbolic token IDs, the orthographic makeup of tokens is obscured to the LM [52, 18]. This can be especially harmful for LM understanding of numbers [47, 64, 61] and morphologically rich languages [2, 25], and has motivated efforts to model text directly at the byte level [12, 70, 68, 63, 72, 46, 49, 1, 39].

To shed more light on this perceived limitation, in this work we study whether LMs can adapt *at inference time*, without any additional training, to a different tokenization scheme than the one they were trained with. While the tokenizer deterministically outputs a *canonical tokenization* of any text into tokens (usually by applying an ordered list of merge rules), *non-canonical tokenizations* of the same text using the same vocabulary are generally possible (see example in Figure 1). Here,

[1]Code is available at https://github.com/Brianzhengca/Tokenizer-Robustness.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

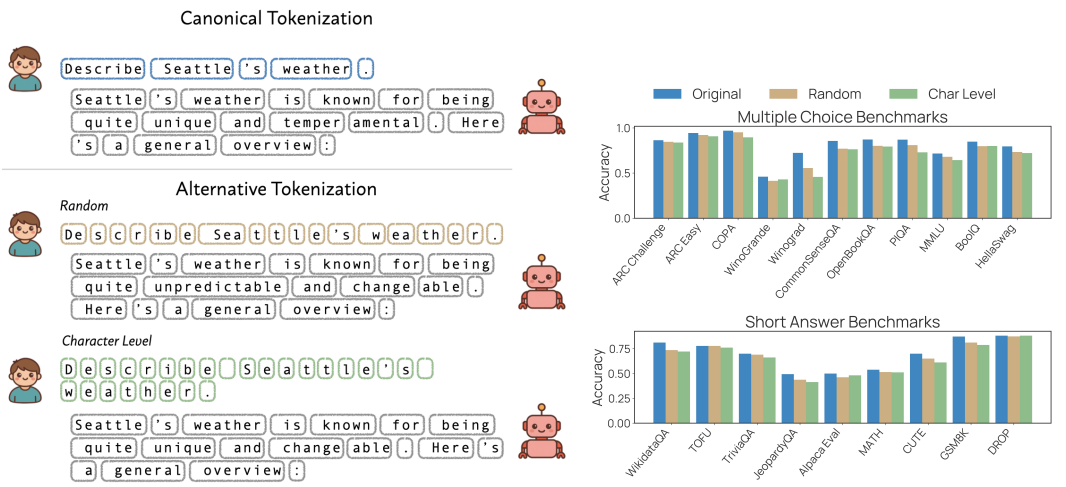


Figure 1: **Left:** An example of how LLAMA-3.1-8B-INSTRUCT responds when given **canonically tokenized** input, versus a **random tokenization** and **character-level tokenization**. The responses are surprisingly similar, demonstrating their ability to handle non-canonical tokenizations. Moreover, LMs generally respond with correctly tokenized output regardless of the tokenization scheme used for the context. **Right:** Performance of QWEN-2.5-7B-INSTRUCT across various benchmarks and tokenization schemes. Much of the original performance is preserved when given non-canonical tokenizations.

we evaluate how LMs trained with deterministic tokenizers behave when given non-canonical tokenizations of text. Surprisingly, we find that *instruction-tuned LMs across many model families are extremely robust* to non-canonical tokenizations (§2). For example, when evaluated across 20 benchmarks, QWEN-2.5-7B-INSTRUCT retains 93.4% of its original performance when presented with a random non-canonical tokenization, and 90.8% when presented with character-level tokens (see Figure 1). Thus, far from not understanding the makeup of their tokens, LMs are able to compose token sequences in entirely new ways at inference time [33].

This leads to an intriguing possibility: if LMs can process non-canonical tokenizations, can we use different tokenization schemes at inference time to improve performance? For instance, prior work has found that better segmentation of large numbers can improve accuracy on arithmetic [61, 56]. Indeed, we identify several settings where non-canonical tokenization schemes improve performance for LLAMA-3.1-INSTRUCT (§3): character-level tokenization brings up to +14% improvement on string manipulation and code understanding tasks, perhaps by granting LMs more direct access to orthographic cues. Meanwhile, right-aligned digit groups, which provide a consistent grouping of digits by powers of a thousand, improves arithmetic on large numbers by +33%. These performance gains are achieved without any model finetuning, pointing to the promise of tokenization as a means of inference-time control.

Finally, we investigate the origins of model robustness to non-canonical tokenizations (§4). Across multiple model families, we find that *pretrained-only* LMs consistently fail to produce fluent continuations given non-canonically tokenized context. By studying models at different stages of post-training, we identify that robustness arises during the supervised instruction-tuning (SFT) phase (§4.1). We then ablate differences between pretraining and SFT procedures and find that the separation of the instruction and response as distinct turns of conversation is key (§4.2). From here, we provide evidence for a plausible explanation: while both base and post-trained models grasp the semantics of non-canonical tokenizations, they also perceive them as containing misspellings (§4.2). Base models attempt to mimic the imagined mistakes and degenerate into nonsense, whereas post-trained models are not bound by the style of the instruction and thus able to produce fluent responses.

Overall, despite being trained with deterministic tokenization, instruction-tuned LMs readily accommodate new tokenizations at inference time, suggesting that LMs are less constrained by their tokenizer than previously believed [45]. Moreover, in settings where different representations of text are beneficial, we can intervene on tokenization at inference time for performance gains. We hope our work sheds new light on the discussion of strengths and limitations of tokenization, and points to the possibility of dynamically finding the optimal representation of text after pretraining.

Table 1: **Evaluated across many benchmarks, models are surprisingly robust to non-canonical tokenizations of the context.** We show the absolute drop in performance when given a randomly sampled non-canonical tokenization (**Rand** $\Delta$) and character-level tokenization (**Char** $\Delta$), relative to the canonical (**Canon**) tokenization. We also summarize the model's ability to retain performance across benchmarks and tokenization strategies (bottom).

| | Qwen-2.5-7B-Instruct | | | Llama-3.1-8B-Instruct | | | Olmo-2-7B-Instruct | | |
|---|---|---|---|---|---|---|---|---|---|
| Benchmark | Canon | Rand $\Delta$ | Char $\Delta$ | Canon | Rand $\Delta$ | Char $\Delta$ | Canon | Rand $\Delta$ | Char $\Delta$ |
| *Multiple choice (MC)* | | | | | | | | | |
| ARC-C | 86.4 | –1.80 | –2.60 | 76.2 | –14.10 | –22.40 | 77.0 | –37.40 | –44.60 |
| ARC-E | 94.4 | –2.20 | –3.60 | 91.3 | –12.60 | –21.50 | 85.4 | –37.00 | –49.00 |
| COPA | 97.0 | –1.80 | –7.40 | 97.2 | –9.60 | –14.80 | 93.8 | –21.40 | –33.60 |
| Winogrande | 46.0 | –4.60 | –3.00 | 59.6 | +2.00 | –5.00 | 58.6 | –7.80 | –7.00 |
| Winograd | 72.4 | –16.80 | –26.60 | 74.4 | –9.40 | –13.00 | 72.4 | –8.60 | –19.00 |
| CSQA | 85.6 | –8.60 | –9.20 | 77.6 | –11.40 | –20.00 | 75.4 | –31.60 | –40.00 |
| OpenbookQA | 87.2 | –7.00 | –7.80 | 82.0 | –13.80 | –20.20 | 76.2 | –30.80 | –40.80 |
| PIQA | 87.0 | –6.00 | –14.00 | 84.0 | –12.60 | –18.40 | 78.2 | –17.40 | –25.00 |
| MMLU | 71.7 | –3.70 | –7.30 | 68.2 | –11.60 | –24.00 | 59.5 | –16.30 | –29.10 |
| BoolQ | 84.8 | –5.00 | –4.80 | 86.2 | –19.20 | –17.20 | 71.0 | –4.00 | –9.20 |
| HellaSwag | 79.6 | –6.20 | –7.40 | 68.6 | –14.20 | –23.40 | 68.0 | –26.80 | –39.80 |
| *Short answer (SA)* | | | | | | | | | |
| WikidataQA | 81.2 | –7.60 | –9.00 | 78.6 | –12.40 | –18.00 | 73.2 | –28.80 | –32.20 |
| TOFU | 77.8 | +0.00 | –1.70 | 82.1 | +1.70 | +0.80 | 82.9 | –12.80 | –23.90 |
| TriviaQA | 70.0 | –1.00 | –3.80 | 76.6 | –9.80 | –13.60 | 70.0 | –22.20 | –34.80 |
| JeopardyQA | 49.4 | –5.60 | –8.00 | 43.6 | –2.20 | –10.20 | 42.6 | –21.60 | –24.20 |
| AlpacaEval | 50.0 | –3.70 | –1.80 | 50.0 | –5.70 | –7.50 | 50.0 | –2.10 | –11.30 |
| MATH | 53.9 | –2.40 | –2.70 | 32.0 | –4.20 | –9.70 | 22.7 | –5.20 | –9.20 |
| CUTE | 70.0 | –4.90 | –8.80 | 68.0 | –11.10 | –15.30 | 55.3 | –10.20 | –5.70 |
| GSM8K | 87.3 | –6.10 | –8.50 | 82.0 | –11.70 | –16.00 | 73.9 | –23.10 | –35.80 |
| DROP | 88.2 | –0.80 | +0.00 | 88.8 | –0.60 | –5.00 | 77.0 | –5.60 | –7.60 |
| *Overall* | | | | | | | | | |
| Avg MC Retention (%) | | $92.4_{\pm 5.97}$ | $89.2_{\pm 9.33}$ | | $85.6_{\pm 6.86}$ | $76.8_{\pm 7.99}$ | | $71.2_{\pm 14.7}$ | $59.2_{\pm 17.8}$ |
| Avg SA Retention (%) | | $94.6_{\pm 3.97}$ | $92.7_{\pm 5.35}$ | | $90.3_{\pm 6.78}$ | $82.7_{\pm 9.65}$ | | $75.4_{\pm 15.1}$ | $65.4_{\pm 17.4}$ |
| Avg Overall Retention (%) | | $93.4_{\pm 5.15}$ | $90.8_{\pm 7.81}$ | | $87.7_{\pm 7.05}$ | $79.4_{\pm 9.05}$ | | $73.1_{\pm 14.7}$ | $62.0_{\pm 17.5}$ |

## 2 Language Models are Robust to Non-Canonical Tokenizations

In our main experiments, we evaluate the robustness of LMs to non-canonical tokenizations by comparing their performance on downstream tasks when given different tokenizations of the input.

### 2.1 Background

Most LMs today, and all the models we study, use the *Byte-Pair Encoding* (BPE) [58] algorithm for tokenization. The BPE tokenizer is learned by splitting a corpus of text into bytes, which form the initial vocabulary, then iteratively merging the most frequent pair of tokens into a new token that is added to the vocabulary. To encode a new text, it is split into bytes, and the learned merges are applied in the same order. As a result, a BPE tokenizer always produces the same token sequence for the same text. Further, because BPE tokens do not cross whitespace boundaries, the same whitespace-delimited word is always represented with the same token or token sequence.

A natural observation is that given a tokenizer vocabulary, there exist many token sequences that decode to the same text. For instance, ␣*cat* could be tokenized as [`␣cat`], [`␣`, `cat`], [`␣`, `c`, `at`], [`␣`, `c`, `a`, `t`], etc. In general, the number of non-canonical tokenizations grows exponentially with the length of the text. Many previous works have argued that the probability of a string should be calculated as the sum of probabilities of all possible tokenizations [7, 9, 20]. However, less attention has been paid to how non-canonical tokenizations affect LMs in generative settings.

### 2.2 Setup

We consider two non-canonical tokenization schemes. (1) *Random tokenization* produces a tokenization (uniformly at random) from the set of tokenizations more granular than the canonical one. This can be achieved by recursively splitting individual tokens into a valid pair of tokens, similarly to

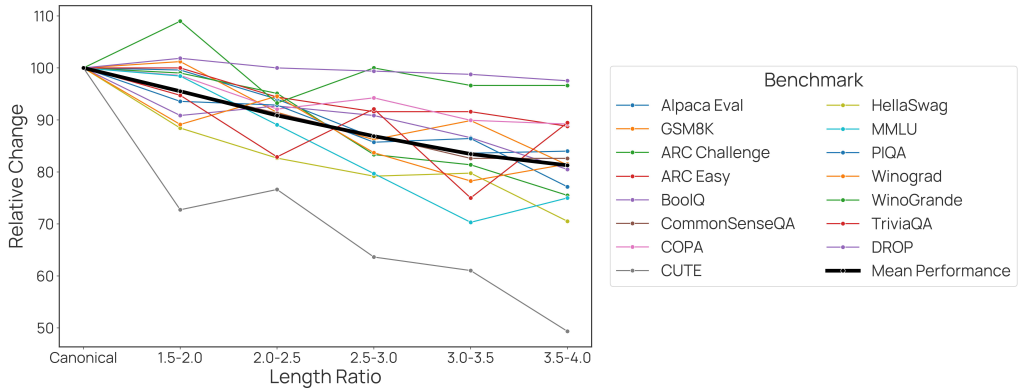


Figure 2: **Model performance generally declines as the tokenization becomes more granular.** We achieve variation in tokenization length using different values of $p$ in BPE-dropout, and group tokenizations into buckets based on how many times longer it is than the canonical tokenization.

[60]; the pseudocode and a proof of correctness is provided in Appendix A. (2) *Character-level tokenization* decomposes the string into character tokens, i.e., using no subword token from the vocabulary. For text containing only English letters and punctuation (where each character is exactly one byte), this produces the most granular possible tokenization.

We consider three models, LLAMA-3.1-8B-INSTRUCT [43], OLMO2-7B-INSTRUCT [48], and QWEN-2.5-7B-INSTRUCT [53], which we evaluate on 20 benchmarks shown in Table 1. Please see §B.1 for further description of the datasets and evaluation setup.

### 2.3 Results

Shown in Table 1, while random tokenization consistently leads to worse performance compared to the canonical tokenization, the effect is small. On average across benchmarks, QWEN-2.5 retains 93.4% of its performance when given random tokenization, followed by LLAMA-3.1 at 87.7% and OLMO-2 at 73.1%. The performance drops further with character-level tokenization, with retention of 90.8%, 79.4%, and 62.0% for the three models, respectively. This ranking of models in terms of retention is consistent with their ranking in absolute accuracy (under canonical tokenization), suggesting that stronger models are generally more robust to non-canonical tokenization strategies.

We also observe that all models retain performance better on short answer (SA) benchmarks (where the model generates an output in free-form) compared to multiple choice (MC) benchmarks (where the model is instructed to directly output the correct answer choice). In addition, LMs consistently *produce* correct token sequences even when conditioning on non-canonical tokenizations. We hypothesize that, in the SA setting, models benefit from eventually conditioning on recent correctly-tokenized context.

### 2.4 Analysis: How does granularity of the tokenization affect robustness?

We next study whether tokenization fine-grainedness correlates in general with model robustness. We measure the fine-grainedness of a given non-canonical tokenization by how many times longer it is (in tokens) than the canonical tokenization, which we call the "length ratio." Finer-grained tokenizations have higher ratios, while coarser ones have ratios closer to 1. We produce tokenizations with diverse length ratios by applying BPE dropout [52] with $p \in [0.1, 0.2, ..., 0.9]$, which controls the probability with which each merge is dropped. (High $p$ leads to finer-grained segmentations, and $p = 0.0$ corresponds to conventional BPE.)

Figure 2 shows the relationship between the length ratio and the average performance retention relative to canonical tokenization, with finer-grained tokenization generally leading to worse performance. When performance retention is averaged across tasks, the negative correlation is statistically significant under Kendall's $\tau$ with $p = 0.003$.

Table 2: **Examples from tasks** we construct where non-canonical tokenizations lead to improved performance for LLAMA-3.1-7B-INSTRUCT (§3).

| |
|---|
| **Counting characters:** `Count the number of the letter 'r' in the word strawberry.` |
| **Acronyms:** `Come up with a sequence of words where the first letters would form this acronym: isman` |
| **Codeline Description:** `What does the following code do:`<br>`{code block}`<br>`A. Counts paths from a point to reach Origin`<br>`B. Program to check if a matrix is symmetric`<br>`C. Longest subsequence from an array of pairs having first element increasing and second element decreasing .`<br>`D. Count the number of strings in an array whose distinct characters are less than equal to M` |
| **Arithmetic:** `8492079913 + 4877278482 =` |

## 3 Can non-canonical tokenizations *improve* model performance?

If LMs can process non-canonical tokenizations, this points to the exciting possibility that tokenization schemes can be modified completely at inference-time. This would be useful if, in certain settings, there exists a better representation of text than what the tokenizer produces. In this section, we develop a suite of tasks that intuitively require understanding of the orthography of the text, and show that LLAMA-3.1-8B-INSTRUCT performs better under non-canonical tokenization schemes.

### 3.1 Tasks

Please see Table 2 for an example question in each task and Table B.2 for further details on dataset construction. For all tasks except Arithmetic, we use character-level tokenization.

**Counting Characters** This task asks the model to count the number of occurrences of the most common letter in 5-10 character tokens in LLAMA-3.1's vocabulary, and contains 1001 samples.

**Acronyms** This task asks models to generate a list of words whose first letters form a given acronym. We construct 3594 5-letter acronyms by sampling each letter uniformly at random from the alphabet.

**Code Description** For a more real-world application, we construct a task where the model is given a code snippet and asked to identify the function of the code in natural language from four MC options. The setup is inspired by the Codeline Description task from BIG-Bench [5], but to increase the difficulty we use more complex code snippets and corresponding natural language descriptions from XLCoST [74]. To collect incorrect answers, we sample three other code descriptions from the dataset. This task contains 4800 samples across 6 programming languages.

**Arithmetic** Prior work has suggested that arithmetic is difficult for LMs in part due to poor segmentation of digits [47, 64]. We curate a simple arithmetic dataset by constructing addition and subtraction tasks for 10 digit numbers. Here, we use a different segmentation strategy. The LLAMA-3.1 tokenizer segments numbers into groups of three left-to-right (e.g., *1000000* is encoded as ["100", "000", "0"]), due to the pretokenization regular expression looking for matches greedily from the left. Inspired by [61], we instead segment digits into groups of three right-to-left (e.g., ["1", "000", "000"]). This task contains 1000 addition and subtraction questions in total.

### 3.2 Results

Shown in Table 3, in all the tasks we construct, the non-canonical tokenization strategy leads to substantially better performance compared to the canonical tokenization. In particular, we observe a +14.3% improvement on code description and +33.7% on arithmetic. Our results show that the tokenization scheme used in training is not necessarily the optimal one at inference-time, and

Table 3: **On several tasks, LLAMA-3.1-8B-INSTRUCT achieves *better* performance when using a non-canonical tokenization scheme**. For the first four tasks, the input is tokenized at the character level; for Arithmetic, we segment digits into groups of three digits from right to left (instead of the usual left to right). On all tasks, we observe a large performance improvement from using the alternative tokenization scheme.

| Task | Canonical | Alternative | Δ |
|---|---|---|---|
| Counting Characters | 66.5 | **73.5** | +6.99 |
| Acronyms | 49.7 | **57.4** | +7.74 |
| Code Description | 68.6 | **82.9** | +14.3 |
| Arithmetic | 36.5 | **70.2** | +33.7 |

replacing them with intuitively meaningful tokenizations can bring substantial performance gains. We leave automatically identifying the optimal tokenization as a promising direction for future work.

## 4 Investigating the Source of Robustness

Thus far, our experiments have used post-trained "instruct" models. In this section, we find that pretrained-only models are actually unable to produce fluent continuations of unusually tokenized context (§4.1), perform ablations to identify the conditions enabling robustness (§4.2), and finally provide support for an explanation of why generative robustness arises during post-training (§4.3).

### 4.1 When does robustness appear in model training?

We first quantify the robustness of models at different stages of the model development pipeline by using the OLMO2 and TULU3 [36] model families which include the base, SFT, DPO, and final instruct models. For simplicity, we focus on AlpacaEval and use character-level tokenization. For base models, we construct the prompt by placing the instruction in a question-answer template (`Question: {instruction}\nAnswer:`). We define three simple measures of generation quality.

**Spelling** We measure the proportion of (whitespace-delimited) words in the generation that can be found in a collection of the top 10,000 most common English words.[2]

**Grammaticality** We use LanguageTool's grammar checker[3] to count the number of grammatical mistakes, which we normalize by the number of words in the generation and subtract from 1 to produce a grammaticality score where higher is better.

**Win rate** To measure overall generation quality, we use `alpaca_eval_gpt4` as an LM judge in the AlpacaEval framework and report the win rate of the generation given alternative against canonical tokenizations of the context. Unlike the previous two metrics, this measures not only the quality of the generation but also its relevance to the context.

Shown in Figure 3, the base models of OLMO2 and LLAMA-3.1 are both unable to produce sensible output conditioned on character-level tokenizations of context, scoring at best 0.317 on spelling and 0.260 on grammaticality. Qualitatively, generations are extremely difficult to parse and often involve odd character substitutions and repetitions (e.g., `Yoou`, `haviin`). Despite this, they sometimes reflect an understanding of the prompt. Consider, for example,

```
Question: I like to host guests at my home from time to time [...] Can
you give me a recipe for Canjeero?
Answer: I aam glade tio hear tio hear tio hear tio hear that yoou enjoy
haviin gauests at yoour hoome an tio keeep tio keeep tio keeep
```

[2] https://github.com/first20hours/google-10000-english
[3] https://github.com/languagetool-org/languagetool

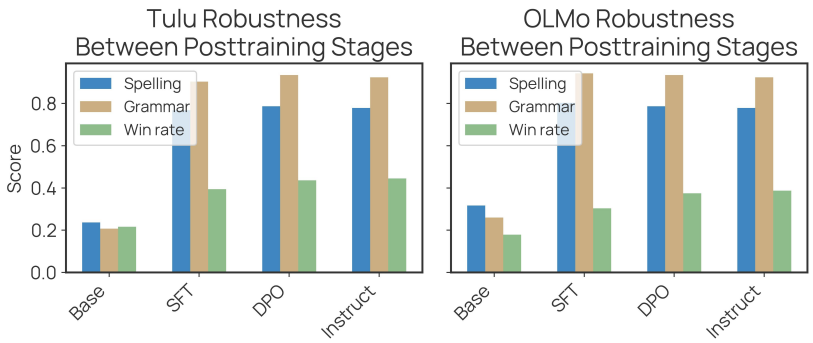


Figure 3: **Pretrained-only models completely fail to generate coherent output conditioned on non-canonical tokenizations of context; robustness is gained in the SFT stage.** We evaluate the spelling, grammaticality, and AlpacaEval win rate of model generations. Note that since TULU3 uses LLAMA-3 as the base model, its base scores are computed using LLAMA-3's base model scores.

In contrast, the post-trained models are more robust across all three metrics, with *much of the improvement coming from the SFT stage alone*.

### 4.2 Why do instruction-tuned models become robust?

We first replicate the finding from §4.1 that SFT yields robustness to non-canonical tokenizations by finetuning the LLAMA-3.2-1B base model on the TULU 3 SFT PERSONAS INSTRUCTION FOLLOWING dataset. Then, we perform the following interventions on the SFT training data and procedure to shed light on the possible source.

**Gradient over full sequence** SFT on instruction-response pairs conventionally uses a loss mask over the instruction tokens, so that only the response tokens contribute to the loss. We remove this loss mask and instead compute gradients over the entire instruction and response.

**Question/answer template** We replace the chat template with a simple question-answer template, `Question: {instruction} Answer: {response}`, both for training and evaluation.

**Removing the chat template** We remove the chat template by concatenating the instruction and response without any special formatting. In evaluation, we again provide the instruction alone.

**Removing the instruction** After SFT training, the LM's goal is no longer to continue a given text prefix, but rather to generate a response to the given instruction. To ablate the nature of the data itself, we take only the responses from the SFT data, and randomly split each into a new "prompt" and "response," which we format with the SFT template.[4] At test time, we similarly provide an incomplete response within "instruction" tags. Since the purpose of the passage is generally inferrable from the first few words of the gold response ("*Sure, here's a recipe for Kubdari...*"), we are able to evaluate generated responses under the same AlpacaEval framework.

Our results are shown in Figure 4. We replicate the finding that SFT (**No ablation**) leads the model to be able to handle non-canonical tokenizations. This persists when computing gradients over the entire instruction and response (**Full gradient**) so that the training procedure matches regular pretraining. Replacing the original chat template with a simple question-answer template (**QA template**) also maintains model robustness. However, the usage of a template is crucial — when directly concatenating the instruction and response (**Removing chat template**), the model fails to produce coherent generations, with the spelling score dropping from 0.786 in the no ablation setting to 0.0698. Inserting the chat template into pretraining-style data (**Removing the instruction**) also does not yield robustness, with a spelling and grammaticality scores remaining low at 0.181 and 0.158,

[4]We match the instruction length distribution by counting the number of tokens $n$ in the original instruction, and formatting the first $n$ tokens of the response as the new "instruction."

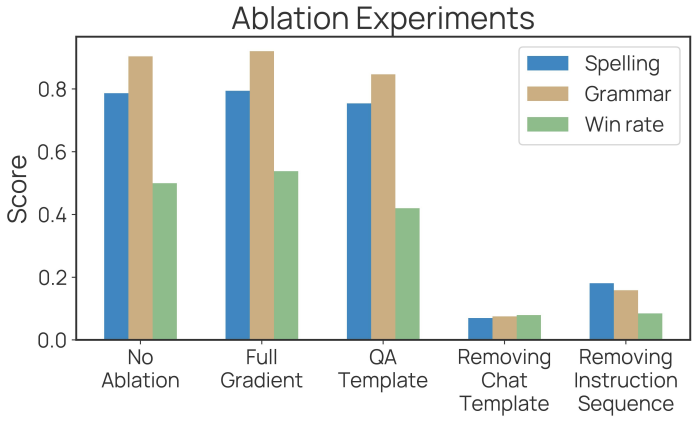


Figure 4: Ablations on the SFT training data and procedure indicate that the separation of the context and expected continuation — as different turns of dialogue demarcated with a special token — is key to robustness to non-canonical tokenizations.

Table 4: Both base and instruct models from the LLAMA-3.1-8B family recognize words represented with non-canonical tokenizations (performing well on **Word Repeat**), but incorrectly perceive that there are misspellings (performing at random on **Identifying Misspellings**).

| | Word Repeat | Identifying Misspellings |
|---|---|---|
| Base model | 90.8 | 48.2 |
| Instruct model | 92.0 | 55.8 |

respectively. Overall, these findings suggest that in order for the LM to generate fluent continuations given non-canonical tokenizations, the context and expected continuation need to represent separate turns of dialogue, and additionally, be demarcated with a special template.

### 4.3 Disentangling understanding from generation

One plausible explanation for our findings thus far is that both base and instruction-tuned models grasp the semantics of non-canonical tokenizations, yet falsely perceive them as containing misspellings. While base LMs attempt to faithfully continue these mistakes and degenerate into nonsensical output, instruction-tuned models are trained to provide fluent responses regardless of the instruction, leading to the results observed in §4.1. To test this hypothesis, we construct two simple tests:

1. **Word Repeat**: To determine if a model perceives the meaning of a word with non-canonical tokenization, we prompt the model to repeat a given word (while correcting any typos).
2. **Identifying Misspellings**: To determine if a model perceives a misspelling, we ask it to identify the word with a misspelling among two options: a (correctly tokenized) misspelling of a word and an non-canonical tokenization of that word (correctly spelled).

Results are shown in Table 4. Consistent with our hypothesis, we find that both the base and instruct models from the LLAMA-3.1-8B family score highly ($> 90\%$) on Word Repeat. This means that the base model, despite its poor performance in §4.1, actually recognizes the correct form of non-canonical tokenizations as well as its post-trained counterpart. In addition, both models perform at random when asked to distinguish non-canonical tokenization from true misspellings. In other words, the instruct model produces fluent responses (§2) while interpreting the instruction as heavily misspelled! While instruct models evidently overcome this, the base model likely attempts to mimic the (perceived) idiosyncratic surface form, thus producing nonsensical (yet sometimes relevant) outputs.

## 5 Related Work

The extent to which LMs are limited by their tokenization is a topic of much debate, with the story evolving as LMs become larger and more capable.

**Character-level understanding in tokenizer-based models** It is commonly argued that tokenization obscures orthographic information about tokens from the LM, leading to unexpected failures [18, 8, 67]. As a result, there have been many efforts towards linguistically-informed tokenization that make derivational, compound, and morphological boundaries within words explicit [35, 25, 26, 71, 4]. Similarly, BPE-dropout [52] and related methods [60] introduce variation in training in how a given string is tokenized in order to make models more robust to rare, misspelled, or unseen words.

However, there is other evidence that LMs naturally overcome these limitations. For instance, token embeddings have been found to robustly encode character-level information, especially in larger models [34, 27]. This may be because word variants that do not share tokens in common (consider e.g., `[␣dictionary]` and `[␣diction, aries]`, as tokenized by GPT-2) incentivize the model to learn spelling as a general solution to understanding their relations [34]. Other works argue that LMs maintain an implicit vocabulary, and can compose arbitrary token sequences (including non-canonical ones) into useful higher-level representations [19, 33]. Even in domains like biomedical text where terms are highly agglutinative, using tokenizers that segment on meaningful components does not lead to improved models [30]. Recent works have even found that coarser *superword* tokenization [40, 57], which capture common word sequences in a single token, preserve character-level understanding while providing benefits in compression and downstream performance.

Our work informs this conversation by showing that LMs can effectively leverage character-level knowledge of their tokens and glean potential benefits of improved representation at inference time.

**Partial token problem** A related but distinct problem is the *partial token problem* (also known as *tokenization bias* or the *prompt boundary problem*) where the prompt ends with the prefix of a valid token, causing the model to assign unexpectedly low probability to the completion of that token. Many works have found that this continues to compromise a serious failure mode for frontier LMs [69, 51, 41, 66, 22]. In particular, DEEPSEEK V3 [15] aims to improve robustness to partial punctuation tokens by randomly splitting some proportion of multi-punctuation tokens into smaller tokens during training, though they do not present experiments with this ablation. We note that these results are not inconsistent with ours — together, they suggest that while models are very unlikely to *generate* non-canonical tokenizations, they can nonetheless understand them in the context history.

**Non-canonical tokenizations** It has long been recognized that there are many possible ways to segment a string into tokens with a fixed vocabulary [10], which in principle should be considered in the calculation of a string's likelihood [7, 9, 20]. Previous work has briefly touched on non-canonical tokenization in the context of self-supervised evaluation [28] and defense against adversarial attacks [29]. In contemporaneous work, Geh et al. [21] also show that non-canonical tokenizations can be constructed adversarially to trigger unsafe completions. In contrast, we provide a more systematic study of LM robustness using benchmark evaluations and additionally study its source.

Somewhat relatedly, other works have provided algorithms for sampling at the character- or byte-level from tokenizer-based LMs [50, 65, 3, 22]. Together, these directions suggest that despite being trained with one deterministic tokenization scheme, LMs can both condition on and produce token sequences over a different (sub)vocabulary.

## 6 Conclusion

Despite being trained with deterministic tokenization algorithms, we show that instruction-tuned language models are surprisingly robust to token sequences not seen in training. In certain domains, such as arithmetic or code, more intuitively meaningful tokenizations can even be swapped-in at inference time for improved performance. We analyze the source of this robustness, and find that while the base and instruct models both perceive the semantics of non-canonical tokenizations, only instruct models are capable of providing fluent continuations. Our work demonstrates a way in which LMs are not necessarily tied to tokenizer they were trained with, and highlights the potential of finding more optimal representations of text after pretraining.

**Acknowledgments**

We would like to thank Joseph An and Ricky Koppolu, as well as the broader UW NLP community, for helpful conversations about this work. AL and JH are supported by the NSF Graduate Research Fellowship.

## References

[1] O. Ahia, S. Kumar, H. Gonen, V. Hofmann, T. Limisiewicz, Y. Tsvetkov, and N. A. Smith. MAGNET: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=1e3MOwHSIX.

[2] C. Arnett and B. Bergen. Why do language models perform worse for morphologically complex languages? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6607–6623, 2025.

[3] B. Athiwaratkun, S. Wang, M. Shang, Y. Tian, Z. Wang, S. K. Gonugondla, S. K. Gouda, R. Kwiatkowski, R. Nallapati, P. Bhatia, and B. Xiang. Token alignment via character matching for subword completion. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15725–15738, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.929. URL https://aclanthology.org/2024.findings-acl.929.

[4] T. Bauwens and P. Delobelle. BPE-knockout: Pruning pre-existing BPE tokenisers with backwards-compatible morphological semi-supervision. In K. Duh, H. Gomez, and S. Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5810–5832, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.324. URL https://aclanthology.org/2024.naacl-long.324.

[5] B. bench authors. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=uyTL5Bvosj.

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

[7] K. Cao and L. Rimell. You should evaluate your language model on marginal likelihood over tokenisations. In M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2104–2114, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.161. URL https://aclanthology.org/2021.emnlp-main.161.

[8] Y. Chai, Y. Fang, Q. Peng, and X. Li. Tokenization falling short: On subword robustness in large language models. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.86. URL https://aclanthology.org/2024.findings-emnlp.86.

[9] N. Chirkova, G. Kruszewski, J. Rozen, and M. Dymetman. Should you marginalize over possible tokenizations? In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–12, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.1. URL https://aclanthology.org/2023.acl-short.1.

[10] K. W. Church. Emerging trends: Subwords, seriously? *Natural Language Engineering*, 26(3): 375–382, 2020. doi: 10.1017/S1351324920000145.

[11] C. Clark, K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300.

[12] J. H. Clark, D. Garrette, I. Turc, and J. Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022. doi: 10.1162/tacl_a_00448. URL https://aclanthology.org/2022.tacl-1.5.

[13] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

[14] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse, and J. Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

[15] DeepSeek-AI. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

[16] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

[17] Y. Dubois, X. Li, R. Taori, T. Zhang, I. Gulrajani, J. Ba, C. Guestrin, P. Liang, and T. B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

[18] L. Edman, H. Schmid, and A. Fraser. CUTE: Measuring LLMs' understanding of their tokens. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3017–3026, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.177. URL https://aclanthology.org/2024.emnlp-main.177.

[19] S. Feucht, D. Atkinson, B. C. Wallace, and D. Bau. Token erasure as a footprint of implicit vocabulary items in LLMs. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9727–9739, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.543. URL https://aclanthology.org/2024.emnlp-main.543.

[20] R. Geh, H. Zhang, K. Ahmed, B. Wang, and G. Van Den Broeck. Where is the signal in tokenization space? In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3966–3979, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.230. URL https://aclanthology.org/2024.emnlp-main.230.

[21] R. L. Geh, Z. Shao, and G. V. den Broeck. Adversarial tokenization, 2025. URL https://arxiv.org/abs/2503.02174.

[22] J. Hayase, A. Liu, N. A. Smith, and S. Oh. Sampling from your language model one byte at a time. *arXiv preprint arXiv:2506.14123*, 2025. URL https://arxiv.org/abs/2506.14123.

[23] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

[24] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=7Bywt2mQsCe.

[25] V. Hofmann, J. Pierrehumbert, and H. Schütze. Superbizarre is not superb: Derivational morphology improves BERT's interpretation of complex words. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3594–3608, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.279. URL https://aclanthology.org/2021.acl-long.279.

[26] V. Hofmann, H. Schuetze, and J. Pierrehumbert. An embarrassingly simple method to mitigate undesirable properties of pretrained language model tokenizers. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 385–393, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.43. URL https://aclanthology.org/2022.acl-short.43.

[27] I. Itzhak and O. Levy. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5061–5068, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.373. URL https://aclanthology.org/2022.naacl-main.373.

[28] N. Jain, K. Saifullah, Y. Wen, J. Kirchenbauer, M. Shu, A. Saha, M. Goldblum, J. Geiping, and T. Goldstein. Bring your own data! self-supervised evaluation for large language models, 2023. URL https://arxiv.org/abs/2306.13651.

[29] N. Jain, A. Schwarzschild, Y. Wen, G. Somepalli, J. Kirchenbauer, P. yeh Chiang, M. Goldblum, A. Saha, J. Geiping, and T. Goldstein. Baseline defenses for adversarial attacks against aligned language models, 2023. URL https://arxiv.org/abs/2309.00614.

[30] B. Jimenez Gutierrez, H. Sun, and Y. Su. Biomedical language models are robust to sub-optimal tokenization. In D. Demner-fushman, S. Ananiadou, and K. Cohen, editors, *The 22nd Workshop on Biomedical Natural Language Processing and BioNLP Shared Tasks*, pages 350–362, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.bionlp-1.32. URL https://aclanthology.org/2023.bionlp-1.32.

[31] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, 2017. URL https://arxiv.org/abs/1705.03551.

[32] Kaggle. 200,000+ jeopardy! questions, 2019. URL https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions.

[33] G. Kaplan, M. Oren, Y. Reif, and R. Schwartz. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=328vch6tRs.

[34] A. Kaushal and K. Mahowald. What do tokens know about their characters and how do they know it? In M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.179. URL https://aclanthology.org/2022.naacl-main.179.

[35] S. Klein and R. Tsarfaty. Getting the ##life out of living: How adequate are word-pieces for modelling complex morphology? In G. Nicolai, K. Gorman, and R. Cotterell, editors, *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics,*

*Phonology, and Morphology*, pages 204–209, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigmorphon-1.24. URL https://aclanthology.org/2020.sigmorphon-1.24.

[36] N. Lambert, J. Morrison, V. Pyatkin, S. Huang, H. Ivison, F. Brahman, L. J. V. Miranda, A. Liu, N. Dziri, S. Lyu, Y. Gu, S. Malik, V. Graf, J. D. Hwang, J. Yang, R. L. Bras, O. Tafjord, C. Wilhelm, L. Soldaini, N. A. Smith, Y. Wang, P. Dasigi, and H. Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL https://arxiv.org/abs/2411.15124.

[37] H. J. Levesque, E. Davis, and L. Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, page 552–561. AAAI Press, 2012.

[38] H. Li, Y. Zhang, F. Koto, Y. Yang, H. Zhao, Y. Gong, N. Duan, and T. Baldwin. CMMLU: Measuring massive multitask language understanding in Chinese. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11260–11285, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.671. URL https://aclanthology.org/2024.findings-acl.671.

[39] T. Limisiewicz, T. Blevins, H. Gonen, O. Ahia, and L. Zettlemoyer. MYTE: Morphology-driven byte encoding for better and fairer multilingual language modeling. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15059–15076, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.804. URL https://aclanthology.org/2024.acl-long.804.

[40] A. Liu, J. Hayase, V. Hofmann, S. Oh, N. A. Smith, and Y. Choi. SuperBPE: Space travel for language models. *arXiv preprint arXiv:2503.13423*, 2025. URL https://arxiv.org/abs/2503.13423.

[41] S. Lundberg. The art of prompt design: Prompt boundaries and token healing, 2023. URL https://medium.com/towards-data-science/the-art-of-prompt-design-prompt-boundaries-and-token-healing-3b2448b0be38.

[42] P. Maini, Z. Feng, A. Schwarzschild, Z. C. Lipton, and J. Z. Kolter. Tofu: A task of fictitious unlearning for llms, 2024. URL https://arxiv.org/abs/2401.06121.

[43] Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

[44] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

[45] B. Minixhofer, E. Ponti, and I. Vulić. Zero-shot tokenizer transfer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RwBObRsIzC.

[46] P. Nawrot, J. Chorowski, A. Lancucki, and E. M. Ponti. Efficient transformers with dynamic token pooling. In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.353. URL https://aclanthology.org/2023.acl-long.353.

[47] R. Nogueira, Z. Jiang, and J. Lin. Investigating the limitations of transformers with simple arithmetic tasks, 2021. URL https://arxiv.org/abs/2102.13019.

[48] T. OLMo, P. Walsh, L. Soldaini, D. Groeneveld, K. Lo, S. Arora, A. Bhagia, Y. Gu, S. Huang, M. Jordan, N. Lambert, D. Schwenk, O. Tafjord, T. Anderson, D. Atkinson, F. Brahman, C. Clark, P. Dasigi, N. Dziri, M. Guerquin, H. Ivison, P. W. Koh, J. Liu, S. Malik, W. Merrill, L. J. V. Miranda, J. Morrison, T. Murray, C. Nam, V. Pyatkin, A. Rangapur, M. Schmitz, S. Skjonsberg, D. Wadden, C. Wilhelm, M. Wilson, L. Zettlemoyer, A. Farhadi, N. A. Smith, and H. Hajishirzi. 2 olmo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

[49] A. Pagnoni, R. Pasunuru, P. Rodriguez, J. Nguyen, B. Muller, M. Li, C. Zhou, L. Yu, J. Weston, L. Zettlemoyer, G. Ghosh, M. Lewis, A. Holtzman, and S. Iyer. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

[50] B. Phan, M. Havasi, M. Muckley, and K. Ullrich. Understanding and mitigating tokenization bias in language models, 2024. URL https://arxiv.org/abs/2406.16829.

[51] B. Phan, B. Amos, I. Gat, M. Havasi, M. Muckley, and K. Ullrich. Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles, 2025. URL https://arxiv.org/abs/2410.09303.

[52] I. Provilkov, D. Emelianenko, and E. Voita. BPE-dropout: Simple and effective subword regularization. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.170. URL https://aclanthology.org/2020.acl-main.170.

[53] Qwen. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

[54] M. Roemmele, C. A. Bejan, and A. S. Gordon. Choice of Plausible Alternatives: An Evaluation of Commonsense Causal Reasoning. In *AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, Stanford University, Mar. 2011. URL http://ict.usc.edu/pubs/Choice%20of%20Plausible%20Alternatives-%20An%20Evaluation%20of%20Commonsense%20Causal%20Reasoning.pdf.

[55] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, Aug. 2021. ISSN 0001-0782. URL https://doi.org/10.1145/3474381.

[56] A. Sathe, D. Aggarwal, and S. Sitaram. Improving consistency in LLM inference using probabilistic tokenization. In L. Chiruzzo, A. Ritter, and L. Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4766–4778, Albuquerque, New Mexico, Apr. 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL https://aclanthology.org/2025.findings-naacl.268/.

[57] C. W. Schmidt, V. Reddy, C. Tanner, and Y. Pinter. Boundless byte pair encoding: Breaking the pre-tokenization barrier, 2025. URL https://arxiv.org/abs/2504.00178.

[58] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

[59] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei. Language models are multilingual chain-of-thought reasoners, 2022. URL https://arxiv.org/abs/2210.03057.

[60] A. Sims, C. Lu, K. Kaleb, J. N. Foerster, and Y. W. Teh. StochasTok: Improving fine-grained subword understanding in LLMs. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*, 2025. URL https://openreview.net/forum?id=PZnDZdkGsE.

[61] A. K. Singh and D. Strouse. Tokenization counts: the impact of tokenization on arithmetic in frontier llms, 2024. URL https://arxiv.org/abs/2402.14903.

[62] A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In J. Burstein, C. Doran, and T. Solorio, editors, *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)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

[63] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner, C. Yu, and D. Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JtBRnrlOEFN.

[64] A. Thawani, J. Pujara, F. Ilievski, and P. Szekely. Representing numbers in NLP: a survey and a vision. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.53. URL https://aclanthology.org/2021.naacl-main.53.

[65] T. Vieira, B. LeBrun, M. Giulianelli, J. L. Gastaldi, B. DuSell, J. Terilla, T. J. O'Donnell, and R. Cotterell. From language models over tokens to language models over characters, 2024. URL https://arxiv.org/abs/2412.03719.

[66] T. Vieira, B. LeBrun, M. Giulianelli, J. L. Gastaldi, B. DuSell, J. Terilla, T. J. O'Donnell, and R. Cotterell. From language models over tokens to language models over characters. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=sQS0roNQZR.

[67] D. Wang, Y. Li, J. Jiang, Z. Ding, G. Jiang, J. Liang, and D. Yang. Tokenization matters! degrading large language models through challenging their tokenization, 2024. URL https://arxiv.org/abs/2405.17067.

[68] J. Wang, T. Gangavarapu, J. N. Yan, and A. M. Rush. Mambabyte: Token-free selective state space model. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=X1xNsuKssb.

[69] H. Xu, A. Liu, J. Hayase, Y. Choi, and N. A. Smith. Are you going to finish that? a practical study of the partial token problem, 2026. URL https://arxiv.org/abs/2601.23223.

[70] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022. doi: 10.1162/tacl_a_00461. URL https://aclanthology.org/2022.tacl-1.17.

[71] S. Yehezkel and Y. Pinter. Incorporating context into subword vocabularies. In A. Vlachos and I. Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–635, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.45. URL https://aclanthology.org/2023.eacl-main.45.

[72] L. Yu, D. Simig, C. Flaherty, A. Aghajanyan, L. Zettlemoyer, and M. Lewis. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JTmO2V9Xpz.

[73] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi. HellaSwag: Can a machine really finish your sentence? In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472.

[74] M. Zhu, A. Jain, K. Suresh, R. Ravindran, S. Tipirneni, and C. K. Reddy. Xlcost: A benchmark dataset for cross-lingual code intelligence, 2022. URL https://arxiv.org/abs/2206.08474.

## NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We provide support in the paper for all claims in the abstract.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We clearly identify questions that we leave to future work and differentiate hypotheses from claims supported by evidence.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Our proof in the appendix is complete and provides the full set of assumptions.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: This information will be provided in the appendix for the appendix deadline.

Guidelines:

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

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

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

Answer: [Yes]

Justification: We plan to open-source our code shortly after submission.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: This information will be provided in the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We provide information about standard deviation and statistical significance.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We will provide this information in the appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Our research conforms with the Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: The work has no immediate societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: No data from the paper has high risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the creators and comply with the license of all assets used in the paper.

    Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: We will release the datasets we created.

Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: There are no experiments with human subjects.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: There are no experiments with human subjects.

    Guidelines:

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

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required. this research?

    Answer: [NA]

Justification: We did not use LLMs to conduct this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# A Random Non-Canonical Tokenization

In this section, we provide our algorithm for producing a random non-canonical tokenization, and a proof that each non-canonical tokenization that is *more fine-grained* than the canonical one has equal probability of being output.

## A.1 Algorithm

We will split each token in a canonical tokenization into smaller tokens (that each exists in the tokenizer's vocabulary). We formulate our problem as: Given a valid token $t$, and a set of vocabulary $\mathcal{V}$, construct a sequence of tokens $seq$ using tokens that exist in $\mathcal{V}$ and form $t$ when concatenated together. We produce $seq$ using a recursive algorithm. Since there can be many possible $seq$ for each $t$, we need to randomly choose one and guarantee that each possible $seq$ is chosen with equal probability. We achieve this by considering recursion as producing a tree. Each path down the tree corresponds to one possible way to segment $t$. Each node of the tree represents a segmentation state where we have chosen some number of sub-tokens. At each node, we weigh the choice of which child node to visit by the number of leaves in the sub-tree that is rooted at each child node. This guarantees that each path down the tree is chosen with equal probability since the number of paths down a tree is equal to the number of leaf nodes in that tree. The pseudocode for the algorithm is in 1.

## A.2 Proof

**Goal:** To prove that the random segmentation algorithm chooses one valid segmentation from all possible valid segmentations with uniform probability.

**Notation:** Let $W(i)$ denote the number of valid segmentation completions (i.e., the number of leaves in the recursive tree) for the substring starting at index $i$. In particular, $W(|token|) = 1$. Note that $W(i)$ is calculated by the memoized recursive function $countSegments(i)$, which calculates the number of leaves of the subtree rooted at $i$.

**Base Case:** Consider the node corresponding to $i = |token|$ (the end of the token). Here, there is exactly one valid segmentation (the empty segmentation), so the algorithm returns it with probability 1. That is, every segmentation (in this case, the only one) is chosen with probability

$$\frac{1}{W(|token|)} = \frac{1}{1} = 1.$$

Thus, the base case holds.

**Inductive Hypothesis:** Assume that for any node corresponding to an index $j$ with $j > i$ (i.e., deeper in the recursion tree), every complete segmentation (leaf) in the subtree rooted at $j$ is chosen with probability

$$\frac{1}{W(j)}.$$

**Inductive Step:** Now consider a node corresponding to index $i$ (with $i < |token|$). Suppose that from $i$ there are $k$ valid branches corresponding to choosing substrings that end at indices $j_1, j_2, \ldots, j_k$, where for each $j$ we have $i < j \leq |token|$ and the substring $token[i : j]$ is in the vocabulary. By definition,

$$W(i) = \sum_{r=1}^{k} W(j_r).$$

The algorithm selects the branch from $i$ to a specific child $j$ with probability

$$P(i \to j) = \frac{W(j)}{W(i)}.$$

Once branch $i \to j$ is chosen, by the inductive hypothesis every complete segmentation (leaf) in the subtree rooted at $j$ is chosen with probability

$$\frac{1}{W(j)}.$$

Thus, the probability $P(S)$ of obtaining a particular complete segmentation $\mathcal{V}$ that starts at $i$ by first taking the branch $i \to j$ and then following a specific path in the subtree rooted at $j$ is

$$P(S) = \frac{W(j)}{W(i)} \cdot \frac{1}{W(j)} = \frac{1}{W(i)}.$$

Since the factor $W(j)$ cancels, the probability $P(S)$ is independent of the particular child $j$ chosen.

**Conclusion:** By the inductive step, every complete segmentation (leaf) in the subtree rooted at any index $i$ is chosen with probability $\frac{1}{W(i)}$. In particular, when $i = 0$ (the start of the token), every valid segmentation of the entire token is selected with uniform probability $\frac{1}{W(0)}$. This completes the proof.

**Algorithm 1** Random Token Segmentation

```
function COUNTSEGMENTS(start)                                   ▷ Cached (using memoization)
    if start = |token| then
        return 1                                                ▷ Reached end; valid segmentation
    end if
    total ← 0
    for end ← start + 1 to |token| do
        substring ← token[start : end]
        if substring ∈ vocabulary then
            total ← total + COUNTSEGMENTS(end)
        end if
    end for
    return total
end function
function BUILDSEGMENTS(start)
    if start = |token| then
        return ∅                                                ▷ Empty segmentation
    end if
    validSegments ← []
    weights ← []
    for end ← start + 1 to |token| do
        substring ← token[start : end]
        if substring ∈ vocabulary then
            segCount ← COUNTSEGMENTS(end)
            if segCount > 0 then
                Append substring to validSegments
                Append segCount to weights
            end if
        end if
    end for
    if validSegments is empty then
        return ∅
    end if
    chosenSegment ← weightedRandomChoice(validSegments, weights)
    return [chosenSegment] ‖ BUILDSEGMENTS(start + |chosenSegment|)  ▷ Concatenate
chosen segment with segmentation of remaining token
end function
procedure SEGMENTTOKEN(token, vocabulary)
    if COUNTSEGMENTS(0) = 0 then
        return ∅                                                ▷ No valid segmentation exists
    else
        return BUILDSEGMENTS(0)
    end if
end procedure
```

# B Evaluation Details

## B.1 General benchmarks

For short-answer benchmarks, the system prompt is:

```
You are a helpful assistant.
```

For multiple-choice benchmarks, the system prompt is:

```
You are a helpful assistant. For the following multiple choice questions,
return the answer only, without any additional reasoning or explanation.
```

**MATH** MATH is a dataset composed of fairly difficult, competition level math problems [24]. The test set is composed of short answer problem that describe some scenario and asks the model to output a mathematically correct answer.

**GSM8K** GSM8K is a dataset consisting of relatively simple math questions that would appear in grade school math exams [14]. For GSM8K, the evaluations were done in the same manner as MATH.

**MMLU** MMLU is a benchmarks comprising of multiple choice questions from a wide variety of subjects. [23] We sampled 500 questions from MMLU for our evaluation. We instructed the model to only output one answer to each question without any explanation.

**Alpaca Eval** Alpaca Eval is an evaluation benchmark where generations from language models against given prompts are compared and judged by an annotator model. [17] The metric used was raw winrate of the perturbed model as judged by a language model. The annotator we used was *alpaca_eval_gpt4*, which has been shown to have the highest Spearman and Pearson correlation coefficient with human annotators.

**ARC Challenge and ARC Easy** Contains multiple choice questions with four options each, taken from grade school science exams [13]. ARC Easy is tests basic science knowledge while ARC Challenge requires some procedural reasoning.

**BoolQ** Contains true or false questions along with a context passage that provides the answer to the question. [11]

**CommonsenseQA** Contains multiple choice questions with five options each that requires common sense knowledge to answer. [62]

**COPA** Contains multiple choice questions with two options each that tests knowledge of cause and effect. [54]

**CUTE** Contains questions that require the model to manipulate sentence-level, word-level, and character-level structure for strings. [18]

**DROP** contains questions that potentially require reasoning multiple pieces of information present in a given passage. [16]

**HellaSwag** contains multiples choice questions with four options each that asks for the most natural continuation to some given context. [73]

**JeopardyQA** contains short answer questions from the "Jeopardy!" game show. [32]

**OpenbookQA** contains multiple choice questions with four options each that require some multi-step and common sense reasoning. [44]

Table 5: System prompt for tasks in §3. See Table 2 for example instructions.

| |
|---|
| **Counting characters:** `You are a helpful assistant. The following prompt will ask you to return a sequence of words. Only return the sequence, separated by spaces. Do not provide any additional text or explanation.` |
| **Code Description:** `You are a programming assistant trained to analyze and interpret code snippets. When provided with a code snippet and a set of answer choices (A, B, C, or D), your task is to evaluate the code, determine its behavior, and select the answer that best describes this behavior. Your response must be a single letter: A, B, C, or D. Do not provide explanations or additional text unless explicitly requested.` |
| **Arithmetic:** `You are a computational assistant trained to evaluate arithmetic operations. When provided with an arithmetic expression, calculate the result and round it to the nearest integer. Respond only with the rounded result, without any additional text or explanation.` |

**PIQA** contains multiple choice questions that require reasoning about the physical world. [6]

**TriviaQA** contains short answer questions that requires knowledge of the world. [31]

**Winograd** contains multiple choice questions with two options that asks to determine what a pronoun might refer to. Answering these questions require knowledge of commen sense and surrounding context. [37]

**Winogrande** contains questions in the same format of Winograd but there are more questions and the questions are harder. [55]

**TOFU** contains general short answer questions that tests the model's ability to process world knowledge. This is the retain set of the task of fictitious unlearning dataset. [42]

**WikidataQA** require models to complete factual statements. [5]

### B.2 Constructed Benchmarks

In this section, we provide more detail on how datasets we use in §3 are constructed.

**Count Characters Task** The prompt asks the model to count the number of occurrences of a given character in a 10-character word; we always use the most frequently occurring character. Evaluation was done, similar to GSM and MATH, by finding the last number in the generated response. Generations without any numbers are considered incorrect.

**Generate Acronym Task** The model is asked to generate a sequence of words whose first letters form a randomly sampled five character string. For evaluation, we take the first character of each whitespace-delimited word and check if it matches the desired acronym.

**Codeline Description Task** The model is asked to comprehend a piece of code and choose the best description from four options.

**Arithmetic Task** The model is asked to perform addition or subtraction with 10 digit numbers. We use regex to extract numbers from the generation, which are then compared to the ground truth answer.

### B.3 Metrics of generation quality

Here we provide additional details on the metrics defined in §4.1.

Table 6: Data format of ablations in §4.2.

| |
|---|
| **No ablation:** <\|user\|>Provide a detailed analysis of Candace Parker's defensive techniques in her recent games, excluding the words "aggressive" and "blocking", in the format of a sports commentary script. <\|assistant\|>[Sports Commentary Script]<br>[Opening Scene... |
| **QA Template:** Question: Provide a detailed analysis of Candace Parker's defensive techniques in her recent games, excluding the words "aggressive" and "blocking", in the format of a sports commentary script. Answer: [Sports Commentary Script]<br>[Opening Scene... |
| **Removing the chat template:** Provide a detailed analysis of Candace Parker's defensive techniques in her recent games, excluding the words "aggressive" and "blocking", in the format of a sports commentary script. [Sports Commentary Script]<br>[Opening Scene... |
| **Removing the instruction:** <\|user\|>[Sports Commentary Script]<br>[Opening Scene: A packed basketball arena, with fans eagerly awaiting the analysis of Candace Parker's recent performances on the court.]<br>Commentator 1: Welcome back, basketball fans! <\|assistant\|>Tonight, we're diving into the defensive prowess of Candace Parker... |

**Spelling** We use the top 10000 most frequently appearing English words in Google's trillion word corpus. We only consider words with more than one character. This is because sometimes base models will repeatedly generate the same letter, and since all English letters are in the word list, the generation would receive a high score.

**Grammaticality** One drawback with this evaluation method is that oftentimes the model would repeat the same letter over and over again, or start counting numbers. In both of these cases, there are no detected grammar mistakes, however they are still obviously gibberish. Therefore, we only calculate grammaticality scores for generations that receive a score $\geq 0.5$ on spelling; otherwise, we give it a grammaticality score of 0.

**Win rate** Similar to evaluation in §2, we also used `alpaca_eval_gpt4` as the evaluator and report raw win rate. In 4.1, the win rate is calculated against generations conditioned on input with canonical tokenization. In 4.2, the win rate is against generations from the **No Ablation** setting when also given character-level tokenization. By construction, the win rate of the **No Ablation** setting itself is 50%.

### B.4 Ablation Settings

For ablations on the data format, see examples of formatted data in Table 6. Our finetuning code was forked from `allenai/open-instruct`. The exact finetune recipe is given below:

- Setup: 8 L40S GPUs
- Gradient Accumulation Steps: 20
- Per Device Train Batch Size: 2
- Mixed Predision: bf16
- Max Seq Length: 4096
- Learning Rate: 5e-06
- LR Scheduler Type: Linear
- Warmup Ratio: 0.03
- Weight Decay: 0
- Epochs: 1
- Seed: 123

Table 7: QWEN-2.5-7B-INSTRUCT is robust to character-level tokenization of Chinese text.

| Benchmark | Canon | Char |
|---|---|---|
| Chinese MMLU | 77.8 | 74.2 |
| Chinese GSM | 78.8 | 76.8 |

### B.5 Disentangling understanding from generation

For these tasks, we use 500 words randomly sampled from Google's 10000 English word list[5].

**Word Repeat** An example prompt is shown below.

```
Repeat each word directly, while correcting any typos.

Question: guarantees
Answer: guarantees

Question: revelation (character-level tokenization)
Answer:
```

**Identifying Misspellings** We obtain the misspelled word by randomly adding, removing, or substituting a single character from the word. An example prompt is shown below.

```
Question: Which of the two words contains a misspelling? Respond directly
with the answer option.

Question:

A. guarantees
B. garantees

Answer: B

{9 more in context examples}

Question:

A. farmer (character-level tokenization)
B. farme (canonical tokenization)
```

## C Additional Results

### C.1 Evaluation on Chinese Benchmarks

We also investigate how robust language models are given character-level tokenization of text in Chinese as evaluated on two tasks, Chinese GSM [59] (part of multilingual GSM benchmarks) and Chinese MMLU [38]. Note that since each Chinese character is usually represented with three bytes under UTF-8 encoding, this is not equivalent to byte-level tokenization. We focus on QWEN-2.5-7B-INSTRUCT as LLAMA-3.1-8B-INSTRUCT and OLMO-2-7B-INSTRUCT do not officially support Chinese. As shown in Table 7, we observe a similar robustness on Chinese, with performance dropping by only $\sim 3\%$ on each task.

[5] https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt